HFileInfo structure

#include <Files.h>

typedef struct **HFileInfo** {

		Size	Offset	Description
ParamBlockHeader		24	0	common fields of ParamBlock types
short	ioFRefNum;	2	24	File reference number
char	ioFVersNum;	1	26	Version (best to use 0)
char	filler1;	1	27	(unused)
short	ioFDirIndex;	2	28	Index
char	ioFlAttrib;	1	30	File Attribute bits (bit 4 set if directory, etc)
char	filler2;	1	31	(unused)
FInfo	ioFlFndrInfo;	16	32	(File type, creator, flags, icon point, etc.)
long	ioDirID;	4	48	'Hard' Directory ID or file number
unsigned short	ioFlStBlk;	2	52	First allocation block of data fork
long	ioFlLgLen;	4	54	Logical end-of-file of data fork
long	ioFlPyLen;	4	68	Physical end-of-file of data fork
unsigned short	ioFlRStBlk;	2	62	First allocation block of resource fork
long	ioFlRLgLen;	4	64	Logical end-of-file of resource fork
long	ioFlRPyLen;	4	68	Physical end-of-file of resource fork
unsigned long	ioFlCrDat;	4	72	Date/Time of creation
unsigned long	ioFlMdDat;	4	76	Date/Time of last modification
unsigned long	ioFlBkDat;	4	80	Date/Time last backed up
FXInfo	ioFlXFndrInfo;	16	84	(icon ID, comment, put-away dir)
long	ioFlParID;	4	100	'Hard' ID of this dir's parent
long	ioFlClpSiz;	4	104	Allocation block size (0=use volume default)

} **HFileInfo**; 108

Notes: Use this **HFileInfo** structure in calls to **PBGetCatInfo** and **PBSetCatInfo** when you access information about a file.

Wherever a function exists to modify one of the values in this structure, you should use that function. For instance, don't use **PBSetCatInfo** to set the lock bit; instead call **PBSetFLock**.

A common way to use this structure is to allocate a CInfoPBRec union which is an aggregate of HFileInfo and DirInfo. Create and initialize a pointer to each data type and use either structure in the call to **PBGetCatInfo**. Upon return, check bit 4 of ioFlAttrib. If bit 4 is set, then the return data is about a directory and you should use the DirInfo structure; otherwise, use HFileInfo. See CInfoPBRec for examples.