DialogRecord structure

#include <Dialogs.h>

typedef struct **DialogRecord** {		Size	Offset	Description
WindowRecord	window;	156	0	Dialog's window. See WindowRecord
Handle	items;	4	156	Leads to item list (see below for format)
TEHandle	textH;	4	160	Leads to a TERec of current editText item(gets reused for all editText items)
short	editField;	2	164	Item number -1 of current editText item
short	editOpen;	2	166	(used internally)
short	aDefItem;	2	168	Default item for alerts and modal dialogs (gets 'hit' when user presses Enter)
} **DialogRecord**;		170		

typedef DialogRecord ***DialogPeek**;
typedef WindowPtr **DialogPtr**; aka GrafPtr

Notes: A DialogRecord begins with a WindowRecord which begins with a GrafPort. The data types GrafPtr, WindowPtr, and DialogPtr may be used interchangeably when you pass a pointer to a function which expects a subset:

```
DialogPtr  myDlg;

SetPort(myDlg);          /* expects a GrafPtr */
ShowWindow(myDlg);       /* expects a WindowPtr */
```

To access the additional fields of this structure, create a DialogPeek variable:

```
DialogPtr  myDlg;
DialogPeek myDlgPeek;

myDlgPeek = (DialogPeek)myDlg;
myDlgPeek->aDefItem = 12;

// To query the contents of a field, you can use type coercion:
i = ((DialogPeek)myDlg)->aDefItem;
```

Although the format of the items field of the DialogRecord is not defined in any MPW header file, it has been defined in Macintosh Tech Note #95 which specifies how to add items to a Print Dialog. See the **AppendDITL** function of **Adding Items to the Print Dialogs** for this defintion. Please note, however, that the routines **AppendDITL**, **CountDITL** and **ShortenDITL** have been provided so that you can avoid accessing this field directly, since its format could change in the future.