CWindowRecord structure

```
#include <Windows.h>

typedef struct CWindowRecord {  Size   Offset  Description
  CGrafPort      port;          108     0      portBits, portRect, pnSize, txFont
  short          windowKind;    2       110    Class; documentProc, etc.
  char           visible;       1       112    TRUE if window is visible
  char           hilited;       1       111    TRUE if window hilited
  char           goAwayFlag;    1       112    TRUE if window has a close box in
                                               top left
  char           spareFlag;     1       113    TRUE=zoom is enabled
  RgnHandle      strucRgn;      4       114    Content region plus the frame
                                               (global coords)
  RgnHandle      contRgn;       4       118    Content rgn, including scroll bars
                                               (global)
  RgnHandle      updateRgn;     4       122    Area needing update, (local)
  Handle         windowDefProc; 4       126    Code to draw the window ('WDEF')
  Handle         dataHandle;    4       130    Additional info; may lead to a
                                               WStateData struct
  StringHandle   titleHandle;   4       134    Leads to pstring of title
  short          titleWidth;    2       138    Width, in pixels, of the window title
                                               text
  ControlHandle  controlList    4       140    Window's first ControlRecord
  CWindowPeek    nextWindow;    4       144    The window behind this one (0 if
                                               this is last)
  PicHandle      windowPic;     4       148    Leads to Picture; 0=none
  long           refCon;        4       152    Anything you want
} CWindowRecord;                156

typedef CWindowRecord *CWindowPeek;   /* use CWindowPeek to access these
                                               fields */
typedef CGrafPtr  CWindowPtr;   /* Note: Not a pointer to CWindowRecord  */
```

Notes: The only difference between a **CWindowRecord** and a WindowRecord is that the CWindowRecord's port field is a cGrafPort rather than a grafPort. Because everything else about the two structures is identical, and because all non-color **Window Manager** routines work with the new structure by accepting CWindowPtrs as well as WindowPtrs, all window management changes should be transparent to your applications.

A **CWindowPeek** (ie, the address of a **CWindowRecord**) is used in nearly all Window Manager calls.

The port field is a CGrafPort (all 108 bytes of it). It contains such important items as the size and location of the window, the text font and display attributes, etc.

The windowKind field identifies which of the standard or user-defined window definition routines will draw the window.

Note: For desk accessories, windowKind contains the driver reference number (a negative value). This affects how DAs must handle calls to

IsDialogEvent.

The dataHandle field may contain either four bytes of data (as used in rDocProc type windows), or a handle to additional data needed by the window definition procedure. In the case of zoomable window types, dataHandle is a handle to a WStateData structure.

The nextWindow field contains the address of the next CWindowRecord in the Window Manager's list. The global variable WindowList (at 0x09D6) contains the address of the first (frontmost) window in that list.

Notice that a CWindowRecord begins with a CGrafPort. Similarly, a DialogRecord begins with a CWindowRecord (and thus begins with a CGrafPort). The data types CGrafPtr, CWindowPtr, and DialogPtr may be used interchangeably when you pass a pointer to a function which expects a subset. You may need to explicitly cast depending on how you have chosen to use prototypes.

```
short dlgRsrcID;

myDlg = GetNewDialog(dlgRsrcID, nil, (WindowPtr) -1);
SetPort( myDlg );          /* expects a CGrafPtr */
ShowWindow( myDlg );       /* expects a CWindowPtr */
```

To access the additional fields of a CWindowRecord structure, create a CWindowPeek variable:

```
CWindowPtr   myWin;
CWindowPeek  myWinPeek = (CWindowPeek) myWin;

myWin->txFont = geneva;              /* access CGrafPort fields */
myWinPeek->windowKind = dBoxProc; /* access CWindowRecord fields */
```

To query the contents of a field, you can use the following type coercion:

```
aLong = ((CWindowPeek)myWin)->refCon;
```