PBGetVInfo Get information about a volume

#include <Files.h> **File Manager (PBxxx)**

```
OSErr      PBGetVInfo(pb, async);
ParmBlkPtr pb ;      address of a 64-byte VolumeParam structure
Boolean    async ;   0=await completion; 1=immediate return
           returns   Error Code; 0=no error
```

PBGetVInfo obtains information about a specified volume or directory, including the volume attributes, creation and modification date/time, size of allocation blocks, number of files in the root (or specified directory), etc.

pb is the address of a 64-byte VolumeParam structure. The following fields are relevant:

Out-In	Name	Type	Size	Offset	Description
–>	ioCompletion	ProcPtr	4	12	Completion routine address (if *async* =TRUE)
–>	ioVolIndex	short	2	28	(>0=index, <0=use name/num, 0=use num)
<->	ioNamePtr	StringPtr	4	18	Address of full or partial path/filename
<->	ioVRefNum	short	2	22	Volume, drive, or working directory reference
<-	ioResult	OSErr	2	16	Error Code (0=no error, 1=not done yet)
<-	ioVCrDate	long	4	30	Date/time volume created
<-	ioVLsBkUp	long	4	34	Date/time volume information was modified
<-	ioVAtrb	short	2	38	volume attributes (bit 15=locked, etc.)
<-	ioVNmFls	short	2	40	Count of files in the (root) directory
<-	ioVDirSt	short	2	42	First allocation block of directory
<-	ioVBlLn	short	2	44	Length of directory in blocks
<-	ioVNmAlBlks	short	2	46	Count of all allocation blocks
<-	ioVAlBlkSiz	long	4	48	Allocation block size, in bytes
<-	ioVClpSiz	long	4	52	Default allocation clump, in bytes
<-	ioAlBlSt	short	2	56	First block in volume block map
<-	ioVNxtFNum	long	4	58	Next unused file number
<-	ioVFrBlk	short	2	62	Count of free allocation blocks

async is a Boolean value. Use FALSE for normal (synchronous) operation or TRUE to enqueue the request and resume control immediately. See Async I/O.

Returns: an operating system Error Code. It will be one of:

noErr	(0)	No error
nsvErr	(-35)	No such volume
paramErr	(-50)	No default volume

Notes: For most applications, the high-level **GetVInfo** function is sufficient for getting the free bytes and volume name. **PBGetVInfo** and the related **PBHGetVInfo** return a lot more information.

The key to using this function is the ioVolIndex field. It will be:

positive (>0) An index; information about the *n* -th volume will be returned. Volume indexes start with 1 and go up sequentially (until the call returns nsvErr).

negative (<0) Use the ioNamePtr and/or ioVRefNum fields to specify the desired volume in the standard way (if ioNamePtr is 0 or invalid, ioVRefNum is used; if both are 0, the default volume is assumed. Note that you must set ioVRefNum to 0, for ioNamePtr to be used). If you are using ioNamePtr to specify a volume

name, the string must be in the following form

```
myVolParam.ioNamePtr = "\pMy HardDisk:"
```

The trailing colon on the string indicates that we are referring to a directory, not a file.

exactly 0 The ioVRefNum (alone) must specify the volume or a working directory.

- If you use a working directory number, then on return ioVNmFls will identify the number of files in that directory
- If you use a real volume number, then ioVNmFls will contain the number of files in the root.

Upon return, ioVRefNum will contain a real volume number, unless you used a working directory number on entry. Compare to **PBHGetVInfo**, which always stores a real volume number in this field.

The ioNamePtr field itself is not changed. If it was not NIL on entry, then the buffer it pointed to will receive a copy of the Pascal-style, length-prefixed volume name (up to 27 characters long).

The returned fields, ioVNmAlBlks and ioVFrBlk, are clipped to 31744 (0x7C00), regardless of the TRUE size of the volume. The HFS function **PBHGetVInfo** does not have this size restriction.