Microsoft Foundation Class (MFC) Quick Reference

Author: Jialong He
Email: **Jialong_he@bigfoot.com**
http://www.bigfoot.com/~jialong_he

CObject

CObject	Default constructor.
operator new	Special **new** operator.
operator delete	Special **delete** operator.
operator =	Assignment operator.
AssertValid	Validates this object's integrity.
Dump	Produces a diagnostic dump of this object.
IsSerializable	Tests to see whether this object can be serialized.
Serialize	Loads or stores an object from/to an archive.
GetRuntimeClass	Returns the **CRuntimeClass** structure corresponding to this object's class.
IsKindOf	Tests this object's relationship to a given class.

CWinApp

m_pszAppName	Specifies the name of the application.
m_hInstance	Identifies the current instance of the application.
m_hPrevInstance	Set to **NULL** in a 32-bit application.
m_lpCmdLine	Points to a null-terminated string that specifies the command line for the application.
m_nCmdShow	Specifies how the window is to be shown initially.
m_bHelpMode	Indicates if the user is in Help context mode (typically invoked with SHIFT+F1).
m_pActiveWnd	Pointer to the main window of the container application when an OLE server is in-place active.
m_pszExeName	The module name of the application.
m_pszHelpFilePath	The path to the application's Help file.
m_pszProfileName	The application's .INI filename.
m_pszRegistryKey	Used to determine the full registry key for storing application profile settings.
CWinApp	Constructs a **CWinApp** object.
LoadCursor	Loads a cursor resource.
LoadStandardCursor	Loads a Windows predefined cursor that the **IDC_** constants specify in WINDOWS.H.
LoadOEMCursor	Loads a Windows OEM predefined cursor that the **OCR_** constants specify in WINDOWS.H.
LoadIcon	Loads an icon resource.
LoadStandardIcon	Loads a Windows predefined icon that the **IDI_** constants specify in WINDOWS.H.
LoadOEMIcon	Loads a Windows OEM predefined icon that the **OIC_** constants specify in WINDOWS.H.
RunAutomated	Tests the application's command line for the **/Automation** option. Obsolete. Use the value in CCommandLineInfo::m_bRunEmbedded after calling ParseCommandLine. instead.
RunEmbedded	Tests the application's command line for the **/Embedding** option. Obsolete. Use the value in CCommandLineInfo::m_bRunEmbedded after calling ParseCommandLine. instead.
ParseCommandLine	Parses individual parameters and flags in the command line.
ProcessShellCommand	Handles command-line arguments and flags.
GetProfileInt	Retrieves an integer from an entry in the application's .INI file.
WriteProfileInt	Writes an integer to an entry in the application's .INI file.
GetProfileString	Retrieves a string from an entry in the application's .INI file.
WriteProfileString	Writes a string to an entry in the application's .INI file.
AddDocTemplate	Adds a document template to the application's list of available document templates.
GetFirstDocTemplatePosition	Retrieves the position of the first document template.
GetNextDocTemplate	Retrieves the position of a document template. Can be used recursively.
OpenDocumentFile	Called by the framework to open a document from a file.
AddToRecentFileList	Adds a filename to the most recently used (MRU) file list.
SelectPrinter	Selects a printer previously indicated by a user through a print dialog box.
CreatePrinterDC	Creates a printer device context.
GetPrinterDeviceDefaults	Retrieves the printer device defaults.
InitInstance	Override to perform Windows instance initialization, such as creating your window objects.
Run	Runs the default message loop. Override to customize the message loop.
OnIdle	Override to perform application-specific idle-time processing.
ExitInstance	Override to clean up when your application terminates.
HideApplication	Hides the application before closing all documents.
CloseAllDocuments	Closes all open documents.
PreTranslateMessage	Filters messages before they are dispatched to the Windows functions ::TranslateMessage and ::DispatchMessage.
SaveAllModified	Prompts the user to save all modified documents.
DoMessageBox	Implements AfxMessageBox for the application.
ProcessMessageFilter	Intercepts certain messages before they reach the application.
ProcessWndProcException	Intercepts all unhandled exceptions thrown by the application's message and command handlers.
DoWaitCursor	Turns the wait cursor on and off.
OnDDECommand	Called by the framework in response to a dynamic data exchange (DDE) execute command.
WinHelp	Calls the **WinHelp** Windows function.
LoadStdProfileSettings	Loads standard .INI file settings and enables the MRU file list feature.
SetDialogBkColor	Sets the default background color for dialog boxes and message boxes.
SetRegistryKey	Causes application settings to be stored in the registry instead of .INI files.
EnableShellOpen	Allows the user to open data files from the Windows File Manager.
RegisterShellFileTypes	Registers all the application's document types with the Windows File Manager.
Enable3dControls	Enables controls with three-dimensional appearance.
Enable3dControlsStatic	Enables controls with a three-dimensional appearance.
OnFileNew	Implements the **ID_FILE_NEW** command.
OnFileOpen	Implements the **ID_FILE_OPEN** command.
OnFilePrintSetup	Implements the **ID_FILE_PRINT_SETUP** command.
OnContextHelp	Handles SHIFT+F1 Help within the application.
OnHelp	Handles F1 Help within the application (using the current context).
OnHelpIndex	Handles the **ID_HELP_INDEX** command and provides a default Help topic.
OnHelpFinder	Handles the **ID_HELP_FINDER** and **ID_DEFAULT_HELP** commands.
OnHelpUsing	Handles the **ID_HELP_USING** command.

CDocument

CDocument	Constructs a **CDocument** object.
AddView	Attaches a view to the document.
GetDocTemplate	Returns a pointer to the document template that describes the type of the document.
GetFirstViewPosition	Returns the position of the first in the list of views; used to begin iteration.
GetNextView	Iterates through the list of views associated with the document.
GetPathName	Returns the path of the document's data file.
GetTitle	Returns the document's title.
IsModified	Indicates whether the document has been modified since it was last saved.
RemoveView	Detaches a view from the document.
SetModifiedFlag	Sets a flag indicating that you have modified the document since it was last saved.
SetPathName	Sets the path of the data file used by the document.
SetTitle	Sets the document's title.
UpdateAllViews	Notifies all views that document has been modified.
CanCloseFrame	Advanced overridable; called before closing a frame window viewing this document.
DeleteContents	Called to perform cleanup of the document.
OnChangedViewList	Called after a view is added to or removed from the document.
OnCloseDocument	Called to close the document.
OnNewDocument	Called to create a new document.
OnOpenDocument	Called to open an existing document.
OnSaveDocument	Called to save the document to disk.
ReportSaveLoadException	Advanced overridable; called when an open or save operation cannot be completed because of an exception.
GetFile	Returns a pointer to the desired **CFile** object.
ReleaseFile	Releases a file to make it available for use by other applications.
SaveModified	Advanced overridable; called to ask the user

	whether the document should be saved.
PreCloseFrame	Called before the frame window is closed.
OnFileSendMail	Sends a mail message with the document attached.
OnUpdateFileSendMail	Enables the Send Mail command if mail support is present.

CView

DoPreparePrinting	Displays Print dialog box and creates printer device context; call when overriding the **OnPreparePrinting** member function.
GetDocument	Returns the document associated with the view.
OnDragEnter	Called when an item is first dragged into the drag-and-drop region of a view.
OnDragLeave	Called when a dragged item leaves the drag-and-drop region of a view.
OnDragOver	Called when an item is dragged over the drag-and-drop region of a view.
OnDrop	Called when an item has been dropped into the drag-and-drop region of a view, default handler.
OnDropEx	Called when an item has been dropped into the drag-and-drop region of a view, primary handler.
OnDragScroll	Called to determine whether the cursor is dragged into the scroll region of the window.
OnInitialUpdate	Called after a view is first attached to a document.
OnScrollBy	Called when a view containing active in-place OLE items is scrolled.
OnScroll	Called when OLE items are dragged beyond the borders of the view.
IsSelected	Tests whether a document item is selected. Required for OLE support.
OnActivateView	Called when a view is activated.
OnActivateFrame	Called when the frame window containing the view is activated or deactivated.
OnBeginPrinting	Called when a print job begins; override to allocate graphics device interface (GDI) resources.
OnDraw	Called to render an image of the document for screen display, printing, or print preview. Implementation required.
OnEndPrinting	Called when a print job ends; override to deallocate GDI resources.
OnEndPrintPreview	Called when preview mode is exited.
OnPrepareDC	Called before the **OnDraw** member function is called for screen display or the **OnPrint** member function is called for printing or print preview.
OnPreparePrinting	Called before a document is printed or previewed; override to initialize Print dialog box.
OnPrint	Called to print or preview a page of the document.
OnUpdate	Called to notify a view that its document has been modified.
CView	Constructs a **CView** object.

CFrameWnd

m_bAutoMenuEnable	Controls automatic enable and disable functionality for menu items.
rectDefault	Pass this static **CRect** as a parameter when creating a **CFrameWnd** object to allow Windows to choose the window's initial size and position.
CFrameWnd	Constructs a **CFrameWnd** object.
Create	Call to create and initialize the Windows frame window associated with the **CFrameWnd** object.
LoadFrame	Call to dynamically create a frame window from resource information.
LoadAccelTable	Call to load an accelerator table.
LoadBarState	Call to restore control bar settings.
SaveBarState	Call to save control bar settings.
ShowControlBar	Call to show the control bar.
SetDockState	Call to dock the frame window in the main window.
GetDockState	Retrieves the dock state of a frame window.
ActivateFrame	Makes the frame visible and available to the user.
InitialUpdateFrame	Causes the **OnInitialUpdate** member function belonging to all views in the frame window to be called.
GetActiveFrame	Returns the active **CFrameWnd** object.
SetActiveView	Sets the active **CView** object.
GetActiveView	Returns the active **CView** object.
CreateView	Creates a view within a frame that is not derived from **CView**.
GetActiveDocument	Returns the active **CDocument** object.
GetControlBar	Retrieves the control bar.
GetMessageString	Retrieves message corresponding to a command ID.
IsTracking	Determines if splitter bar is currently being moved.
SetMessageText	Sets the text of a standard status bar.
EnableDocking	Allows a control bar to be docked.
DockControlBar	Docks a control bar.
FloatControlBar	Floats a control bar.
BeginModalState	Sets the frame window to modal.
EndModalState	Ends the frame window's modal state. Enables all of the windows disabled by **BeginModalState** .
InModalState	Returns a value indicating whether or not a frame window is in a modal state.
ShowOwnedWindows	Shows all windows that are descendants of the **CFrameWnd** object.
RecalcLayout	Repositions the control bars of the **CFrameWnd** object.
OnCreateClient	Creates a client window for the frame.
OnSetPreviewMode	Sets the application's main frame window into and out of print-preview mode.
GetMessageBar	Returns a pointer to the status bar belonging to the frame window.
NegotiateBorderSpace	Negotiates border space in the frame window.
OnContextHelp	Handles SHIFT+F1 Help for in-place items.

CScrollView

CScrollView	Constructs a **CScrollView** object.
FillOutsideRect	Fills the area of a view outside the scrolling area.
GetDeviceScrollPosition	Gets the current scroll position in device units.
GetDeviceScrollSizes	Gets t he current mapping mode, the total size, and the line and page sizes of the scrollable view. Sizes are in device units.
GetScrollPosition	Gets the current scroll position in logical units.
GetTotalSize	Gets the total size of the scroll view in logical units.
ResizeParentToFit	Causes the size of the view to dictate the size of its frame.
ScrollToPosition	Scrolls the view to a given point, specified in logical units.
SetScaleToFitSize	Puts the scroll view into scale-to-fit mode.
SetScrollSizes	Sets the scroll view's mapping mode, total size, and horizontal and vertical scroll amounts.

CWnd

m_hWnd	Indicates the **HWND** attached to this **CWnd**.
CWnd	Constructs a **CWnd** object.
DestroyWindow	Destroys the attached Windows window.
Create	Creates and initializes the child window associated with the **CWnd** object.
PreCreateWindow	Called before the creation of the Windows window attached to this **CWnd** object.
CalcWindowRect	Called to calculate the window rectangle from the client rectangle.
GetStyle	Returns the current window style.
GetExStyle	Returns the window's extended style.
Attach	Attaches a Windows handle to a **CWnd** object.
Detach	Detaches a Windows handle from a **CWnd** object and returns the handle.
PreSubclassWindow	Allows other necessary subclassing to occur before SubclassWindow is called.
SubclassWindow	Attaches a window to a **CWnd** object and makes it route messages through the **CWnd**'s message map.
UnsubclassWindow	Detaches a window from a **CWnd** object
FromHandle	Returns a pointer to a **CWnd** object when given a handle to a window. If a **CWnd** object is not attached to the handle, a temporary **CWnd** object is created and attached.
FromHandlePermanent	Returns a pointer to a **CWnd** object when given a handle to a window. If a **CWnd** object is not attached to the handle, **NULL** is returned.
DeleteTempMap	Called automatically by the **CWinApp** idle-time handler and deletes any temporary **CWnd** objects created by **FromHandle**.
GetSafeHwnd	Returns **m_hWnd**, or **NULL** if the **this** pointer is **NULL**.
CreateEx	Creates a Windows overlapped, pop-up, or child window and attaches it to a **CWnd** object.
CreateControl	Create an OLE control that will be represented in an MFC program by a **CWnd** object.
IsWindowEnabled	Determines whether the window is enabled for mouse and keyboard input.
EnableWindow	Enables or disables mouse and keyboard input.

GetActiveWindow	Retrieves the active window.
SetActiveWindow	Activates the window.
GetCapture	Retrieves the **CWnd** that has the mouse capture.
SetCapture	Causes all subsequent mouse input to be sent to the **CWnd**.
GetFocus	Retrieves the **CWnd** that currently has the input focus.
SetFocus	Claims the input focus.
GetDesktopWindow	Retrieves the Windows desktop window.
GetForegroundWindow	Returns a pointer to the foreground window (the top-level window with which the user is currently working).
SetForegroundWindow	Puts the thread that created the window into the foreground and activates the window.
GetIcon	Retrieves the handle to an icon.
SetIcon	Sets the handle to a specific icon.
GetWindowContextHelpId	Retrieves the help context identifier.
SetWindowContextHelpId	Sets the help context identifier.
ModifyStyle	Modifies the current window style.
ModifyStyleEx	Modifies the window's extended style.
GetWindowPlacement	Retrieves the show state and the normal (restored), minimized, and maximized positions of a window.
SetWindowPlacement	Sets the show state and the normal (restored), minimized, and maximized positions for a window.
GetWindowRgn	Retrieves a copy of the window region of a window.
SetWindowRgn	Sets the region of a window.
IsIconic	Determines whether **CWnd** is minimized (iconic).
IsZoomed	Determines whether **CWnd** is maximized.
MoveWindow	Changes the position and dimensions of **CWnd**.
SetWindowPos	Changes the size, position, and ordering of child, pop-up, and top-level windows.
ArrangeIconicWindows	Arranges all the minimized (iconic) child windows.
BringWindowToTop	Brings **CWnd** to the top of a stack of overlapping windows.
GetWindowRect	Gets the screen coordinates of **CWnd**.
GetClientRect	Gets the dimensions of the **CWnd** client area.
ChildWindowFromPoint	Determines which, if any, of the child windows contains the specified point.
FindWindow	Returns the handle of the window, which is identified by its window name and window class.
GetNextWindow	Returns the next (or previous) window in the window manager's list.
GetOwner	Retrieves a pointer to the owner of a **CWnd**.
SetOwner	Changes the owner of a **CWnd**.
GetTopWindow	Returns the first child window that belongs to the **CWnd**.
GetWindow	Returns the window with the specified relationship to this window.
GetLastActivePopup	Determines which pop-up window owned by **CWnd** was most recently active.
IsChild	Indicates whether **CWnd** is a child window or other direct descendant of the specified window.
GetParent	Retrieves the parent window of **CWnd** (if any).
GetSafeOwner	Retrieves the safe owner for the given window.
SetParent	Changes the parent window.
WindowFromPoint	Identifies the window that contains the given point.
GetDlgItem	Retrieves the control with the specified ID from the specified dialog box.
GetDlgCtrlID	If the **CWnd** is a child window, calling this function returns its ID value.
GetDescendantWindow	Searches all descendant windows and returns the window with the specified ID.
GetParentFrame	Retrieves the **CWnd** object's parent frame window.
SendMessageToDescendants	Sends a message to all descendant windows of the window.
GetTopLevelParent	Retrieves the window's top-level parent.
GetTopLevelOwner	Retrieves the top-level window.
GetParentOwner	Returns a pointer to a child window's parent window.
GetTopLevelFrame	Retrieves the window's top-level frame window.
UpdateDialogControls	Call to update the state of dialog buttons and other controls.
UpdateData	Initializes or retrieves data from a dialog box.
CenterWindow	Centers a window relative to its parent.
BeginPaint	Prepares **CWnd** for painting.
EndPaint	Marks the end of painting.
Print	Draws the current window in the specified device context.
PrintClient	Draws any window in the specified device context (usually a printer device context).
LockWindowUpdate	Disables or reenables drawing in the given window.
UnlockWindowUpdate	Unlocks a window that was locked with **CWnd::LockWindowUpdate**.
GetDC	Retrieves a display context for the client area.
GetDCEx	Retrieves a display context for the client area, and enables clipping while drawing.
RedrawWindow	Updates the specified rectangle or region in the client area.
GetWindowDC	Retrieves the display context for the whole window, including the caption bar, menus, and scroll bars.
ReleaseDC	Releases client and window device contexts, freeing them for use by other applications.
UpdateWindow	Updates the client area.
SetRedraw	Allows changes in **CWnd** to be redrawn or prevents changes from being redrawn.
GetUpdateRect	Retrieves the coordinates of the smallest rectangle that completely encloses the **CWnd** update region.
GetUpdateRgn	Retrieves the **CWnd** update region.
Invalidate	Invalidates the entire client area.
InvalidateRect	Invalidates the client area within the given rectangle by adding that rectangle to the current update region.
InvalidateRgn	Invalidates the client area within the given region by adding that region to the current update region.
ValidateRect	Validates the client area within the given rectangle by removing the rectangle from the current update region.
ValidateRgn	Validates the client area within the given region by removing the region from the current update region.
ShowWindow	Shows or hides the window.
IsWindowVisible	Determines whether the window is visible.
ShowOwnedPopups	Shows or hides all pop-up windows owned by the window.
EnableScrollBar	Enables or disables one or both arrows of a scroll bar.
MapWindowPoints	Converts (maps) a set of points from the coordinate space of the **CWnd** to the coordinate space of another window.
ClientToScreen	Converts the client coordinates of a given point or rectangle on the display to screen coordinates.
ScreenToClient	Converts the screen coordinates of a given point or rectangle on the display to client coordinates.
SetWindowText	Sets the window text or caption title (if it has one) to the specified text.
GetWindowText	Returns the window text or caption title (if it has one).
GetWindowTextLength	Returns the length of the window's text or caption title.
SetFont	Sets the current font.
GetFont	Retrieves the current font.
GetScrollPos	Retrieves the current position of a scroll box.
GetScrollRange	Copies the current minimum and maximum scroll-bar positions for the given scroll bar.
ScrollWindow	Scrolls the contents of the client area.
ScrollWindowEx	Scrolls the contents of the client area. Similar to **ScrollWindow**, with additional features.
GetScrollInfo	Retrieves the information that the **SCROLLINFO** structure maintains about a scroll bar.
GetScrollLimit	Retrieves the limit of the scroll bar.
SetScrollInfo	Sets information about the scroll bar.
SetScrollPos	Sets the current position of a scroll box and, if specified, redraws the scroll bar to reflect the new position.
SetScrollRange	Sets minimum and maximum position values for the given scroll bar.
ShowScrollBar	Displays or hides a scroll bar.
EnableScrollBarCtrl	Enables or disables a sibling scroll-bar control.
GetScrollBarCtrl	Returns a sibling scroll-bar control.
RepositionBars	Repositions control bars in the client area.
DragAcceptFiles	Indicates the window will accept dragged files.
CreateCaret	Creates a new shape for the system caret and gets ownership of the caret.
CreateSolidCaret	Creates a solid block for the system caret and gets ownership of the caret.
CreateGrayCaret	Creates a gray block for the system caret and gets ownership of the caret.
GetCaretPos	Retrieves the client coordinates of the caret's current position.
SetCaretPos	Moves the caret to a specified position.
HideCaret	Hides the caret by removing it from the display screen.

ShowCaret	Shows the caret on the display at the caret's current position. Once shown, the caret begins flashing automatically.
CheckDlgButton	Places a check mark next to or removes a check mark from a button control.
CheckRadioButton	Checks the specified radio button and removes the check mark from all other radio buttons in the specified group of buttons.
GetCheckedRadioButton	Returns the ID of the currently checked radio button in a group of buttons.
DlgDirList	Fills a list box with a file or directory listing.
DlgDirListComboBox	Fills the list box of a combo box with a file or directory listing.
DlgDirSelect	Retrieves the current selection from a list box.
DlgDirSelectComboBox	Retrieves the current selection from the list box of a combo box.
GetDlgItemInt	Translates the text of a control in the given dialog box to an integer value.
GetDlgItemText	Retrieves the caption or text associated with a control.
GetNextDlgGroupItem	Searches for the next (or previous) control within a group of controls.
GetNextDlgTabItem	Retrieves the first control with the WS_TABSTOP style that follows (or precedes) the specified control.
IsDlgButtonChecked	Determines whether a button control is checked.
IsDialogMessage	Determines whether the given message is intended for the modeless dialog box and, if so, processes it.
SendDlgItemMessage	Sends a message to the specified control.
SetDlgItemInt	Sets the text of a control to the string that represents an integer value.
SetDlgItemText	Sets the caption or text of a control in the specified dialog box.
SubclassDlgItem	Attaches a Windows control to a **CWnd** object and makes it route messages through the **CWnd**'s message map.
ExecuteDlgInit	Initiates a dialog resource.
RunModalLoop	Retrieves, translates, or dispatches messages for a window that is in modal status.
ContinueModal	Continues a window's modal status.
EndModalLoop	Ends a window's modal status.
BindDefaultProperty	Binds the calling object's default simple bound property, as marked in the type library, to a cursor associated with a data-source control.
BindProperty	Binds a cursour-bound property on a data-bound control to a data-source control and registers that relationship with the MFC binding manager.
GetDSCCursor	Retrieves a pointer to the underlying cursor that is defined by the DataSource, UserName, Password, and SQL properties of a data-source control.
GetMenu	Retrieves a pointer to the specified menu.
SetMenu	Sets the menu to the specified menu.
DrawMenuBar	Redraws the menu bar.
GetSystemMenu	Allows the application to access the Control menu for copying and modification.
HiliteMenuItem	Highlights or removes the highlighting from a top-level (menu-bar) menu item.
EnableToolTips	Enables the tooltip control.
CancelToolTips	Disables the tooltip control.
FilterToolTipMessage	Retrieves the title or text associated with a control in a dialog box.
OnToolHitTest	Detemines whether a point is in the bounding rectangle of the specified tool and retrieves information about the tool.
SetTimer	Installs a system timer that sends a WM_TIMER message when triggered.
KillTimer	Kills a system timer.
FlashWindow	Flashes the window once.
MessageBox	Creates and displays a window that contains an application-supplied message and caption.
GetCurrentMessage	Returns a pointer to the message this window is currently processing. Should only be called when in an **On**_Message_ message-handler member function.
Default	Calls the default window procedure, which provides default processing for any window messages that an application does not process.
PreTranslateMessage	Used by **CWinApp** to filter window messages before they are dispatched to the **TranslateMessage** and **DispatchMessage** Windows functions.
SendMessage	Sends a message to the **CWnd** object and does not return until it has processed the message.
PostMessage	Places a message in the application queue, then returns without waiting for the window to process the message.
SendNotifyMessage	Sends the specified message to the window and returns as soon as possible, depending on whether the calling thread created the window.
ChangeClipboardChain	Removes **CWnd** from the chain of Clipboard viewers.
SetClipboardViewer	Adds **CWnd** to the chain of windows that are notified whenever the contents of the Clipboard are changed.
OpenClipboard	Opens the Clipboard. Other applications will not be able to modify the Clipboard until the Windows CloseClipboard function is called.
GetClipboardOwner	Retrieves a pointer to the current owner of the Clipboard.
GetOpenClipboardWindow	Retrieves a pointer to the window that currently has the Clipboard open.
GetClipboardViewer	Retrieves a pointer to the first window in the chain of Clipboard viewers.
SetProperty	Sets an OLE control property.
OnAmbientProperty	Implement ambient property values.
GetControlUnknown	Retrieves a pointer to an unknown OLE control.
GetProperty	Retrieves an OLE control property.
InvokeHelper	Invokes an OLE control method or property.
WindowProc	Provides a window procedure for a **CWnd**. The default dispatches messages through the message map.
DefWindowProc	Calls the default window procedure, which provides default processing for any window messages that an application does not process.
PostNcDestroy	This virtual function is called by the default OnNcDestroy function after the window has been destroyed.
OnNotify	Called by the framework to inform a parent window an event has occurred in one of its controls or that the control needs information.
OnChildNotify	Called by a parent window to give a notifying control a chance to respond to a control notification.
DoDataExchange	For dialog data exchange and validation. Called by **UpdateData**.
OnInitMenu	Called when a menu is about to become active.
OnInitMenuPopup	Called when a pop-up menu is about to become active.
OnSysChar	Called when a keystroke translates to a system character.
OnSysCommand	Called when the user selects a command from the Control menu, or when the user selects the Maximize or Minimize button.
OnSysDeadChar	Called when a keystroke translates to a system dead character (such as accent characters).
OnSysKeyDown	Called when the user holds down the ALT key and then presses another key.
OnSysKeyUp	Called when the user releases a key that was pressed while the ALT key was held down.
OnCompacting	Called when Windows detects that system memory is low.
OnDevModeChange	Called for all top-level windows when the user changes device-mode settings.
OnFontChange	Called when the pool of font resources changes.
OnPaletteIsChanging	Informs other applications when an application is going to realize its logical palette.
OnPaletteChanged	Called to allow windows that use a color palette to realize their logical palettes and update their client areas.
OnSysColorChange	Called for all top-level windows when a change is made in the system color setting.
OnWindowPosChanging	Called when the size, position, or Z-order is about to change as a result of a call to SetWindowPos or another window-management function.
OnWindowPosChanged	Called when the size, position, or Z-order has changed as a result of a call to SetWindowPos or another window-management function.
OnDropFiles	Called when the user releases the left mouse button over a window that has registered itself as the recipient of dropped files.
OnSpoolerStatus	Called from Print Manager whenever a job is added to or removed from the Print Manager queue.
OnTimeChange	Called for all top-level windows after the system time changes.
OnWinIniChange	Called for all top-level windows after the Windows initialization file, WIN.INI, is changed.
OnCommand	Called when the user selects a command.
OnActivate	Called when **CWnd** is being activated or deactivated.
OnActivateApp	Called when the application is about to be activated or deactivated.
OnCancelMode	Called to allow **CWnd** to cancel any internal modes, such as mouse capture.

OnChildActivate	Called for multiple document interface (MDI) child windows whenever the size or position of **CWnd** changes or **CWnd** is activated.
OnClose	Called as a signal that **CWnd** should be closed.
OnCopyData	Copies data from one application to another.
OnCreate	Called as a part of window creation.
OnCtlColor	Called if **CWnd** is the parent of a control when the control is about to be drawn.
OnDestroy	Called when **CWnd** is being destroyed.
OnEnable	Called when **CWnd** is enabled or disabled.
OnEndSession	Called when the session is ending.
OnEnterIdle	Called to inform an application's main window procedure that a modal dialog box or a menu is entering an idle state.
OnEraseBkgnd	Called when the window background needs erasing.
OnGetMinMaxInfo	Called whenever Windows needs to know the maximized position or dimensions, or the minimum or maximum tracking size.
OnIconEraseBkgnd	Called when **CWnd** is minimized (iconic) and the background of the icon must be filled before painting the icon.
OnKillFocus	Called immediately before **CWnd** loses the input focus.
OnMenuChar	Called when the user presses a menu mnemonic character that doesn't match any of the predefined mnemonics in the current menu.
OnMenuSelect	Called when the user selects a menu item.
OnMove	Called after the position of the **CWnd** has been changed.
OnMoving	Indicates that a user is moving a **CWnd** object.
OnDeviceChange	Notifies an application or device driver of a change to the hardware configuration of a device or the computer.
OnStyleChanged	Indicates that the ::SetWindowLong Windows function has changed one or more of the window's styles.
OnStyleChanging	Indicates that the ::SetWindowLong Windows function is about to change one or more of the window's styles.
OnPaint	Called to repaint a portion of the window.
OnParentNotify	Called when a child window is created or destroyed, or when the user clicks a mouse button while the cursor is over the child window.
OnQueryDragIcon	Called when a minimized (iconic) **CWnd** is about to be dragged by the user.
OnQueryEndSession	Called when the user chooses to end the Windows session.
OnQueryNewPalette	Informs **CWnd** that it is about to receive the input focus.
OnQueryOpen	Called when **CWnd** is an icon and the user requests that the icon be opened.
OnSetFocus	Called after **CWnd** gains the input focus.
OnShowWindow	Called when **CWnd** is to be hidden or shown.
OnSize	Called after the size of **CWnd** has changed.
OnSizing	Indicates that the user is resizing the rectangle.
OnStyleChanged	Indicates that one or more of the window's styles has changed.
OnStyleChanging	Indicates that one or more of the window's styles is about to change.
OnCharToItem	Called by a child list box with the LBS_WANTKEYBOARDINPUT style in response to a WM_CHAR message.
OnCompareItem	Called to determine the relative position of a new item in a child sorted owner-draw combo box or list box.
OnDeleteItem	Called when an owner-draw child list box or combo box is destroyed or when items are removed from the control.
OnDrawItem	Called when a visual aspect of an owner-draw child button control, combo-box control, list-box control, or menu needs to be drawn.
OnDSCNotify	Called in response to an event that a data-source control fires when a control to which the data-source control is bound modifies or attempts to modify the underlying cursor.
OnGetDlgCode	Called for a control so the control can process arrow-key and TAB-key input itself.
OnMeasureItem	Called for an owner-draw child combo box, list box, or menu item when the control is created. **CWnd** informs Windows of the dimensions of the control.
SendChildNotifyLastMsg	Provides a notification message to a child window, from the parent window, so the child window can handle a task .
ReflectChildNotify	Helper function which reflects a message to its source.
OnWndMsg	Indicates if a windows message was handled.
ReflectLastMsg	Reflects the last message to the child window.
OnVKeyToItem	Called by a list box owned by **CWnd** in response to a WM_KEYDOWN message.
OnChar	Called when a keystroke translates to a nonsystem character.
OnDeadChar	Called when a keystroke translates to a nonsystem dead character (such as accent characters).
OnHScroll	Called when the user clicks the horizontal scroll bar of **CWnd**.
OnKeyDown	Called when a nonsystem key is pressed.
OnKeyUp	Called when a nonsystem key is released.
OnLButtonDblClk	Called when the user double-clicks the left mouse button.
OnLButtonDown	Called when the user presses the left mouse button.
OnLButtonUp	Called when the user releases the left mouse button.
OnMButtonDblClk	Called when the user double-clicks the middle mouse button.
OnMButtonDown	Called when the user presses the middle mouse button.
OnMButtonUp	Called when the user releases the middle mouse button.
OnMouseActivate	Called when the cursor is in an inactive window and the user presses a mouse button.
OnMouseMove	Called when the mouse cursor moves.
OnMouseWheel	Called when a user rotates the mouse wheel. Uses Windows NT 4.0 message handling.
OnRegisteredMouseWheel	Called when a user rotates the mouse wheel. Uses Windows 95 and Windows NT 3.51 message-handling.
OnRButtonDblClk	Called when the user double-clicks the right mouse button.
OnRButtonDown	Called when the user presses the right mouse button.
OnRButtonUp	Called when the user releases the right mouse button.
OnSetCursor	Called if mouse input is not captured and the mouse causes cursor movement within a window.
OnTimer	Called after each interval specified in SetTimer.
OnVScroll	Called when the user clicks the window's vertical scroll bar.
OnCaptureChanged	Sends a message to the window that is losing the mouse capture.
OnNcActivate	Called when the nonclient area needs to be changed to indicate an active or inactive state.
OnNcCalcSize	Called when the size and position of the client area need to be calculated.
OnNcCreate	Called prior to OnCreate when the nonclient area is being created.
OnNcDestroy	Called when the nonclient area is being destroyed.
OnNcHitTest	Called by Windows every time the mouse is moved if **CWnd** contains the cursor or has captured mouse input with **SetCapture**.
OnNcLButtonDblClk	Called when the user double-clicks the left mouse button while the cursor is within a nonclient area of **CWnd**
OnOnNcLButtonDown	Called when the user presses the left mouse button while the cursor is within a nonclient area of **CWnd**.
OnNcLButtonUp	Called when the user releases the left mouse button while the cursor is within a nonclient area of **CWnd**.
OnNcMButtonDblClk	Called when the user double-clicks the middle mouse button while the cursor is within a nonclient area of **CWnd**
OnNcMButtonDown	Called when the user presses the middle mouse button while the cursor is within a nonclient area of **CWnd**.
OnNcMButtonUp	Called when the user releases the middle mouse button while the cursor is within a nonclient area of **CWnd**
OnNcMouseMove	Called when the cursor is moved within a nonclient area of **CWnd**
OnNcPaint	Called when the nonclient area needs painting.
OnNcRButtonDblClk	Called when the user double-clicks the right mouse button while the cursor is within a nonclient area of **CWnd**
OnNcRButtonDown	Called when the user presses the right mouse button while the cursor is within a nonclient area of **CWnd**.
OnNcRButtonUp	Called when the user releases the right mouse button while the cursor is within a nonclient area of **CWnd**.
OnMDIActivate	Called when an MDI child window is activated or deactivated.
OnAskCbFormatName	Called by a Clipboard viewer application when a Clipboard owner will display the Clipboard contents.
OnChangeCbChain	Notifies that a specified window is being

	removed from the chain.
OnDestroyClipboard	Called when the Clipboard is emptied through a call to the Windows EmptyClipboard function.
OnDrawClipboard	Called when the contents of the change.
OnHScrollClipboard	Called when a Clipboard owner should scroll the Clipboard image, invalidate the appropriate section, and update the scroll-bar values.
OnPaintClipboard	Called when the client area of the Clipboard viewer needs repainting.
OnRenderAllFormats	Called when the owner application is being destroyed and needs to render all its formats.
OnRenderFormat	Called for the Clipboard owner when a particular format with delayed rendering needs to be rendered.
OnSizeClipboard	Called when the size of the client area of the Clipboard-viewer window has changed.
OnVScrollClipboard	Called when the owner should scroll the Clipboard image, invalidate the appropriate section, and update the scroll-bar values.
OnEnterMenuLoop	Called when a menu modal loop has been entered.
OnExitMenuLoop	Called when a menu modal loop has been exited.

CDialog

CDialog	Constructs a **CDialog** object.
Create	Initializes the **CDialog** object. Creates a modeless dialog box and attaches it to the **CDialog** object.
CreateIndirect	Creates a modeless dialog box from a dialog-box template in memory (not resource-based).
InitModalIndirect	Creates a modal dialog box from a dialog-box template in memory (not resource-based). The parameters are stored until the function **DoModal** is called.
DoModal	Calls a modal dialog box and returns when done.
MapDialogRect	Converts the dialog-box units of a rectangle to screen units.
NextDlgCtrl	Moves the focus to the next dialog-box control in the dialog box.
PrevDlgCtrl	Moves the focus to the previous dialog-box control in the dialog box.
GotoDlgCtrl	Moves the focus to a specified dialog-box control in the dialog box.
SetDefID	Changes the default pushbutton control for a dialog box to a specified pushbutton.
GetDefID	Gets the ID of the default pushbutton control for a dialog box.
SetHelpID	Sets a context-sensitive help ID for the dialog box.
EndDialog	Closes a modal dialog box.
OnInitDialog	Override to augment dialog-box initialization.
OnSetFont	Override to specify the font that a dialog-box control is to use when it draws text.
OnOK	Override to perform the OK button action in a modal dialog box. The default closes the dialog box and **DoModal** returns **IDOK**.
OnCancel	Override to perform the Cancel button or ESC key action. The default closes the dialog box

CFileDialog

m_ofn	The Windows **OPENFILENAME** structure. Provides access to basic file dialog box parameters.
CFileDialog	Constructs a **CFileDialog** object.
DoModal	Displays the dialog box and allows the user to make a selection.
GetPathName	Returns the full path of the selected file.
GetFileName	Returns the filename of the selected file.
GetFileExt	Returns the file extension of the selected file.
GetFileTitle	Returns the title of the selected file.
GetNextPathName	Returns the full path of the next selected file.
GetReadOnlyPref	Returns the read-only status of the selected file.
GetStartPosition	Returns the position of the first element of the filename list.
OnShareViolation	Called when a share violation occurs.
OnFileNameOK	Called to validate the filename entered in the dialog box.
OnLBSelChangedNotify	Called when the list box selection changes.
OnInitDone	Called to handle the **WM_NOTIFY CDN_INITDONE** message.
OnFileNameChange	Called to handle the **WM_NOTIFY CDN_SELCHANGE** message.
OnFolderChange	Called to handle the **WM_NOTIFY CDN_FOLDERCHANGE** message.
OnTypeChange	Called to handle the **WM_NOTIFY CDN_TYPECHANGE** message.

CFontDialog

m_cf	A structure used to customize a **CFontDialog** object.
CFontDialog	Constructs a **CFontDialog** object.
DoModal	Displays the dialog and allows the user to make a selection.
GetCurrentFont	Retrieves the name of the currently selected font.
GetFaceName	Returns the face name of the selected font.
GetStyleName	Returns the style name of the selected font.
GetSize	Returns the point size of the selected font.
GetColor	Returns the color of the selected font.
GetWeight	Returns the weight of the selected font.
IsStrikeOut	Determines whether the font is displayed with strikeout.
IsUnderline	Determines whether the font is underlined.
IsBold	Determines whether the font is bold.
IsItalic	Determines whether the font is italic.

CColorDialog

m_cc	A structure used to customize the settings of the dialog box.
CColorDialog	Constructs a **CColorDialog** object.
DoModal	Displays a color dialog box and allows the user to make a selection.
GetColor	Returns a **COLORREF** structure containing the values of the selected color.
GetSavedCustomColors	Retrieves custom colors created by the user.
SetCurrentColor	Forces the current color selection to the specified color.
OnColorOK	Override to validate the color entered into the dialog box.

CPrintDialog

m_pd	A structure used to customize a **CPrintDialog** object.
CPrintDialog	Constructs a **CPrintDialog** object.
CreatePrinterDC	Creates a printer device context without displaying the Print dialog box.
DoModal	Displays the dialog box and allows the user to make a selection.
GetCopies	Retrieves the number of copies requested.
GetDefaults	Retrieves device defaults without displaying a dialog box.
GetDeviceName	Retrieves the name of the currently selected printer device.
GetDevMode	Retrieves the **DEVMODE** structure.
GetDriverName	Retrieves the name of the currently selected printer driver.
GetFromPage	Retrieves the starting page of the print range.
GetToPage	Retrieves the ending page of the print range.
GetPortName	Retrieves the name of the currently selected printer port.
GetPrinterDC	Retrieves a handle to the printer device context.
PrintAll	Determines whether to print all pages of the document.
PrintCollate	Determines whether collated copies are requested.
PrintRange	Determines whether to print only a specified range of pages.
PrintSelection	Determines whether to print only the currently selected items.

CFindReplaceDialog

m_fr	A structure used to customize a **CFindReplaceDialog** object.
CFindReplaceDialog	Call this function to construct a **CFindReplaceDialog** object.
Create	Creates and displays a **CFindReplaceDialog** dialog box.
FindNext	Call this function to determine whether the user wants to find the next occurrence of the find string.
GetNotifier	Call this function to retrieve the **FINDREPLACE** structure in your registered message handler.
GetFindString	Call this function to retrieve the current find string.
GetReplaceString	Call this function to retrieve the current replace string.
IsTerminating	Call this function to determine whether the dialog box is terminating.
MatchCase	Call this function to determine whether the user wants to match the case of the find string exactly.
MatchWholeWord	Call this function to determine whether the user wants to match entire words only.
ReplaceAll	Call this function to determine whether the

	user wants all occurrences of the string to be replaced.
ReplaceCurrent	Call this function to determine whether the user wants the current word to be replaced.
SearchDown	Call this function to determine whether the user wants the search to proceed in a downward direction.

CStatic

CStatic	Constructs a **CStatic** object.
Create	Creates the Windows static control and attaches it to the **CStatic** object.
SetBitmap	Specifies a bitmap to be displayed in the static control.
GetBitmap	Retrieves the handle of the bitmap previously set with SetBitmap.
SetIcon	Specifies an icon to be displayed in the static control.
GetIcon	Retrieves the handle of the icon previously set with SetIcon.
SetCursor	Specifies a cursor image to be displayed in the static control.
GetCursor	Retrieves the handle of the cursor image previously set with SetCursor.
SetEnhMetaFile	Specifies an enhanced metafile to be displayed in the static control.
GetEnhMetaFile	Retrieves the handle of the enhanced metafile previously set with SetEnhMetaFile.

CButton

CButton	Constructs a **CButton** object.
Create	Creates the Windows button control and attaches it to the **CButton** object.
GetState	Retrieves the check state, highlight state, and focus state of a button control.
SetState	Sets the highlighting state of a button control.
GetCheck	Retrieves the check state of a button control.
SetCheck	Sets the check state of a button control.
GetButtonStyle	Retrieves information about the button control style.
SetButtonStyle	Changes the style of a button.
GetIcon	Retrieves the handle of the icon previously set with SetIcon.
SetIcon	Specifies an icon to be displayed on the button.
GetBitmap	Retrieves the handle of the bitmap previously set with SetBitmap.
SetBitmap	Specifies a bitmap to be displayed on the button.
GetCursor	Retrieves the handle of the cursor image previously set with SetCursor.
SetCursor	Specifies a cursor image to be displayed on the button.
DrawItem	Override to draw an owner-drawn **CButton** object.

CEdit

CEdit	Constructs a **CEdit** control object.
Create	Creates the Windows edit control and attaches it to the **CEdit** object.
CanUndo	Determines whether an edit-control operation
GetLineCount	Retrieves the number of lines in a multiple-line edit control.
GetModify	Determines whether the contents of an edit control have been modified.
SetModify	Sets or clears the modification flag for an edit control.
GetRect	Gets the formatting rectangle of an edit control.
GetSel	Gets the starting and ending character positions of the current selection in an edit control.
GetHandle	Retrieves a handle to the memory currently allocated for a multiple-line edit control.
SetHandle	Sets the handle to the local memory that will be used by a multiple-line edit control.
SetMargins	Sets the left and right margins for this **CEdit**.
GetMargins	Gets the left and right margins for this **CEdit**.
SetLimitText	Sets the maximum amount of text this **CEdit** can contain.
GetLimitText	Gets the maximum amount of text this **CEdit** can contain.
PosFromChar	Retrieves the coordinates of the upper-left corner of a specified character index.
CharFromPos	Retrieves the line and character indices for the character closest to a specified position.
GetLine	Retrieves a line of text from an edit control.
GetPasswordChar	Retrieves the password character displayed in an edit control when the user enters text.
GetFirstVisibleLine	Determines the topmost visible line in an edit control.
EmptyUndoBuffer	Resets (clears) the undo flag of an edit control.
FmtLines	Sets the inclusion of soft line-break characters on or off within a multiple-line edit control.
LimitText	Limits the length of the text that the user may enter into an edit control.
LineFromChar	Retrieves the line number of the line that contains the specified character index.
LineIndex	Retrieves the character index of a line within a multiple-line edit control.
LineLength	Retrieves the length of a line in an edit control.
LineScroll	Scrolls the text of a multiple-line edit control.
ReplaceSel	Replaces the current selection in an edit control with the specified text.
SetPasswordChar	Sets or removes a password character displayed in an edit control when the user enters text.
SetRect	Sets the formatting rectangle of a multiple-line edit control and updates the control.
SetRectNP	Sets the formatting rectangle of a multiple-line edit control without redrawing the control window.
SetSel	Selects a range of characters in an edit control.
SetTabStops	Sets the tab stops in a multiple-line edit control.
SetReadOnly	Sets the read-only state of an edit control.
Undo	Reverses the last edit-control operation.
Clear	Deletes (clears) the current selection (if any) in the edit control.
Copy	Copies the current selection (if any) in the edit control to the Clipboard in **CF_TEXT** format.
Cut	Deletes (cuts) the current selection (if any) in the edit control and copies the deleted text to the Clipboard in **CF_TEXT** format.
Paste	Inserts the data from the Clipboard into the edit control at the current cursor position. Data is inserted only if the Clipboard contains data in **CF_TEXT** format.

CListBox

CListBox	Constructs a **CListBox** object.
Create	Creates the Windows list box and attaches it to the **CListBox** object.
InitStorage	Preallocates blocks of memory for list box items and strings.
GetCount	Returns the number of strings in a list box.
GetHorizontalExtent	Returns the width in pixels that a list box can be scrolled horizontally.
SetHorizontalExtent	Sets the width in pixels that a list box can be scrolled horizontally.
GetTopIndex	Returns the index of the first visible string in a list box.
SetTopIndex	Sets the zero-based index of the first visible string in a list box.
GetItemData	Returns the 32-bit value associated with the list-box item.
GetItemDataPtr	Returns a pointer to a list-box item.
SetItemData	Sets the 32-bit value associated with the list-box item.
SetItemDataPtr	Sets a pointer to the list-box item.
GetItemRect	Returns the bounding rectangle of the list-box item as it is currently displayed.
ItemFromPoint	Returns the index of the list-box item nearest a point.
SetItemHeight	Sets the height of items in a list box.
GetItemHeight	Determines the height of items in a list box.
GetSel	Returns the selection state of a list-box item.
GetText	Copies a list-box item into a buffer.
GetTextLen	Returns the length in bytes of a list-box item.
SetColumnWidth	Sets the column width of a multicolumn list box.
SetTabStops	Sets the tab-stop positions in a list box.
GetLocale	Retrieves the locale identifier for a list box.
SetLocale	Sets the locale identifier for a list box.
GetCurSel	Returns the zero-based index of the currently selected string in a list box.
SetCurSel	Selects a list-box string.
SetSel	Selects or deselects a list-box item in a multiple-selection list box.
GetCaretIndex	Determines the index of the item that has the focus rectangle in a multiple-selection list box.
SetCaretIndex	Sets the focus rectangle to the item at the specified index in a multiple-selection list box.
GetSelCount	Returns the number of strings currently selected in a multiple-selection list box.
GetSelItems	Returns the indices of the strings currently selected in a list box.
SelItemRange	Selects or deselects a range of strings in a multiple-selection list box.
SetAnchorIndex	Sets the anchor in a multiple-selection list box to begin an extended selection.
GetAnchorIndex	Retrieves the zero-based index of the current anchor item in a list box.
AddString	Adds a string to a list box.
DeleteString	Deletes a string from a list box.

InsertString	Inserts a string at a specific location in a list box.
ResetContent	Clears all the entries from a list box.
Dir	Adds filenames from the current directory to a list box.
FindString	Searches for a string in a list box.
FindStringExact	Finds the first list-box string that matches a specified string.
SelectString	Searches for and selects a string in a single-selection list box.
DrawItem	Called by the framework when a visual aspect of an owner-draw list box changes.
MeasureItem	Called by the framework when an owner-draw list box is created to determine list-box dimensions.
CompareItem	Called by the framework to determine the position of a new item in a sorted owner-draw list box.
DeleteItem	Called by the framework when the user deletes an item from an owner-draw list box.
VKeyToItem	Override to provide custom **WM_KEYDOWN** handling for list boxes with the **LBS_WANTKEYBOARDINPUT** style set.
CharToItem	Override to provide custom **WM_CHAR** handling for owner-draw list boxes which don't have strings.

CComboBox

CComboBox	Constructs a **CComboBox** object.
Create	Creates the combo box and attaches it to the **CComboBox** object.
InitStorage	Preallocates blocks of memory for items and strings in the list-box portion of the combo box.
GetCount	Retrieves the number of items in the list box of a combo box.
GetCurSel	Retrieves the index of the currently selected item, if any, in the list box of a combo box.
SetCurSel	Selects a string in the list box of a combo box.
GetEditSel	Gets the starting and ending character positions of the current selection in the edit control of a combo box.
SetEditSel	Selects characters in the edit control of a combo box.
SetItemData	Sets the 32-bit value associated with the specified item in a combo box.
SetItemDataPtr	Sets the 32-bit value associated with the specified item in a combo box to the specified pointer (**void***).
GetItemData	Retrieves the application-supplied 32-bit value associated with the specified combo-box item.
GetItemDataPtr	Retrieves the application-supplied 32-bit value associated with the specified combo-box item as a pointer (**void***).
GetTopIndex	Returns the index of the first visible item in the list-box portion of the combo box.
SetHorizontalExtent	Sets the width in pixels that the list-box portion of the combo box can be scrolled horizontally.
GetHorizontalExtent	Returns the width in pixels that the list-box portion of the combo box can be scrolled horizontally.
SetDroppedWidth	Sets the minimum allowable width for the drop-down list-box portion of a combo box.
GetDroppedWidth	Retrieves the minimum allowable width for the drop-down list-box portion of a combo box.
Clear	Deletes (clears) the current selection (if any) in the edit control.
Copy	Copies the current selection (if any) onto the Clipboard in **CF_TEXT** format.
Cut	Deletes (cuts) the current selection, if any, in the edit control and copies the deleted text onto the Clipboard in **CF_TEXT** format.
Paste	Inserts the data from the Clipboard into the edit control at the current cursor position. Data is inserted only if the Clipboard contains data in **CF_TEXT** format.
LimitText	Limits the length of the text that the user can enter into the edit control of a combo box.
SetItemHeight	Sets the height of list items in a combo box or the height of the edit-control (or static-text) portion of a combo box.
GetItemHeight	Retrieves the height of list items in a combo box.
GetLBText	Gets a string from the list box of a combo box.
GetLBTextLen	Gets the length of a string in the list box of a combo box.
ShowDropDown	Shows or hides the list box of a combo box that has the **CBS_DROPDOWN** or **CBS_DROPDOWNLIST** style.
GetDroppedControlRect	Retrieves the screen coordinates of the visible (dropped-down) list box of a drop-down combo box.
GetDroppedState	Determines whether the list box of a drop-down combo box is visible (dropped down).
SetExtendedUI	Selects either the default user interface or the extended user interface for a combo box that has the **CBS_DROPDOWN** or **CBS_DROPDOWNLIST** style.
GetExtendedUI	Determines whether a combo box has the default user interface or the extended user interface.
GetLocale	Retrieves the locale identifier for a combo box.
SetLocale	Sets the locale identifier for a combo box.
AddString	Adds a string to the end of the list in the list box of a combo box or at the sorted position for list boxes with the **CBS_SORT** style.
DeleteString	Deletes a string from the list box of a combo box.
InsertString	Inserts a string into the list box of a combo box.
ResetContent	Removes all items from the list box and edit control of a combo box.
Dir	Adds a list of filenames to the list box of a combo box.
FindString	Finds the first string that contains the specified prefix in the list box of a combo box.
FindStringExact	Finds the first list-box string (in a combo box) that matches the specified string.
SelectString	Searches for a string in the list box of a combo box and, if the string is found, selects the string in the list box and copies the string to the edit control.
DrawItem	Called by the framework when a visual aspect of an owner-draw combo box changes.
MeasureItem	Called by the framework to determine combo box dimensions when an owner-draw combo box is created.
CompareItem	Called by the framework to determine the relative position of a new list item in a sorted owner-draw combo box.
DeleteItem	Called by the framework when a list item is deleted from an owner-draw combo box.

CToolBar

CToolBar	Constructs a **CToolBar** object.
Create	Creates the Windows toolbar and attaches it to the **CToolBar** object.
CreateEx	Creates a **CToolBar** object with additional styles for the embedded **CToolBarCtrl** object.
SetSizes	Sets the sizes of buttons and their bitmaps.
SetHeight	Sets the height of the toolbar.
LoadToolBar	Loads a toolbar resource created with the resource editor.
LoadBitmap	Loads the bitmap containing bitmap-button images.
SetBitmap	Sets a bitmapped image.
SetButtons	Sets button styles and an index of button images within the bitmap.
CommandToIndex	Returns the index of a button with the given command ID.
GetItemID	Returns the command ID of a button or separator at the given index.
GetItemRect	Retrieves the display rectangle for the item at the given index.
GetButtonStyle	Retrieves the style for a button.
SetButtonStyle	Sets the style for a button.
GetButtonInfo	Retrieves the ID, style, and image number of a button.
SetButtonInfo	Sets the ID, style, and image number of a button.
GetButtonText	Retrieves the text that will appear on a button.
SetButtonText	Sets the text that will appear on a button.
GetToolBarCtrl	Allows direct access to the underlying common control.

CStatusBar

CStatusBar	Constructs a **CStatusBar** object.
Create	Creates the status bar, attaches it to the **CStatusBar** object, and sets the initial font and bar height.
CreateEx	Creates a **CStatusBar** object with additional styles for the embedded **CStatusBarCtrl** object.
SetIndicators	Sets indicator IDs.
CommandToIndex	Gets index for a given indicator ID.
GetItemID	Gets indicator ID for a given index.
GetItemRect	Gets display rectangle for a given index.
GetPaneInfo	Gets indicator ID, style, and width for a given index.
GetPaneStyle	Gets indicator style for a given index.
GetPaneText	Gets indicator text for a given index.
GetStatusBarCtrl	Allows direct access to the underlying common control.

SetPaneStyle	Sets indicator style for a given index.
SetPaneText	Sets indicator text for a given index.
SetPaneInfo	Sets indicator ID, style, and width for a given index.
DrawItem	Called when a visual aspect of an owner-draw status bar control changes.

CScrollBar

CScrollBar	Constructs a **CScrollBar** object.
Create	Creates the Windows scroll bar and attaches it to the **CScrollBar** object.
GetScrollPos	Retrieves the current position of a scroll box.
SetScrollPos	Sets the current position of a scroll box.
GetScrollRange	Retrieves the current minimum and maximum scroll-bar positions for the given scroll bar.
SetScrollRange	Sets minimum and maximum position values for the given scroll bar.
ShowScrollBar	Shows or hides a scroll bar.
EnableScrollBar	Enables or disables one or both arrows of a scroll bar.
SetScrollInfo	Sets information about the scroll bar.
GetScrollInfo	Retrieves information about the scroll bar.
GetScrollLimit	Retrieves the limit of the scroll bar

CImageList

m_hImageList	A handle containing the image list attached to this object.
CImageList	Constructs a **CImageList** object.
Create	Initializes an image list and attaches it to a **CImageList** object.
GetSafeHandle	Retrieves **m_hImageList**.
operator HIMAGELIST	Returns the **HIMAGELIST** attached to the **CImageList**.
FromHandle	Returns a pointer to a **CImageList** object when given a handle to a device context. If a **CImageList** object is not attached to the handle, a temporary **CImageList** object is created and attached.
FromHandlePermanent	Returns a pointer to a **CImageList** object when given a handle to an image list. If a **CImageList** object is not attached to the handle, **NULL** is returned.
DeleteTempMap	Called by the CWinApp idle-time handler to delete any temporary **CImageList** object created by **FromHandle**.
GetImageCount	Retrieves the number of images in an image list.
SetBkColor	Sets the background color for an image list.
GetBkColor	Retrieves the current background color for an image list.
GetImageInfo	Retrieves information about an image.
Attach	Attaches an image list to a **CImageList** object.
Detach	Detaches an image list object from a **CImageList** object and returns a handle to an image list.
DeleteImageList	Deletes an image list.
SetImageCount	Resets the count of images in an image list.
Add	Adds an image or images to an image list.
Remove	Removes an image from an image list.
Replace	Replaces an image in an image list with a new image.
ExtractIcon	Creates an icon based on an image and mask in an image list.
Draw	Draws the image that is being dragged during a drag-and-drop operation.
SetOverlayImage	Adds the zero-based index of an image to the list of images to be used as overlay masks.
Copy	Copies an image within a **CImageList** object.
DrawIndirect	Draws an image from an image list.
SetDragCursorImage	Creates a new drag image.
GetDragImage	Gets the temporary image list that is used for dragging.
Read	Reads an image list from an archive.
Write	Writes an image list to an archive.
BeginDrag	Begins dragging an image.
DragEnter	Locks updates during a drag operation and displays the drag image at a specified position.
EndDrag	Ends a drag operation.
DragLeave	Unlocks the window and hides the drag image so that the window can be updated.
DragMove	Moves the image that is being dragged during a drag-and-drop operation.
DragShowNolock	Shows or hides the drag image during a drag operation, without locking the window.

CListCtrl

CListCtrl	Constructs a **CListCtrl** object.
Create	Creates a list control and attaches it to a **CListCtrl** object.
GetBkColor	Retrieves the background color of a list view control.
SetBkColor	Sets the background color of the list view control.
GetImageList	Retrieves the handle of an image list used for drawing list view items.
SetImageList	Assigns an image list to a list view control.
GetItemCount	Retrieves the number of items in a list view control.
GetItem	Retrieves a list view item's attributes.
SetItem	Sets some or all of a list view item's attributes.
GetCallbackMask	Retrieves the callback mask for a list view control.
SetCallbackMask	Sets the callback mask for a list view control.
GetNextItem	Searches for a list view item with specified properties and with specified relationship to a given item.
GetFirstSelectedItemPosition	Retrieves the position of the first selected list view item in a list view control.
GetNextSelectedItem	Retrieves the next selected list view item for iterating.
GetItemRect	Retrieves the bounding rectangle for an item.
SetItemPosition	Moves an item to a specified position in a list view control.
GetItemPosition	Retrieves the position of a list view item.
GetStringWidth	Determines the minimum column width necessary to display all of a given string.
GetEditControl	Retrieves the handle of the edit control used to edit an item's text.
GetColumn	Retrieves the attributes of a control's column.
SetColumn	Sets the attributes of a list view column.
GetColumnWidth	Retrieves the width of a column in report view or list view.
SetColumnWidth	Changes the width of a column in report view or list view.
GetCheck	Retrieves the current display status of the state image associated with an item.
SetCheck	Sets the the current display status of the state image associated with an item.
GetViewRect	Retrieves the bounding rectangle of all items in the list view control.
GetTextColor	Retrieves the text color of a list view control.
SetTextColor	Sets the text color of a list view control.
GetTextBkColor	Retrieves the text background color of a list view control.
SetTextBkColor	Sets the background color of text in a list view control.
GetTopIndex	Retrieves the index of the topmost visible item.
GetCountPerPage	Calculates the number of items that can fit vertically in a list view control.
GetOrigin	Retrieves the current view origin for a list view control.
SetItemState	Changes the state of an item in a list view control.
GetItemState	Retrieves the state of a list view item.
GetItemText	Retrieves the text of a list view item or subitem.
SetItemText	Changes the text of a list view item or subitem.
SetItemCount	Prepares a list view control for adding a large number of items.
SetItemData	Sets the item's application-specific value.
GetItemData	Retrieves the application-specific value associated with an item.
GetSelectedCount	Retrieves the number of selected items in the list view control.
SetColumnOrderArray	Sets the column order (left to right) of a list view control.
GetColumnOrderArray	Retrieves the column order (left to right) of a list view control.
SetIconSpacing	Sets the spacing between icons in a list view control.
GetHeaderCtrl	Retrieves the header control of a list view control.
GetHotCursor	Retrieves the cursor used when hot tracking is enabled for a list view control.
SetHotCursor	Sets the cursor used when hot tracking is enabled for a list view control.
GetSubItemRect	Retrieves the bounding rectangle of an item in a list view control.
GetHotItem	Retrieves the list view item currently under the cursor.
SetHotItem	Sets the current hot item of a list view control.
GetSelectionMark	Retrieves the selection mark of a list view control.
SetSelectionMark	Sets the selection mark of a list view control.
GetExtendedStyle	Retrieves the current extended styles of a list view control.
SetExtendedStyle	Sets the current extended styles of a list view control.
SubItemHitTest	Determines which list view item, if any, is at a given position.
GetWorkAreas	Retrieves the current working areas of a list view control.

GetNumberOfWorkAreas	Retrieves the current number of working areas for a list view control.
SetItemCountEx	Sets the item count for a virtual list view control.
SetWorkAreas	Sets the area where icons can be displayed in a list view control.
ApproximateViewRect	Determines the width and height required to display the items of a list view control.
GetBkImage	Retreives the current background image of a list view control.
SetBkImage	Sets the current background image of a list view control.
GetHoverTime	Retrieves the current hover time of a list view control.
SetHoverTime	Sets the current hover time of a list view control.
InsertItem	Inserts a new item in a list view control.
DeleteItem	Deletes an item from the control.
DeleteAllItems	Deletes all items from the control.
FindItem	Searches for a list view item having specified characteristics.
SortItems	Sorts list view items using an application-defined comparison function.
HitTest	Determines which list view item is at a specified position.
EnsureVisible	Ensures that an item is visible.
Scroll	Scrolls the content of a list view control.
RedrawItems	Forces a list view control to repaint a range of items.
Update	Forces the control to repaint a specified item.
Arrange	Aligns items on a grid.
EditLabel	Begins in-place editing of an item's text.
InsertColumn	Inserts a new column in a list view control.
DeleteColumn	Deletes a column from the list view control.
CreateDragImage	Creates a drag image list for a specified item.
DrawItem	Called when a visual aspect of an owner-draw control changes.

CSliderCtrl

CSliderCtrl	Constructs a **CSliderCtrl** object.
Create	Creates a slider control and attaches it to a **CSliderCtrl** object.
GetLineSize	Retrieves the line size of a slider control.
SetLineSize	Sets the line size of a slider control.
GetPageSize	Retrieves the page size of a slider control.
SetPageSize	Sets the page size of a slider control.
GetRangeMax	Retrieves the maximum position for a slider.
GetRangeMin	Retrieves the minimum position for a slider.
GetRange	Retrieves the minimum and maximum positions for a slider.
SetRangeMin	Sets the minimum position for a slider.
SetRangeMax	Sets the maximum position for a slider.
SetRange	Sets the minimum and maximum positions for a slider.
GetSelection	Retrieves the range of the current selection.
SetSelection	Sets the range of the current selection.
GetChannelRect	Retrieves the size of the slider control's channel.
GetThumbRect	Retrieves the size of the slider control's thumb.
GetPos	Retrieves the current position of the slider.
SetPos	Sets the current position of the slider.
GetNumTics	Retrieves the number of tick marks in a slider control.
GetTicArray	Retrieves the array of tick mark positions for a slider control.
GetTic	Retrieves the position of the specified tick mark.
GetTicPos	Retrieves the position of the specified tick mark, in client coordinates.
SetTic	Sets the position of the specified tick mark.
SetTicFreq	Sets the frequency of tick marks per slider control increment.
GetBuddy	Retrieves the handle to a slider control buddy window at a given location.
SetBuddy	Assigns a window as the buddy window for a slider control.
GetToolTips	Retrieves the handle to the tooltip control assigned to the slider control, if any.
SetToolTips	Assigns a tooltip control to a slider control.
SetTipSide	Positions a tooltip control used by a trackbar control.
ClearSel	Clears the current selection in a slider control.
VerifyPos	Verifies that the position of a slider control is between the minimum and maximum values.
ClearTics	Removes the current tick marks from a slider control.

CDC

m_hDC	The output-device context used by this **CDC** object.
m_hAttribDC	The attribute-device context used by this **CDC** object.
CDC	Constructs a **CDC** object.
CreateDC	Creates a device context for a specific device.
CreateIC	Creates an information context for a specific device. This provides a fast way to get information about the device without creating a device context.
CreateCompatibleDC	Creates a memory-device context that is compatible with another device context. You can use it to prepare images in memory.
DeleteDC	Deletes the Windows device context associated with this **CDC** object.
FromHandle	Returns a pointer to a **CDC** object when given a handle to a device context. If a **CDC** object is not attached to the handle, a temporary **CDC** object is created and attached.
DeleteTempMap	Called by the **CWinApp** idle-time handler to delete any temporary **CDC** object created by **FromHandle**. Also detaches the device context.
Attach	Attaches a Windows device context to this **CDC** object.
Detach	Detaches the Windows device context from this **CDC** object.
SetAttribDC	Sets **m_hAttribDC**, the attribute device context.
SetOutputDC	Sets **m_hDC**, the output device context.
ReleaseAttribDC	Releases **m_hAttribDC**, the attribute device context.
ReleaseOutputDC	Releases **m_hDC**, the output device context.
GetCurrentBitmap	Returns a pointer to the currently selected **CBitmap** object.
GetCurrentBrush	Returns a pointer to the currently selected **CBrush** object.
GetCurrentFont	Returns a pointer to the currently selected **CFont** object.
GetCurrentPalette	Returns a pointer to the currently selected **CPalette** object.
GetCurrentPen	Returns a pointer to the currently selected **CPen** object.
GetWindow	Returns the window associated with the display device context.
GetSafeHdc	Returns m_hDC, the output device context.
SaveDC	Saves the current state of the device context.
RestoreDC	Restores the device context to a previous state saved with **SaveDC**.
ResetDC	Updates the **m_hAttribDC** device context.
GetDeviceCaps	Retrieves a specified kind of device-specific information about a given display device's capabilities.
IsPrinting	Determines whether the device context is being used for printing.
GetBrushOrg	Retrieves the origin of the current brush.
SetBrushOrg	Specifies the origin for the next brush selected into a device context.
EnumObjects	Enumerates the pens and brushes available in a device context.
SelectObject	Selects a GDI drawing object such as a pen.
SelectStockObject	Selects one of the predefined stock pens, brushes, or fonts provided by Windows.
GetNearestColor	Retrieves the closest logical color to a specified logical color that the given device can represent.
SelectPalette	Selects the logical palette.
RealizePalette	Maps palette entries in the current logical palette to the system palette.
UpdateColors	Updates the client area of the device context by matching the current colors in the client area to the system palette on a pixel-by-pixel basis.
GetHalftoneBrush	Retrieves a halftone brush.
GetBkColor	Retrieves the current background color.
SetBkColor	Sets the current background color.
GetBkMode	Retrieves the background mode.
SetBkMode	Sets the background mode.
GetPolyFillMode	Retrieves the current polygon-filling mode.
SetPolyFillMode	Sets the polygon-filling mode.
GetROP2	Retrieves the current drawing mode.
SetROP2	Sets the current drawing mode.
GetStretchBltMode	Retrieves the current bitmap-stretching mode.
SetStretchBltMode	Sets the bitmap-stretching mode.
GetTextColor	Retrieves the current text color.
SetTextColor	Sets the text color.
GetColorAdjustment	Retrieves the color adjustment values for the device context.
SetColorAdjustment	Sets the color adjustment values for the device context using the specified values.
GetMapMode	Retrieves the current mapping mode.
SetMapMode	Sets the current mapping mode.
GetViewportOrg	Retrieves the x- and y-coordinates of the viewport origin.
SetViewportOrg	Sets the viewport origin.
OffsetViewportOrg	Modifies the viewport origin relative to the

	coordinates of the current viewport origin.
GetViewportExt	Retrieves the x- and y-extents of the viewport.
SetViewportExt	Sets the x- and y-extents of the viewport.
ScaleViewportExt	Modifies the viewport extent relative to the current values.
GetWindowOrg	Retrieves the x- and y-coordinates of the origin of the associated window.
SetWindowOrg	Sets the window origin of the device context.
OffsetWindowOrg	Modifies the window origin relative to the coordinates of the current window origin.
GetWindowExt	Retrieves the x- and y-extents of the associated window.
SetWindowExt	Sets the x- and y-extents of the associated window.
ScaleWindowExt	Modifies the window extents relative to the current values.
DPtoHIMETRIC	Converts device units into **HIMETRIC** units.
DPtoLP	Converts device units into logical units.
HIMETRICtoDP	Converts **HIMETRIC** units into device units.
HIMETRICtoLP	Converts **HIMETRIC** units into logical units.
LPtoDP	Converts logical units into device units.
LPtoHIMETRIC	Converts logical units into **HIMETRIC** units.
FillRgn	Fills a specific region with the specified brush.
FrameRgn	Draws a border around a specific region using a brush.
InvertRgn	Inverts the colors in a region.
PaintRgn	Fills a region with the selected brush.
SetBoundsRect	Controls the accumulation of bounding-rectangle information for the specified device context.
GetBoundsRect	Returns the current accumulated bounding rectangle for the specified device context.
GetClipBox	Retrieves the dimensions of the tightest bounding rectangle around the current clipping boundary.
SelectClipRgn	Combines the given region with the current clipping region by using the specified mode.
ExcludeClipRect	Creates a new clipping region that consists of the existing clipping region minus the specified rectangle.
ExcludeUpdateRgn	Prevents drawing within invalid areas of a window by excluding an updated region in the window from a clipping region.
IntersectClipRect	Creates a new clipping region by forming the intersection of the current region and a rectangle.
OffsetClipRgn	Moves the clipping region of the given device.
PtVisible	Specifies whether the given point is within the clipping region.
RectVisible	Determines whether any part of the given rectangle lies within the clipping region.
GetCurrentPosition	Retrieves the current position of the pen (in logical coordinates).
MoveTo	Moves the current position.
LineTo	Draws a line from the current position up to, but not including, a point.
Arc	Draws an elliptical arc.
ArcTo	Draws an elliptical arc. This function is similar to **Arc**, except that the current position is updated.
AngleArc	Draws a line segment and an arc, and moves
GetArcDirection	Returns the current arc direction for the device context.
SetArcDirection	Sets the drawing direction to be used for arc and rectangle functions.
PolyDraw	Draws a set of line segments and Bézier splines. This function updates the current position.
Polyline	Draws a set of line segments connecting the specified points.
PolyPolyline	Draws multiple series of connected line segments. The current position is neither used nor updated by this function.
PolylineTo	Draws one or more straight lines and moves the current position to the ending point of the last line.
PolyBezier	Draws one or more Bézier splines. The current position is neither used nor updated.
PolyBezierTo	Draws one or more Bézier splines, and moves the current position to the ending point of the last Bézier spline.
FillRect	Fills a given rectangle by using a specific brush.
FrameRect	Draws a border around a rectangle.
InvertRect	Inverts the contents of a rectangle.
DrawIcon	Draws an icon.
DrawDragRect	Erases and redraws a rectangle as it is dragged.
FillSolidRect	Fills a rectangle with a solid color.
Draw3dRect	Draws a three-dimensional rectangle.
DrawEdge	Draws the edges of a rectangle.
DrawFrameControl	Draw a frame control.
DrawState	Displays an image and applies a visual effect to indicate a state.
Chord	Draws a chord (a closed figure bounded by the intersection of an ellipse and a line segment).
DrawFocusRect	Draws a rectangle in the style used to indicate focus.
Ellipse	Draws an ellipse.
Pie	Draws a pie-shaped wedge.
Polygon	Draws a polygon consisting of two or more points (vertices) connected by lines.
PolyPolygon	Creates two or more polygons that are filled using the current polygon-filling mode. The polygons may be disjoint or they may overlap.
Polyline	Draws a polygon consisting of a set of line segments connecting specified points.
Rectangle	Draws a rectangle using the current pen and fills it using the current brush.
RoundRect	Draws a rectangle with rounded corners using the current pen and filled using the current brush.
PatBlt	Creates a bit pattern.
BitBlt	Copies a bitmap from a specified device context.
StretchBlt	Moves a bitmap from a source rectangle and device into a destination rectangle, stretching or compressing the bitmap if necessary to fit the dimensions of the destination rectangle.
GetPixel	Retrieves the RGB color value of the pixel at the specified point.
SetPixel	Sets the pixel at the specified point to the closest approximation of the specified color.
SetPixelV	Sets the pixel at the specified coordinates to the closest approximation of the specified color. **SetPixelV** is faster than **SetPixel** because it does not need to return the color value of the point actually painted.
FloodFill	Fills an area with the current brush.
ExtFloodFill	Fills an area with the current brush. Provides more flexibility than the FloodFill member function.
MaskBlt	Combines the color data for the source and destination bitmaps using the given mask and raster operation.
PlgBlt	Performs a bit-block transfer of the bits of color data from the specified rectangle in the source device context to the specified parallelogram in the given device context.
TextOut	Writes a character string at a specified location using the currently selected font.
ExtTextOut	Writes a character string within a rectangular region using the currently selected font.
TabbedTextOut	Writes a character string at a specified location, expanding tabs to the values specified in an array of tab-stop positions.
DrawText	Draws formatted text in the specified rectangle.
GetTextExtent	Computes the width and height of a line of text on the attribute device context using the current font to determine the dimensions.
GetOutputTextExtent	Computes the width and height of a line of text on the output device context using the current font to determine the dimensions.
GetTabbedTextExtent	Computes the width and height of a character string on the attribute device context.
GetOutputTabbedTextExtent	Computes the width and height of a character string on the output device context.
GrayString	Draws dimmed (grayed) text at the given location.
GetTextAlign	Retrieves the text-alignment flags.
SetTextAlign	Sets the text-alignment flags.
GetTextFace	Copies the typeface name of the current font into a buffer as a null-terminated string.
GetTextMetrics	Retrieves the metrics for the current font from the attribute device context.
GetOutputTextMetrics	Retrieves the metrics for the current font from the output device context.
SetTextJustification	Adds space to the break characters in a string.
GetTextCharacterExtra	Retrieves the current setting for the amount of intercharacter spacing.
SetTextCharacterExtra	Sets the amount of intercharacter spacing.
GetFontData	Retrieves font metric information from a scalable font file. The information to retrieve is identified by specifying an offset into the font file and the length of the information to return.
GetKerningPairs	Retrieves the character kerning pairs for the font that is currently selected in the specified device context.
GetOutlineTextMetrics	Retrieves font metric information for TrueType fonts.
GetGlyphOutline	Retrieves the outline curve or bitmap for an

	outline character in the current font.
GetCharABCWidths	Retrieves the widths, in logical units, of consecutive characters in a given range from the current font.
GetCharWidth	Retrieves the fractional widths of consecutive characters in a given range from the current font.
GetOutputCharWidth	Retrieves the widths of individual characters in a consecutive group of characters from the current font using the output device context.
SetMapperFlags	Alters the algorithm that the font mapper uses when it maps logical fonts to physical fonts.
GetAspectRatioFilter	Retrieves the setting for the current aspect-ratio filter.
QueryAbort	Calls the AbortProc callback function for a printing application and queries whether the printing should be terminated.
Escape	Allows applications to access facilities that are not directly available from a particular device through GDI. Also allows access to Windows escape functions. Escape calls made by an application are translated and sent to the device driver.
DrawEscape	Accesses drawing capabilities of a video display that are not directly available through the graphics device interface (GDI).
StartDoc	Informs the device driver that a new print job is starting.
StartPage	Informs the device driver that a new page is starting.
EndPage	Informs the device driver that a page is ending.
SetAbortProc	Sets a programmer-supplied callback function that Windows calls if a print job must be aborted.
AbortDoc	Terminates the current print job, erasing everything the application has written to the device since the last call of the **StartDoc** member function.
EndDoc	Ends a print job started by the **StartDoc** member function.
ScrollDC	Scrolls a rectangle of bits horizontally and vertically.
PlayMetaFile	Plays the contents of the specified metafile on the given device. The enhanced version of **PlayMetaFile** displays the picture stored in the given enhanced-format metafile. The metafile can be played any number of times.
AddMetaFileComment	Copies the comment from a buffer into a specified enhanced-format metafile.
AbortPath	Closes and discards any paths in the device context.
BeginPath	Opens a path bracket in the device context.
CloseFigure	Closes an open figure in a path.
EndPath	Closes a path bracket and selects the path defined by the bracket into the device context.
FillPath	Closes any open figures in the current path and fills the path's interior by using the current brush and polygon-filling mode.
FlattenPath	Transforms any curves in the path selected into the current device context, and turns each curve into a sequence of lines.
GetMiterLimit	Returns the miter limit for the device context.
GetPath	Retrieves the coordinates defining the endpoints of lines and the control points of curves found in the path that is selected into the device context.
SelectClipPath	Selects the current path as a clipping region for the device context, combining the new region with any existing clipping region by using the specified mode.
SetMiterLimit	Sets the limit for the length of miter joins for the device context.
StrokeAndFillPath	Closes any open figures in a path, strikes the outline of the path by using the current pen, and fills its interior by using the current brush.
StrokePath	Renders the specified path by using the current pen.
WidenPath	Redefines the current path as the area that would be painted if the path were stroked using the pen currently selected into the device context.

CBrush

CBrush	Constructs a **CBrush** object.
CreateSolidBrush	Initializes a brush with the specified solid color.
CreateHatchBrush	Initializes a brush with the specified hatched pattern and color.
CreateBrushIndirect	Initializes a brush with the style, color, and pattern specified in a LOGBRUSH structure.
CreatePatternBrush	Initializes a brush with a pattern specified by a bitmap.
CreateDIBPatternBrush	Initializes a brush with a pattern specified by a device-independent bitmap (DIB).
CreateSysColorBrush	Creates a brush that is the default system color.
FromHandle	Returns a pointer to a **CBrush** object when given a handle to a Windows **HBRUSH** object.
GetLogBrush	Gets a LOGBRUSH structure.
operator HBRUSH	Returns the Windows handle attached to the **CBrush** object.

CPen

CPen	Constructs a **CPen** object.
CreatePen	Creates a logical cosmetic or geometric pen with the specified style, width, and brush attributes, and attaches it to the **CPen** object.
CreatePenIndirect	Creates a pen with the style, width, and color given in a LOGPEN structure, and attaches it to the **CPen** object.
FromHandle	Returns a pointer to a **CPen** object when given a Windows **HPEN**.
operator HPEN	Returns the Windows handle attached to the **CPen** object.
GetLogPen	Gets a LOGPEN underlying structure.
GetExtLogPen	Gets an EXTLOGPEN underlying structure.

CFont

CFont	Constructs a **CFont** object.
CreateFontIndirect	Initializes a **CFont** object with the characteristics given in a **LOGFONT** structure.
CreateFont	Initializes a **CFont** with the specified characteristics.
CreatePointFont	Initializes a **CFont** with the specified height, measured in tenths of a point, and typeface.
CreatePointFontIndirect	Same as **CreateFontIndirect** except that the font height is measured in tenths of a point rather than logical units.
FromHandle	Returns a pointer to a **CFont** object when given a Windows **HFONT**.
operator HFONT	Returns the Windows GDI font handle attached to the **CFont** object.
GetLogFont	Fills a **LOGFONT** with information about the logical font attached to the **CFont** object.

CBitmap

CBitmap	Constructs a **CBitmap** object.
LoadBitmap	Initializes the object by loading a named bitmap resource from the application's executable file and attaching the bitmap to the object.
LoadOEMBitmap	Initializes the object by loading a predefined Windows bitmap and attaching the bitmap to the object.
LoadMappedBitmap	Loads a bitmap and maps colors to current system colors.
CreateBitmap	Initializes the object with a device-dependent memory bitmap that has a specified width, height, and bit pattern.
CreateBitmapIndirect	Initializes the object with a bitmap with the width, height, and bit pattern (if one is specified) given in a **BITMAP** structure.
CreateCompatibleBitmap	Initializes the object with a bitmap so that it is compatible with a specified device.
CreateDiscardableBitmap	Initializes the object with a discardable bitmap that is compatible with a specified device.
GetBitmap	Fills a **BITMAP** structure with information about the bitmap.
operator HBITMAP	Returns the Windows handle attached to the **CBitmap** object.

CString

CString	Constructs **CString** objects in various ways.
GetLength	Returns the number of characters in a **CString** object. For multibyte characters, counts each 8-bit character; that is, a lead and trail byte in one multibyte character are counted as two characters.
IsEmpty	Tests whether a **CString** object contains no characters.
Empty	Forces a string to have 0 length.
GetAt	Returns the character at a given position.
operator []	Returns the character at a given position — operator substitution for **GetAt**.
SetAt	Sets a character at a given position.
operator LPCTSTR	Directly accesses characters stored in a **CString** object as a C-style string.
operator =	Assigns a new value to a **CString** object.
operator +	Concatenates two strings and returns a new string.
operator +=	Concatenates a new string to the end of an

	existing string.
operator == <, etc.	Comparison operators (case sensitive).
Compare	Compares two strings (case sensitive).
CompareNoCase	Compares two strings (case insensitive).
Collate	Compares two strings (case sensitive, uses locale-specific information).
CollateNoCase	Compares two strings (case insensitive, uses locale-specific information).
Mid	Extracts the middle part of a string (like the Basic MID$ function).
Left	Extracts the left part of a string (like the Basic LEFT$ function).
Right	Extracts the right part of a string (like the Basic RIGHT$ function).
SpanIncluding	Extracts a substring that contains only the characters in a set.
SpanExcluding	Extracts a substring that contains only the characters not in a set.
MakeUpper	Converts all the characters in this string to uppercase characters.
MakeLower	Converts all the characters in this string to lowercase characters.
MakeReverse	Reverses the characters in this string.
Replace	Replaces indicated characters with other characters.
Remove	Removes indicated characters from a string.
Insert	Inserts a single character or a substring at the given index within the string.
Delete	Deletes a character or characters from a string.
Format	Format the string as **sprintf** does.
FormatV	Formats the string as **vsprintf** does.
TrimLeft	Trim leading whitespace characters from the string.
TrimRight	Trim trailing whitespace characters from the string.
FormatMessage	Formats a message string.
Find	Finds a character or substring inside a larger string.
ReverseFind	Finds a character inside a larger string; starts from the end.
FindOneOf	Finds the first matching character from a set.
operator <<	Inserts a **CString** object to an archive or dump context.
operator >>	Extracts a **CString** object from an archive.
GetBuffer	Returns a pointer to the characters in the **CString**.
GetBufferSetLength	Returns a pointer to the characters in the **CString**, truncating to the specified length.
ReleaseBuffer	Releases control of the buffer returned by GetBuffer.
FreeExtra	Removes any overhead of this string object by freeing any extra memory previously allocated to the string.
LockBuffer	Disables reference counting and protects the string in the buffer.
UnlockBuffer	Enables reference counting and releases the string in the buffer.
AllocSysString	Allocates a **BSTR** from **CString** data.
SetSysString	Sets an existing **BSTR** object with data from a **CString** object.
LoadString	Loads an existing **CString** object from a

AnsiToOem	Makes an in-place conversion from the ANSI character set to the OEM character set.
OemToAnsi	Makes an in-place conversion from the OEM character set to the ANSI character set.

CPoint

CPoint	Constructs a **CPoint**
Offset	Adds values to the **x** and **y** members of the **CPoint**.
operator ==	Checks for equality between two points.
operator !=	Checks for inequality between two points.
operator +=	Offsets **CPoint** by adding a size or point.
operator –=	Offsets **CPoint** by subtracting a size or point.
operator +	Returns the sum of a **CPoint** and a size or point.
operator –	Returns the difference of a **CPoint** and a size, or the negation of a point.
operator –	Returns the size difference between two points.
operator +	Returns a **CRect** offset by a size.
operator –	Returns a **CRect** offset by a negative size.

CSize

CSize	Constructs a **CSize** object.
operator ==	Checks for equality between **CSize** and a size.
operator !=	Checks for inequality between **CSize** and a size.
operator +=	Adds a size to **CSize**.
operator –=	Subtracts a size from **CSize**.
operator +	Adds two sizes.
operator –	Subtracts two sizes.

CRect

CRect	Constructs a **CRect** object.
Width	Calculates the width of **CRect**.
Height	Calculates the height of **CRect**.
Size	Calculates the size of **CRect**.
TopLeft	Returns the top-left point of **CRect**.
BottomRight	Returns the bottom-right point of **CRect**.
CenterPoint	Returns the centerpoint of **CRect**.
IsRectEmpty	Determines whether **CRect** is empty. **CRect** is empty if the width and/or height are 0.
IsRectNull	Determines whether the **top**, **bottom**, **left**, and **right** member variables are all equal to 0.
PtInRect	Determines whether the specified point lies within **CRect**.
SetRect	Sets the dimensions of **CRect**.
SetRectEmpty	Sets **CRect** to an empty rectangle (all coordinates equal to 0).
CopyRect	Copies the dimensions of a source rectangle to **CRect**.
EqualRect	Determines whether **CRect** is equal to the given rectangle.
InflateRect	Increases the width and height of **CRect**.
DeflateRect	Decreases the width and height of **CRect**.
NormalizeRect	Standardizes the height and width of **CRect**.
OffsetRect	Moves **CRect** by the specified offsets.
SubtractRect	Subtracts one rectangle from another.
IntersectRect	Sets **CRect** equal to the intersection of two rectangles.
UnionRect	Sets **CRect** equal to the union of two rectangles.
operator LPCRECT	Converts a **CRect** to an **LPCRECT**.
operator LPRECT	Converts a **CRect** to an **LPRECT**.
operator =	Copies the dimensions of a rectangle to **CRect**.
operator ==	Determines whether **CRect** is equal to a rectangle.
operator !=	Determines whether **CRect** is not equal to a rectangle.
operator +=	Adds the specified offsets to **CRect** or inflates **CRect**.
operator –=	Subtracts the specified offsets from **CRect** or deflates **CRect**.
operator &=	Sets **CRect** equal to the intersection of **CRect** and a rectangle.
operator |=	Sets **CRect** equal to the union of **CRect** and a rectangle.
operator +	Adds the given offsets to **CRect** or inflates **CRect** and returns the resulting **CRect**.
operator –	Subtracts the given offsets from **CRect** or deflates **CRect** and returns the resulting **CRect**.
operator &	Creates the intersection of **CRect** and a rectangle and returns the resulting **CRect**.
operator |	Creates the union of **CRect** and a rectangle and returns the resulting **CRect**.

CTime

CTime	Constructs **CTime** objects in various ways.
GetCurrentTime	Creates a **CTime** object that represents the current time (static member function).
GetTime	Returns a **time_t** that corresponds to this **CTime** object.
GetYear	Returns the year that this **CTime** object represents.
GetMonth	Returns the month that this **CTime** object represents (1 through 12).
GetDay	Returns the day that this **CTime** object represents (1 through 31).
GetHour	Returns the hour that this **CTime** object represents (0 through 23).
GetMinute	Returns the minute that this **CTime** object represents (0 through 59).
GetSecond	Returns the second that this **CTime** object represents (0 through 61).
GetDayOfWeek	Returns the day of the week (1 for Sunday, 2 for Monday, and so forth).
GetGmtTm	Breaks down a **CTime** object into components — based on UTC.
GetLocalTm	Breaks down a **CTime** object into components — based on the local time zone.
GetAsSystemTime	Converts the time information stored in the **CTime** object to a Win32-compatible SYSTEMTIME structure
Format	Converts a **CTime** object into a formatted string — based on the local time zone.
FormatGmt	Converts a **CTime** object into a formatted string — based on UTC.
operator =	Assigns new time values.
operator + –	Add and subtract **CTimeSpan** and **CTime** objects.

operator +=, –=	Add and subtract a **CTimeSpan** object to and from this **CTime** object.
operator ==, < , etc.	Compare two absolute times.
operator <<	Outputs a **CTime** object to **CArchive** or **CDumpContext**.
operator >>	Inputs a **CTime** object from **CArchive** .

CTimeSpan

CTimeSpan	Constructs **CTimeSpan** objects in various ways.
GetDays	Returns the number of complete days in this **CTimeSpan**.
GetHours	Returns the number of hours in the current day (–23 through 23).
GetTotalHours	Returns the total number of complete hours in this **CTimeSpan**.
GetMinutes	Returns the number of minutes in the current hour (–59 through 59).
GetTotalMinutes	Returns the total number of complete minutes in this **CTimeSpan**.
GetSeconds	Returns the number of seconds in the current minute (–59 through 59).
GetTotalSeconds	Returns the total number of complete seconds in this **CTimeSpan**.
Format	Converts a **CTimeSpan** into a formatted string.
operator =	Assigns new time-span values.
operator + –	Adds and subtracts **CTimeSpan** objects.
operator += –=	Adds and subtracts a **CTimeSpan** object to and from this **CTimeSpan**.
operator == < etc.	Compares two relative time values.
operator <<	Outputs a **CTimeSpan** object to **CArchive** or **CDumpContext**.
operator >>	Inputs a **CTimeSpan** object from **CArchive** .

CArray

CArray	Constructs an empty array.
GetSize	Gets the number of elements in this array.
GetUpperBound	Returns the largest valid index.
SetSize	Sets the number of elements to be contained in this array.
FreeExtra	Frees all unused memory above the current upper bound.
RemoveAll	Removes all the elements from this array.
GetAt	Returns the value at a given index.
SetAt	Sets the value for a given index; array not allowed to grow.
ElementAt	Returns a temporary reference to the element pointer within the array.
GetData	Allows access to elements in the array. Can be **NULL**.
SetAtGrow	Sets the value for a given index; grows the array if necessary.
Add	Adds an element to the end of the array; grows the array if necessary.
Append	Appends another array to the array; grows the array if necessary
Copy	Copies another array to the array; grows the array if necessary.
InsertAt	Inserts an element (or all the elements in another array) at a specified index.
RemoveAt	Removes an element at a specific index.
operator []	Sets or gets the element at the specified index.

CList

CList	Constructs an empty ordered list.
GetHead	Returns the head element of the list (cannot be empty).
GetTail	Returns the tail element of the list (cannot be empty).
RemoveHead	Removes the element from the head of the list.
RemoveTail	Removes the element from the tail of the list.
AddHead	Adds an element (or all the elements in another list) to the head of the list (makes a new head).
AddTail	Adds an element (or all the elements in another list) to the tail of the list (makes a new tail).
RemoveAll	Removes all the elements from this list.
GetHeadPosition	Returns the position of the head element of the list.
GetTailPosition	Returns the position of the tail element of the list.
GetNext	Gets the next element for iterating.
GetPrev	Gets the previous element for iterating.
GetAt	Gets the element at a given position.
SetAt	Sets the element at a given position.
RemoveAt	Removes an element from this list, specified by position.
InsertBefore	Inserts a new element before a given position.
InsertAfter	Inserts a new element after a given position.
Find	Gets the position of an element specified by pointer value.
FindIndex	Gets the position of an element specified by a zero-based index.
GetCount	Returns the number of elements in this list.
IsEmpty	Tests for the empty list condition (no elements).

CMap

CMap	Constructs a collection that maps keys to values.
Lookup	Looks up the value mapped to a given key.
SetAt	Inserts an element into the map; replaces an existing element if a matching key is found.
operator []	Inserts an element into the map — operator substitution for **SetAt**.
RemoveKey	Removes an element specified by a key.
RemoveAll	Removes all the elements from this map.
GetStartPosition	Returns the position of the first element.
GetNextAssoc	Gets the next element for iterating.
GetHashTableSize	Returns the size (number of elements) of the hash table.
InitHashTable	Initializes the hash table and specifies its size.
GetCount	Returns the number of elements in this map.
IsEmpty	Tests for the empty-map condition (no elements).

CObArray

CObArray	Constructs an empty array for **CObject** pointers.
GetSize	Gets the number of elements in this array.
GetUpperBound	Returns the largest valid index.
SetSize	Sets the number of elements to be contained in this array.
FreeExtra	Frees all unused memory above the current upper bound.
RemoveAll	Removes all the elements from this array.
GetAt	Returns the value at a given index.
SetAt	Sets the value for a given index; array not allowed to grow.
ElementAt	Returns a temporary reference to the element pointer within the array.
GetData	Allows access to elements in the array. Can be **NULL**.
SetAtGrow	Sets the value for a given index; grows the array if necessary.
Add	Adds an element to the end of the array; grows the array if necessary.
Append	Appends another array to the array; grows the array if necessary.
Copy	Copies another array to the array; grows the array if necessary.
InsertAt	Inserts an element (or all the elements in another array) at a specified index.
RemoveAt	Removes an element at a specific index.
operator []	Sets or gets the element at the specified index.

CFile

m_hFile	Usually contains the operating-system file handle.
CFile	Constructs a **CFile** object from a path or file handle.
Abort	Closes a file ignoring all warnings and errors.
Duplicate	Constructs a duplicate object based on this file.
Open	Safely opens a file with an error-testing option.
Close	Closes a file and deletes the object.
Read	Reads (unbuffered) data from a file at the current file position.
ReadHuge	Can read more than 64K of (unbuffered) data from a file at the current file position. Obsolete in 32-bit programming. See Read.
Write	Writes (unbuffered) data in a file to the current file position.
WriteHuge	Can write more than 64K of (unbuffered) data in a file to the current file position. Obsolete in 32-bit programming. See Write.
Flush	Flushes any data yet to be written.
Seek	Positions the current file pointer.
SeekToBegin	Positions the current file pointer at the beginning of the file.
SeekToEnd	Positions the current file pointer at the end of the file.
GetLength	Retrieves the length of the file.
SetLength	Changes the length of the file.
LockRange	Locks a range of bytes in a file.
UnlockRange	Unlocks a range of bytes in a file.
GetPosition	Retrieves the current file pointer.
GetStatus	Retrieves the status of this open file.
GetFileName	Retrieves the filename of the selected file.
GetFileTitle	Retrieves the title of the selected file.
GetFilePath	Retrieves the full file path of the selected file.
SetFilePath	Sets the full file path of the selected file.
Rename	Renames the specified file (static function).
Remove	Deletes the specified file (static function).

GetStatus	Retrieves the status of the specified file (static, virtual function).
SetStatus	Sets the status of the specified file (static, virtual function).

CStdioFile

m_pStream	Contains a pointer to an open file.
CStdioFile	Constructs a **CStdioFile** object from a path or file pointer.
ReadString	Reads a single line of text.
WriteString	Writes a single line of text.

CAsyncSocket

CAsyncSocket	Constructs a **CAsyncSocket** object.
Create	Creates a socket.
Attach	Attaches a socket handle to a **CAsyncSocket** object.
Detach	Detaches a socket handle from a **CAsyncSocket** object.
FromHandle	Returns a pointer to a **CAsyncSocket** object, given a socket handle.
GetLastError	Gets the error status for the last operation that failed.
GetPeerName	Gets the address of the peer socket to which the socket is connected.
GetSockName	Gets the local name for a socket.
GetSockOpt	Retrieves a socket option.
SetSockOpt	Sets a socket option.
Accept	Accepts a connection on the socket.
AsyncSelect	Requests event notification for the socket.
Bind	Associates a local address with the socket.
Close	Closes the socket.
Connect	Establishes a connection to a peer socket.
IOCtl	Controls the mode of the socket.
Listen	Establishes a socket to listen for incoming connection requests.
Receive	Receives data from the socket.
ReceiveFrom	Receives a datagram and stores the source address.
Send	Sends data to a connected socket.
SendTo	Sends data to a specific destination.
ShutDown	Disables **Send** and/or **Receive** calls on the socket.
OnAccept	Notifies a listening socket that it can accept pending connection requests by calling **Accept**.
OnClose	Notifies a socket that the socket connected to it has closed.
OnConnect	Notifies a connecting socket that the connection attempt is complete, whether successfully or in error.
OnOutOfBandData	Notifies a receiving socket that there is out-of-band data to be read on the socket, usually an urgent message.
OnReceive	Notifies a listening socket that there is data to be retrieved by calling **Receive**.
OnSend	Notifies a socket that it can send data by calling **Send**.
m_hSocket	Indicates the **SOCKET** handle attached to this **CAsyncSocket** object.

CSocket

CSocket	Constructs a **CSocket** object.
Create	Creates a socket.
IsBlocking	Determines whether a blocking call is in progress.
FromHandle	Returns a pointer to a **CSocket** object, given a **SOCKET** handle.
Attach	Attaches a **SOCKET** handle to a **CSocket** object.
CancelBlockingCall	Cancels a blocking call that is currently in progress.
OnMessagePending	Called to process pending messages while waiting for a blocking call to complete.

CWinThread

m_bAutoDelete	Specifies whether to destroy the object at thread termination.
m_hThread	Handle to the current thread.
m_nThreadID	ID of the current thread.
m_pMainWnd	Holds a pointer to the application's main window.
m_pActiveWnd	Pointer to the main window of the container application when an OLE server is in-place active.
CWinThread	Constructs a **CWinThread** object.
CreateThread	Starts execution of a **CWinThread** object.
GetMainWnd	Retrieves a pointer to the main window for the thread.
GetThreadPriority	Gets the priority of the current thread.
PostThreadMessage	Posts a message to another **CWinThread** object.
ResumeThread	Decrements a thread's suspend count.
SetThreadPriority	Sets the priority of the current thread.
SuspendThread	Increments a thread's suspend count.
ExitInstance	Override to clean up when your thread terminates.
InitInstance	Override to perform thread instance initialization.
OnIdle	Override to perform thread-specific idle-time processing.
PreTranslateMessage	Filters messages before they are dispatched to the Windows functions TranslateMessage and DispatchMessage.
IsIdleMessage	Checks for special messages.
ProcessWndProcException	Intercepts all unhandled exceptions thrown by the thread's message and command handlers.
ProcessMessageFilter	Intercepts certain messages before they reach the application.
Run	Controlling function for threads with a message pump. Override to customize the default message loop.

CCmdTarget

FromIDispatch	Returns a pointer to the **CCmdTarget** object associated with the **IDispatch** pointer.
GetIDispatch	Returns a pointer to the **IDispatch** object associated with the **CCmdTarget** object.
IsResultExpected	Returns nonzero if an automation function should return a value.
BeginWaitCursor	Displays the cursor as an hourglass cursor.
EnableAutomation	Allows OLE automation for the **CCmdTarget** object.
EndWaitCursor	Returns to the previous cursor.
RestoreWaitCursor	Restores the hourglass cursor.
OnCmdMsg	Routes and dispatches command messages.
OnFinalRelease	Cleans up after the last OLE reference is released.

CCmdUI

m_nID	The ID of the user-interface object.
m_nIndex	The index of the user-interface object.
m_pMenu	Points to the menu represented by the **CCmdUI** object.
m_pSubMenu	Points to the contained sub-menu represented by the **CCmdUI** object.
m_pOther	Points to the window object that sent the notification.
Enable	Enables or disables the user-interface item for this command.
SetCheck	Sets the check state of the user-interface item for this command.
SetRadio	Like the **SetCheck** member function, but operates on radio groups.
SetText	Sets the text for the user-interface item for this command.
ContinueRouting	Tells the command-routing mechanism to continue routing the current message down the chain of handlers.

CControlBar

m_bAutoDelete	If nonzero, the **CControlBar** object is deleted when the Windows control bar is destroyed.
GetBarStyle	Retrieves the control bar style settings.
SetBarStyle	Modifies the control bar style settings.
GetBorders	Retrieves the border values of the control bar.
SetBorders	Sets the border values of the control bar.
GetCount	Returns the number of non-**HWND** elements in the control bar.
GetDockingFrame	Returns a pointer to the frame to which a control bar is docked.
IsFloating	Returns a nonzero value if the control bar in question is a floating control bar.
CalcFixedLayout	Returns the size of the control bar as a CSize object.
CalcDynamicLayout	Returns the size of a dynamic control bar as a CSize object.
OnUpdateCmdUI	Calls the Command UI handlers.
EnableDocking	Allows a control bar to be docked or floating.

CMenu

m_hMenu	Specifies the handle to the Windows menu attached to the **CMenu** object.
CMenu	Constructs a **CMenu** object.
Attach	Attaches a Windows menu handle to a **CMenu** object.
Detach	Detaches a Windows menu handle from a **CMenu** object and returns the handle.
FromHandle	Returns a pointer to a **CMenu** object given a Windows menu handle.
GetSafeHmenu	Returns the **m_hMenu** wrapped by this

	CMenu object.
DeleteTempMap	Deletes any temporary **CMenu** objects created by the **FromHandle** member function.
CreateMenu	Creates an empty menu and attaches it to a **CMenu** object.
CreatePopupMenu	Creates an empty pop-up menu and attaches it to a **CMenu** object.
LoadMenu	Loads a menu resource from the executable file and attaches it to a **CMenu** object.
LoadMenuIndirect	Loads a menu from a menu template in memory and attaches it to a **CMenu** object.
DestroyMenu	Destroys the menu attached to a **CMenu** object and frees any memory that the menu occupied.
DeleteMenu	Deletes a specified item from the menu. If the menu item has an associated pop-up menu, destroys the handle to the pop-up menu and frees the memory used by it.
TrackPopupMenu	Displays a floating pop-up menu at the specified location and tracks the selection of items on the pop-up menu.
AppendMenu	Appends a new item to the end of this menu.
CheckMenuItem	Places a check mark next to or removes a check mark from a menu item in the pop-up menu.
CheckMenuRadioItem	Places a radio button next to a menu item and removes the radio button from all of the other menu items in the group.
SetDefaultItem	Sets the default menu item for the specified menu.
GetDefaultItem	Determines the default menu item on the specified menu.
EnableMenuItem	Enables, disables, or dims (grays) a menu item.
GetMenuItemCount	Determines the number of items in a pop-up or top-level menu.
GetMenuItemID	Obtains the menu-item identifier for a menu item located at the specified position.
GetMenuState	Returns the status of the specified menu item or the number of items in a pop-up menu.
GetMenuString	Retrieves the label of the specified menu item.
GetMenuItemInfo	Retrieves information about a menu item.
GetSubMenu	Retrieves a pointer to a pop-up menu.
InsertMenu	Inserts a new menu item at the specified position, moving other items down the menu.
ModifyMenu	Changes an existing menu item at the specified position.
RemoveMenu	Deletes a menu item with an associated pop-up menu from the specified menu.
SetMenuItemBitmaps	Associates the specified check-mark bitmaps with a menu item.
GetMenuContextHelpId	Retrieves the help context ID associated with the menu.
SetMenuContextHelpId	Sets the help context ID to be associated with the menu.
DrawItem	Called by the framework when a visual aspect of an owner-drawn menu changes.
MeasureItem	Called by the framework to determine menu dimensions when an owner-drawn menu is created.

CMemoryState

CMemoryState	Constructs a class-like structure that controls
Checkpoint	Obtains a snapshot or "checkpoint" of the current memory state.
Difference	Computes the difference between two objects of type **CMemoryState**.
DumpAllObjectsSince	Dumps a summary of all currently allocated objects since a previous checkpoint.
DumpStatistics	Prints memory allocation statistics for a **CMemoryState** object.

CPropertySheet

m_psh	The Windows PROPSHEETHEADER structure. Provides access to basic property sheet parameters.
CPropertySheet	Constructs a **CPropertySheet** object.
Construct	Constructs a **CPropertySheet** object.
GetActiveIndex	Retrieves the index of the active page of the property sheet.
GetPageIndex	Retrieves the index of the specified page of the property sheet.
GetPageCount	Retrieves the number of pages in the property sheet.
GetPage	Retrieves a pointer to the specified page.
GetActivePage	Returns the active page object.
SetActivePage	Programmatically sets the active page object.
SetTitle	Sets the caption of the property sheet.
GetTabControl	Retrieves a pointer to a tab control.
SetFinishText	Sets the text for the Finish button.
SetWizardButtons	Enables the wizard buttons.
SetWizardMode	Enables the wizard mode.
EnableStackedTabs	Indicates whether the property sheet uses stacked or scrolling tabs.
DoModal	Displays a modal property sheet.
Create	Displays a modeless property sheet.
AddPage	Adds a page to the property sheet.
RemovePage	Removes a page from the property sheet.
PressButton	Simulates the choice of the specified button in a property sheet.
EndDialog	Terminates the property sheet.

CArchive

m_pDocument	Points to the **CDocument** object being serialized.
CArchive	Creates a **CArchive** object.
Abort	Closes an archive without throwing an exception.
Close	Flushes unwritten data and disconnects from the **CFile**.
Flush	Flushes unwritten data from the archive buffer.
operator >>	Loads objects and primitive types from the archive.
operator <<	Stores objects and primitive types to the archive.
Read	Reads raw bytes.
Write	Writes raw bytes.
WriteString	Writes a single line of text.
ReadString	Reads a single line of text.
GetFile	Gets the **CFile** object pointer for this archive.
GetObjectSchema	Called from the **Serialize** function to determine the version of the object that is being deserialized.
SetObjectSchema	Sets the object schema stored in the archive object.
IsLoading	Determines whether the archive is loading.
IsStoring	Determines whether the archive is storing.
IsBufferEmpty	Determines whether the buffer has been emptied during a Windows Sockets receive process.
ReadObject	Calls an object's **Serialize** function for loading.
WriteObject	Calls an object's **Serialize** function for storing.
MapObject	Places objects in the map that are not serialized to the file, but that are available for subobjects to reference.
SetStoreParams	Sets the hash table size and the block size of the map used to identify unique objects during the serialization process.
SetLoadParams	Sets the size to which the load array grows. Must be called before any object is loaded or before **MapObject** or **ReadObject** is called.
ReadClass	Reads a class reference previously stored with **WriteClass**.
WriteClass	Writes a reference to the **CRuntimeClass** to the **CArchive**.
SerializeClass	Reads or writes the class reference to the **CArchive** object depending on the direction of the **CArchive**.

CDocTemplate

CDocTemplate	Constructs a **CDocTemplate** object.
SetContainerInfo	Determines the resources for OLE containers when editing an in-place OLE item.
SetServerInfo	Determines the resources and classes when the server document is embedded or edited in-place.
GetFirstDocPosition	Retrieves the position of the first document associated with this template.
GetNextDoc	Retrieves a document and the position of the next one.
LoadTemplate	Loads the resources for a given **CDocTemplate** or derived class.
AddDocument	Adds a document to a template.
RemoveDocument	Removes a document from a template.
GetDocString	Retrieves a string associated with the document type.
CreateOleFrame	Creates an OLE-enabled frame window.
MatchDocType	Determines the degree of confidence in the match between a document type and this template.
CreateNewDocument	Creates a new document.
CreateNewFrame	Creates a new frame window containing a document and view.
InitialUpdateFrame	Initializes the frame window, and optionally makes it visible.
SaveAllModified	Saves all documents associated with this template which have been modified.
CloseAllDocuments	Closes all documents associated with this template.
OpenDocumentFile	Opens a file specified by a pathname.
SetDefaultTitle	Displays the default title in the document window's title bar.

Diagnostic Macros

ASSERT	Prints a message and then aborts the program if the specified expression evaluates to **FALSE** in the Debug version of the library.
ASSERT_KINDOF	Tests that an object is an object of the specified class or of a class derived from the specified class.
ASSERT_VALID	Tests the internal validity of an object by calling its **AssertValid** member function; typically overridden from **CObject**.
DEBUG_NEW	Supplies a filename and line number for all object allocations in Debug mode to help find memory leaks.
TRACE	Provides **printf**-like capability in the Debug version of the library.
TRACE0	Similar to **TRACE** but takes a format string with no arguments.
TRACE1	Similar to **TRACE** but takes a format string with a single argument.
TRACE2	Similar to **TRACE** but takes a format string with two arguments.
TRACE3	Similar to **TRACE** but takes a format string with three arguments.
VERIFY	Similar to **ASSERT** but evaluates the expression in the Release version of the library as well as in the Debug version.

Afx functions

AfxFreeLibrary	Decrements the reference count of the loaded dynamic-link library (DLL) module; when the reference count reaches zero, the module is unmapped.
AfxGetApp	Returns a pointer to the application's single **CWinApp** object.
AfxGetAppName	Returns a string containing the application's name.
AfxGetInstanceHandle	Returns an **HINSTANCE** representing this instance of the application.
AfxGetMainWnd	Returns a pointer to the current "main" window of a non-OLE application, or the in-place frame window of a server application.
AfxGetResourceHandle	Returns an **HINSTANCE** to the source of the application's default resources. Use this to access the application's resources directly.
AfxInitRichEdit	Initializes the rich edit control for the application and initializes the common controls library, if the library hasn't already been initialized for the process.
AfxLoadLibrary	Maps a DLL module and returns a handle that can be used to get the address of a DLL function.
AfxRegisterWndClass	Registers a Windows window class to supplement those registered automatically by MFC.
AfxSocketInit	Called in a **CWinApp::InitInstance** override to initialize Windows Sockets.
AfxSetResourceHandle	Sets the **HINSTANCE** handle where the default resources of the application are loaded.
AfxRegisterClass	Registers a window class in a DLL that uses MFC.
AfxBeginThread	Creates a new thread.
AfxEndThread	Terminates the current thread.
AfxGetThread	Retrieves a pointer to the current CWinThread object.
AfxWinInit	Called by the MFC-supplied **WinMain** function, as part of the CWinApp initialization of a GUI-based application, to initialize MFC. Must be called directly for console applications using MFC.

Common Data Type

BOOL	Boolean value.
BSTR	32-bit character pointer.
BYTE	8-bit integer that is not signed.
COLORREF	32-bit value used as a color value.
DWORD	32-bit unsigned integer or the address of a segment and its associated offset.
LONG	32-bit signed integer.
LPARAM	32-bit value passed as a parameter to a window procedure or callback function.
LPCSTR	32-bit pointer to a constant character string.
LPSTR	32-bit pointer to a character string.
LPCTSTR	32-bit pointer to a constant character string that is portable for Unicode and DBCS.
LPTSTR	32-bit pointer to a character string that is portable for Unicode and DBCS.
LPVOID	32-bit pointer to an unspecified type.
LRESULT	32-bit value returned from a window procedure or callback function.
UINT	16-bit unsigned integer on Windows versions 3.0 and 3.1; a 32-bit unsigned integer on Win32.
WNDPROC	32-bit pointer to a window procedure.
WORD	16-bit unsigned integer.
WPARAM	value passed as a parameter to a window procedure or callback function:
POSITION	value used to denote the position of an element in a collection; used by MFC collection classes.
LPCRECT	32-bit pointer to a constant (nonmodifiable) RECT structure.

Message Map Macros

DECLARE_MESSAGE_MAP	Declares that a message map will be used in a class to map messages to functions (must be used in the class declaration).
BEGIN_MESSAGE_MAP	Begins the definition of a message map (must be used in the class implementation).
END_MESSAGE_MAP	Ends the definition of a message map (must be used in the class implementation).
ON_COMMAND	Indicates which function will handle a specified command message.
ON_CONTROL	Indicates which function will handle a specified control-notification message.
ON_MESSAGE	Indicates which function will handle a user-defined message.
ON_OLECMD	Indicates which function will handle a menu command from a DocObject or its container.
ON_REGISTERED_MESSAGE	Indicates which function will handle a registered user-defined message.
ON_REGISTERED_THREAD_MESSAGE	Indicates which function will handle a registered user-defined message when you have a **CWinThread** class.
ON_THREAD_MESSAGE	Indicates which function will handle a user-defined message when you have a **CWinThread** class.
ON_UPDATE_COMMAND_UI	Indicates which function will handle a specified user-interface update command message.
ON_COMMAND_RANGE	Indicates which function will handle the range of command IDs specified in the first two parameters to the macro.
ON_UPDATE_COMMAND_UI_RANGE	Indicates which update handler will handle the range of command IDs specified in the first two parameters to the macro.
ON_CONTROL_RANGE	Indicates which function will handle notifications from the range of control IDs specified in the second and third parameters to the macro. The first parameter is a control-notification message, such as **BN_CLICKED**.

WM_Messages Handlers

ON_WM_ACTIVATE( )	Afx_msg void OnActivate(UINT, CWnd*, BOOL);
ON_WM_ACTIVATEAPP( )	afx_msg void OnActivateApp(BOOL, HANDLE);
ON_WM_ASKCBFORMATNAME( )	afx_msg void OnAskCbFormatName(UINT, LPSTR);
ON_WM_CANCELMODE( )	afx_msg void OnCancelMode( );
ON_WM_CAPTURECHANGED( )	afx_msg void OnCaptureChanged(CWnd*);
ON_WM_CHANGECBCHAIN( )	afx_msg void OnChangeCbChain(HWND, HWND);
ON_WM_CHAR( )	afx_msg void OnChar(UINT, UINT, UINT);
ON_WM_CHARTOITEM( )	afx_msg int OnCharToItem(UINT, CWnd*, UINT);
ON_WM_CHILDACTIVATE( )	afx_msg void OnChildActivate( );
ON_WM_CLOSE( )	afx_msg void OnClose( );
ON_WM_COMPACTING( )	afx_msg void OnCompacting(UINT);
ON_WM_COMPAREITEM( )	afx_msg int OnCompareItem(LPCOMPAREITEMSTRUCT);
ON_WM_CONTEXTMENU( )	afx_msg void OnContextMenu(CWnd*, CPoint);
ON_WM_COPYDATA( )	afx_msg BOOL OnCopyData(CWnd* pWnd, COPYDATASTRUCT* pCopyDataStruct);
ON_WM_CREATE( )	afx_msg int OnCreate(LPCREATESTRUCT);
ON_WM_CTLCOLOR( )	afx_msg HBRUSH OnCtlColor(CDC*, CWnd*, UINT);
ON_WM_DEADCHAR( )	afx_msg void OnDeadChar(UINT, UINT, UINT);
ON_WM_DELETEITEM( )	afx_msg void OnDeleteItem(LPDELETEITEMSTRUCT);
ON_WM_DESTROY( )	afx_msg void OnDestroy( );
ON_WM_DESTROYCLIPBOARD( )	afx_msg void OnDestroyClipboard( );
ON_WM_DEVICECHANGE( )	afx_msg void OnDeviceChange(UINT, DWORD);
ON_WM_DEVMODECHANGE( )	afx_msg void OnDevModeChange(LPSTR);

ON_WM_DRAWCLIPBOARD( )	afx_msg void OnDrawClipboard( );
ON_WM_DRAWITEM( )	afx_msg void OnDrawItem(LPDRAWITEMSTRUCT);
ON_WM_DROPFILES( )	afx_msg void OnDropFiles(HDROP);
ON_WM_ENABLE( )	afx_msg void OnEnable(BOOL);
ON_WM_ENDSESSION( )	afx_msg void OnEndSession(BOOL);
ON_WM_ENTERIDLE( )	afx_msg void OnEnterIdle(UINT, CWnd*);
ON_WM_ERASEBKGND( )	afx_msg BOOL OnEraseBkgnd(CDC *);
ON_WM_FONTCHANGE( )	afx_msg void OnFontChange( );
ON_WM_GETDLGCODE( )	afx_msg UINT OnGetDlgCode( );
ON_WM_GETMINMAXINFO( )	afx_msg void OnGetMinMaxInfo(LPPOINT);
ON_WM_HELPINFO( )	afx_msg BOOL OnHelpInfo(HELPINFO*);
ON_WM_HSCROLL( )	afx_msg void OnHScroll(UINT, UINT, CWnd*);
ON_WM_HSCROLLCLIPBOARD( )	afx_msg void OnHScrollClipboard CWnd*, UINT, UINT);
ON_WM_ICONERASEBKGND( )	afx_msg void OnIconEraseBkgnd(CDC *);
ON_WM_INITMENU( )	afx_msg void OnInitMenu(CMenu *);
ON_WM_INITMENUPOPUP( )	afx_msg void OnInitMenuPopup(CMenu *, UINT, BOOL);
ON_WM_KEYDOWN( )	afx_msg void OnKeyDown(UINT, UINT, UINT);
ON_WM_KEYUP( )	afx_msg void OnKeyUp(UINT, UINT, UINT);
ON_WM_KILLFOCUS( )	afx_msg void OnKillFocus(CWnd*);
ON_WM_LBUTTONDBLCLK( )	afx_msg void OnLButtonDblClk(UINT, CPoint);
ON_WM_LBUTTONDOWN( )	afx_msg void OnLButtonDown(UINT, CPoint);
ON_WM_LBUTTONUP( )	afx_msg void OnLButtonUp(UINT, CPoint);
ON_WM_MBUTTONDBLCLK( )	afx_msg void OnMButtonDblClk(UINT, CPoint);
ON_WM_MBUTTONDOWN( )	afx_msg void OnMButtonDown(UINT, CPoint);
ON_WM_MBUTTONUP( )	afx_msg void OnMButtonUp(UINT, CPoint);
ON_WM_MDIACTIVATE( )	afx_msg void OnMDIActivate(BOOL, CWnd*, CWnd*);
ON_WM_MEASUREITEM( )	afx_msg void OnMeasureItem(LPMEASUREITEMSTRUCT);
ON_WM_MENUCHAR( )	afx_msg LONG OnMenuChar(UINT, UINT, CMenu *);
ON_WM_MENUSELECT( )	afx_msg void OnMenuSelect(UINT, UINT, HMENU);
ON_WM_MOUSEACTIVATE( )	afx_msg int OnMouseActivate(CWnd*, UINT, UINT);
ON_WM_MOUSEMOVE( )	afx_msg void OnMouseMove(UINT, CPoint);
ON_WM_MOUSEWHEEL( )	afx_msg BOOL OnMouseWheel(UINT, short, CPoint);
ON_WM_MOVE( )	afx_msg void OnMove(int, int);
ON_WM_MOVING( )	afx_msg void OnMoving(UINT, LPRECT);
ON_WM_NCACTIVATE( )	afx_msg BOOL OnNcActivate(BOOL);
ON_WM_NCCALCSIZE( )	afx_msg void OnNcCalcSize(BOOL, NCCALCSIZE_PARAMS FAR*);
ON_WM_NCCREATE( )	afx_msg BOOL OnNcCreate(LPCREATESTRUCT);
ON_WM_NCDESTROY( )	afx_msg void OnNcDestroy( );
ON_WM_NCHITTEST( )	afx_msg UINT OnNcHitTest(CPoint);
ON_WM_NCLBUTTONDBLCLK( )	afx_msg void OnNcLButtonDblClk(UINT, CPoint);
ON_WM_NCLBUTTONDOWN( )	afx_msg void OnNcLButtonDown(UINT, CPoint);
ON_WM_NCLBUTTONUP( )	afx_msg void OnNcLButtonUp(UINT, CPoint);
ON_WM_NCMBUTTONDBLCLK( )	afx_msg void OnNcMButtonDblClk(UINT, CPoint);
ON_WM_NCMBUTTONDOWN( )	afx_msg void OnNcMButtonDown(UINT, CPoint);
ON_WM_NCMBUTTONUP( )	afx_msg void OnNcMButtonUp(UINT, CPoint);
ON_WM_NCMOUSEMOVE( )	afx_msg void OnNcMouseMove(UINT, CPoint);
ON_WM_NCPAINT( )	afx_msg void OnNcPaint( );
ON_WM_NCRBUTTONDBLCLK( )	afx_msg void OnNcRButtonDblClk(UINT , CPoint);
ON_WM_NCRBUTTONDOWN( )	afx_msg void OnNcRButtonDown(UINT, CPoint);
ON_WM_NCRBUTTONUP( )	afx_msg void OnNcRButtonUp(UINT, CPoint);
ON_WM_PAINT( )	afx_msg void OnPaint( );
ON_WM_PAINTCLIPBOARD( )	afx_msg void OnPaintClipboard(CWnd*, HANDLE);
ON_WM_PALETTECHANGED( )	afx_msg void OnPaletteChanged(CWnd*);
ON_WM_PALETTEISCHANGING( )	afx_msg void OnPaletteIsChanging(CWnd*);
ON_WM_PARENTNOTIFY( )	afx_msg void OnParentNotify(UINT, LONG);
ON_WM_QUERYDRAGICON( )	afx_msg HCURSOR OnQueryDragIcon( );
ON_WM_QUERYENDSESSION( )	afx_msg BOOL OnQueryEndSession( );
ON_WM_QUERYNEWPALETTE( )	afx_msg BOOL OnQueryNewPalette( );
ON_WM_QUERYOPEN( )	afx_msg BOOL OnQueryOpen( );
ON_WM_RBUTTONDBLCLK( )	afx_msg void OnRButtonDblClk(UINT, CPoint);
ON_WM_RBUTTONDOWN( )	afx_msg void OnRButtonDown(UINT, CPoint);
ON_WM_RBUTTONUP( )	afx_msg void OnRButtonUp(UINT, CPoint);
ON_WM_RENDERALLFORMATS( )	afx_msg void OnRenderAllFormats( );
ON_WM_RENDERFORMAT( )	afx_msg void OnRenderFormat(UINT);
ON_WM_SETCURSOR( )	afx_msg BOOL OnSetCursor(CWnd*, UINT, UINT);
ON_WM_SETFOCUS( )	afx_msg void OnSetFocus(CWnd*);
ON_WM_SHOWWINDOW( )	afx_msg void OnShowWindow(BOOL, UINT);
ON_WM_SIZE( )	afx_msg void OnSize(UINT, int, int);
ON_WM_SIZECLIPBOARD( )	afx_msg void OnSizeClipboard(CWnd*, HANDLE);
ON_WM_SIZING( )	afx_msg void OnSizing(UINT, LPRECT);
ON_WM_SPOOLERSTATUS( )	afx_msg void OnSpoolerStatus(UINT, UINT);
ON_WM_STYLECHANGED( )	afx_msg void OnStyleChanged(int, LPSTYLESTRUCT);
ON_WM_STYLECHANGING( )	afx_msg void OnStyleChanging(int, LPSTYLESTRUCT);
ON_WM_SYSCHAR( )	afx_msg void OnSysChar(UINT, UINT, UINT);
ON_WM_SYSCOLORCHANGE( )	afx_msg void OnSysColorChange( );
ON_WM_SYSCOMMAND( )	afx_msg void OnSysCommand(UINT, LONG);
ON_WM_SYSDEADCHAR( )	afx_msg void OnSysDeadChar(UINT, UINT, UINT);
ON_WM_SYSKEYDOWN( )	afx_msg void OnSysKeyDown(UINT, UINT, UINT);
ON_WM_SYSKEYUP( )	afx_msg void OnSysKeyUp(UINT, UINT, UINT);
ON_WM_TCARD( )	afx_msg void OnTCard(UINT, DWORD);
ON_WM_TIMECHANGE( )	afx_msg void OnTimeChange( );
ON_WM_TIMER( )	afx_msg void OnTimer(UINT);
ON_WM_VKEYTOITEM( )	afx_msg int OnVKeyToItem(UINT, CWnd*, UINT);
ON_WM_VSCROLL( )	afx_msg void OnVScroll(UINT, UINT, CWnd*);
ON_WM_VSCROLLCLIPBOARD( )	afx_msg void OnVScrollClipboard(CWnd*, UINT, UINT);
ON_WM_WINDOWPOSCHANGED( )	afx_msg void OnWindowPosChanged(WINDOWPOS* lpwndpos);
ON_WM_WINDOWPOSCHANGING( )	afx_msg void OnWindowPosChanging(WINDOWPOS* lpwndpos);
ON_WM_WININICHANGE( )	afx_msg void OnWinIniChange(LPSTR);

Message Handlers

ON_COMMAND(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_CONTROL(<wNotifyCode>, <id>, <memberFxn>)	afx_msg void memberFxn( );
ON_MESSAGE(	afx_msg LRESULT memberFxn(WPARAM,

<message>, <memberFxn>)	LPARAM);
ON_REGISTERED_MESSAGE(<nMessageVariable>, <memberFxn>)	afx_msg LRESULT memberFxn(WPARAM, LPARAM);
ON_THREAD_MESSAGE(<message>, <memberFxn>)	afx_msg void memberFxn(UINT, LONG);
ON_REGISTERED_THREAD_MESSAGE(<nMessageVariable>, <memberFxn>)	afx_msg void memberFxn(UINT, LONG);

Button Messages

ON_BN_CLICKED(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_BN_DISABLE(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_BN_DOUBLECLICKED(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_BN_HILITE(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_BN_PAINT(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_BN_UNHILITE(<id>, <memberFxn>)	afx_msg void memberFxn( );

ComboBox Messages

ON_CBN_CLOSEUP(<id>, <memberFxn>)	afx_msg void memberFxn( )
ON_CBN_DBLCLK(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_CBN_DROPDOWN(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_CBN_EDITCHANGE(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_CBN_EDITUPDATE(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_CBN_ERRSPACE(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_CBN_KILLFOCUS(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_CBN_SELCHANGE(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_CBN_SELENDCANCEL(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_CBN_SELENDOK(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_CBN_SETFOCUS(<id>, <memberFxn>)	afx_msg void memberFxn( );

Edit Messages

ON_EN_CHANGE(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_EN_ERRSPACE(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_EN_HSCROLL(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_EN_KILLFOCUS(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_EN_MAXTEXT(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_EN_SETFOCUS(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_EN_UPDATE(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_EN_VSCROLL(<id>, <memberFxn>)	afx_msg void memberFxn( );

ListBox Messages

ON_LBN_DBLCLK(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_LBN_ERRSPACE(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_LBN_KILLFOCUS(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_LBN_SELCHANGE(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_LBN_SETFOCUS(<id>, <memberFxn>)	afx_msg void memberFxn( );
ON_LBN_DBLCLK(<id>, <memberFxn>)	afx_msg void memberFxn( );