NOTE: This Technical Note has been retired. Please see the Technical Notes page for current documentation.

Technical Note FL530
File Manager Volume Handling Q&As

This Technical Note contains a collection of archived Q&As relating to a specific topic--questions sent the Developer Support Center (DSC) along with answers from the DSC engineers. Current Q&As can be found on the Macintosh Technical Q&As web site.

[Oct 01 1990]

20 mounted volume limitation

Date Written:

Last reviewed:

How many mounted volumes does the Macintosh support? I've heard that during startup the Macintosh mounts only the first 20. I'm writing a system extension that will boot additional volumes because we need them to be available when the system starts up. Mounting the extra volumes is no problem but once inside an application going to the desktop in an open dialog freezes the applications. Is there any documentation of this problem or this topic?

___

The reason all the disk partitions aren't mounted at boot time is an Event Manager limitation. The Event Manager can queue only 20 events. Depending on what else is going on at boot time that might post an event (or events), you'll probably get only somewhere between 15 and 20 volumes mounted.

The lockup/crash problem is caused by a Standard File limitation. Standard File (under System 7) works only with up to 20 mounted volumes. Any more than that and Standard File trashes its own local variables. We've filed a bug report against Standard File for this behavior. The Radar bug system number for this is #1070708 if you need to refer to it in the future.

As a workaround until the Standard File limitation is fixed, you can use shared folders to reduce the number of mounted volumes on an AppleShare server.

PBGetVolMountInfoSize & PBGetVolMountInfo doc fix

Date Written: 2/3/93

Last reviewed: 6/14/93

Our code crashes when we call `PBGetVolMountInfoSize` or `PBGetVolMountInfo`. Are their parameter blocks correctly defined in *Inside Macintosh: Files* ?

___

The documentation is missing a parameter for both calls; `ioNamePtr` (type `StringPtr`) should follow `ioResult` in the parameter block description. With `PBGetVolMountInfoSize`, `PBGetVolMountInfo`, `PBHGetLogInInfo`, and any other File Manager call where you specify the volume reference number as an input, `ioNamePtr` must either point to the volume name, or be set to NIL.

In *Inside Macintosh: Files* , page 2-220, and *Inside Macintosh* Volume VI, page 25-48, the parameter block passed to PBGetVolMountInfoSize should look like this:

Parameter block

```
->  ioCompletion  LongInt    A pointer to a completion routine.
<-  ioResult      OSErr      The function's result code.
->  ioNamePtr     StringPtr  A pointer to the volume's name.
->  ioVRefNum     Integer    A volume specification.
->  ioBuffer      LongInt    A pointer to storage for size.
```

In *Inside Macintosh: Files* , page 2-220, and *Inside Macintosh* Volume VI, page 25-48, the parameter block passed to PBGetVolMountInfo should look like this:

Parameter block

```
->  ioCompletion  LongInt    A pointer to a completion routine.
<-  ioResult      OSErr      The function's result code.
->  ioNamePtr     StringPtr  A pointer to the volume's name.
->  ioVRefNum     Integer    A volume specification.
->  ioBuffer      LongInt    A pointer to mounting information.
```

FSWrite and error handling for dskFulErr

Date Written: 10/9/92

Last reviewed: 6/14/93

If I call `FSWrite` and attempt to write more than space allows, what happens? Of course I get a Disk Full error, but does `FSWrite` write as much as possible before quitting, and then return the number of bytes written in the count parameter?

___

In the current implementation of the file system, writes to local volumes owned by the file system are an all-or-nothing deal. If the space for a write can't be allocated, the write call fails and no bytes are written.

However, do not depend on that, because the Macintosh file system doesn't control all volumes that might be mounted. Today, Apple ships four external file systems: CD-ROM, AppleShare, ProDOS File System (for Apple II ProDOS volumes), and PC Exchange (for MS-DOS volumes). Various third parties have written other external file systems. The way they react to error conditions may not be the same as local volumes controlled entirely by the file system.

To make your application always work correctly, you should check for errors and handle them appropriately. If you get a `dskFulErr`, you should assume that if any information was written to the file, it wasn't written correctly. You should either reset the file's EOF to its previous position (if you're appending to an existing file) or delete the file (if you had just created the file and were writing to it for the first time).

PBHSetVolParms volume attribute bits

Date Written: 7/29/92

Last reviewed: 3/1/93

I'd like to change `PBHGetVolParms` fields such as `vMVolumeGrade` and `vMAttrib` (especially `bNoBootBlks` and `bNoSysDir`). In addition, I want a way to prevent a volume from being shared. How can I set this up? Is there something I need to set up in my driver's format call?

___

A disk driver has no say in what volume attribute bits are returned by `PBHGetVolParms`. The volume attributes are determined by the file system that handles a particular volume. So, a disk driver cannot set or clear specific volume attribute bits.

The System 7 file system sets these three volume attribute bits on every local HFS volume: `bHasCatSearch`, `bHasFileIDs`, and `bHasBTreeMgr`. In addition to those three bits, the `bHasDesktopMgr` bit is set on local HFS volumes if they are either nonejectable or ejectable but bigger than 2 Mb. The volume attribute bits, `bLimitFCBs`, `bNoVNEdit`, `bAccessCntl`, `bHasOpenDeny`, `bHasShortName`, `bHasFolderLock`, `bHasPersonalAccessPrivileges`, `bHasUserGroupList`, and `bHasBlankAccessPrivileges`, are set on local shared volumes by Macintosh File Sharing or AppleShare 3.0 when the file service is on.

An example of an external file system is AppleShare. For AppleShare volumes, the AppleShare external file system sets these volume attribute bits: `bLimitFCBs`, `bLocalWList`, `bNoMiniFndr`, `bNoVNEdit`, `bNoLclSync`, `bTrshOffLine`, `bNoSwitchTo`, `bNoDeskItems`, `bNoBootBlks`, `bAccessCntl`, `bNoSysDir`, `bHasExtFSVol`, `bHasOpenDeny`, `bHasCopyFile`, `bHasMoveRename`, `bHasDesktopMgr`, `bHasShortName`, `bHasFolderLock`. AppleShare volumes that support AFP 2.1 also have these volume attribute bits set: `bHasCatSearch`, `bHasFileIDs`, `bHasBlankAccessPrivileges`.

There isn't a way to keep a volume from being shared if it's mounted when Macintosh File Sharing or AppleShare starts up. The file system doesn't give the file servers a way to tell which volumes should not be shared, so the file servers uses this method:

- If volume's signature <> $4244 (hierarchical directory volume), then the volume is not sharable.
- If volume's driver reference number = $fffb (the .SONY driver), then the volume is not sharable.
- If the volume is a volume controlled by our AFP code, then the volume is not sharable.

As you can see, your disk driver doesn't have control over any of that.

ioVAtrb bit 7 and bit 15 locked volume bits

Date Written: 8/15/90

Last reviewed: 6/14/93

I want to determine whether a disk is locked before trying to mount the volume. When I examine bit 15 of ioVAtrb using `PBGetVInfo`, as suggested on page 104 of *Inside Macintosh* Volume II, bit 15 is clear for a locked volume such as a CD-ROM, but bit 7 is set. Why is this happening?

___

The reason for your observed discrepancy is that bit 15 is set for a software lock and bit 7 is set for a hardware lock. In the case of the CD-ROM there's no software lock but only a hardware lock, so bit 7 is set and bit 15 is clear. Volumes II and IV of *Inside Macintosh* both say that "only bit 15 can be changed" and should be set if the volume is locked. The fact that you can set it with `PBSetVol` means that it's a software lock. What the documentation fails to mention is that using `PBGetVInfo` you can also check bit 7 of `ioVAtrb` to see if there's a hardware lock. The recommended procedure is to first check the hardware lock (bit 7 of `ioVAtrb`) and then check the software lock (bit 15 of ioVAtrb).

How to display mounted volumes in a dialog box

Date Written: 11/1/90

Last reviewed: 6/14/93

In a MacApp Pascal program, how can I display all mounted volumes in a dialog box including each volume's icon and name (similar to the way the Finder displays volumes)? I cannot get the resource ID of the icons.

___

Inside Macintosh IV-223 and V-469 describe how to read a volume's ICON. "The Disc Driver" discusses `csCode`=21, which you can use to make special control calls that will work for all drives to get icon and other information. The Technote "What Your Sony Drives for You," describes how to make floppy calls if necessary. The `csCode` control call should work for you.

Distinguishing between Macintosh vRefNums and WDRefNums

Date Written: 12/3/90

Last reviewed: 6/14/93

How do I find the real Macintosh `vRefNum` and real `dirID`? If I pass a working directory to `PBHGetFInfo`, will the `ioDirID` field return a real dirID? Will the `ioVRefNum` field contain the working directory I passed in or the real `vRefNum`? Also, if I pass a real `vRefNum` to `GetWDInfo`, will it do nothing and yet return no error?

———

Here's the way to tell the difference between vRefNums and WDRefNums (copied from the Macintosh Technical Note "HFS Ruminations":

A `vRefNum` is a small negative word (e.g. $FFFE).

A `WDRefNum` is a large negative word (e.g. $8033).

A `dirID` is a long word (for example, 38). The root directory of an HFS volume always has a dirID of 2.

When you pass a real `vRefNum` into `GetWDInfo`, it will look at the `ioWDIndex` field in the `WDPBPtr` and return information about the corresponding working directory on that volume (*Inside Macintosh* Volume IV, page 159). You can also specify an `ioWDProcID` and `GetWDInfo` will only index through Working Directories on the specified volume with the specified proc ID.

`PBHGetFInfo` will not return a valid `dirID` and `vRefNum` if you pass it a working directory; if you want that information you should pass the WDRefNum to `PBGetWDInfo` in `ioVRefNum` (making sure you set `ioWDIndex` to zero).

Remember that working directories are provided simply to facilitate compatibility between HFS and programs that were written under MFS. You should be converting working directories to their corresponding `VRefNum` and `dirID` pairs before using them with your PBH variant calls.

Technique for maintaining up-to-date list of Macintosh volumes

Date Written: 6/7/91

Last reviewed: 6/14/93

How can my Macintosh application maintain an up-to-date list of active volumes? Is there a way to detect if a volume is ejected?

———

No event is posted, and no other notification is available for when a volume is ejected. The best way of keeping your list of volumes up to date is to index periodically through all mounted volumes with `PBHGetVInfo` or `PBGetVInfo` and cross-check them against your program's most up-to-date list of volumes.

It's probably best to do this check also on every resume event, in case the user ejects a disk while in the Finder. The periodic check is still necessary, however, in case a Desk Accessory in your own heap ejects a disk or the user hits command-shift-1 or 2. Since you probably don't want to check every call through `WaitNextEvent`, just keep track of the last time you updated the volume list and compare it to `TickCount`.

Unlocking a Macintosh volume

Date Written: 6/14/91

Last reviewed: 11/19/91

`PBSetVInfo` failed to reset a software lock bit that had been set using `PBSetVInfo`, because the volume was, of course, locked. How do you allow a volume to be unlocked?

———

DTS discourages developers from locking and unlocking volumes through software because it's not readily apparent to users. Many users are aware only of the physical reasons that they would be prevented from modifying a disk. For example, it's easy to identify CD-ROMs and locked floppies as unchangeable.

Additionally, the Finder caches the contents of its windows, so a change in the locked status of a volume may not be reflected immediately. An unlocked volume may still show a lock icon in its window, misleading users into thinking that the particular volume is still protected. For these reasons, we recommend that volumes not be locked by software.

If you need to software lock a volume, be sure it's done only after explicitly warning the user that the disk will be locked and providing an opportunity to cancel the operation.

Locking a volume is done by calling `PBSetVInfo` with bit 15 of the `ioVAtrb` field set. However, this creates a Catch-22 in that you can't make the call to unlock the disk. To override the locked status for the volume and unlock it, you must walk the volume-control-block queue in memory and clear the lock bit for the drive, as shown in the following code:

```
USES Files, Errors;

FUNCTION SoftwareVolumeUnlock(targetVRefNum: INTEGER): OSErr;
    TYPE VCBPtr = ^VCB;
    VAR
        theVCBQHdrPtr: QHdrPtr;
        myVCBPtr: VCBPtr;
    BEGIN
        theVCBQHdrPtr := GetVCBQHdr;
        myVCBPtr := VCBPtr(theVCBQHdrPtr^.qHead);

        WHILE (myVCBPtr <> NIL) AND
                (myVCBPtr^.vcbVRefNum <> targetVRefNum) DO
            myVCBPtr := VCBPtr(myVCBPtr^.qLink);

        IF myVCBPtr^.vcbVRefNum = targetVRefNum THEN
            BEGIN
                { clear locked bit }
                myVCBPtr^.vcbAtrb := BAND(myVCBPtr^.vcbAtrb, $7FFF);

                { write change to disk }
                SoftwareVolumeUnlock := FlushVol(NIL, targetVRefNum)
            END
        ELSE SoftwareVolumeUnlock := nsvErr { volume not found }
    END;
```

The VCB queue and volume attributes are discussed under "Volume Control Blocks" in the File Manager chapter of *Inside Macintosh* Volume IV.

Booting a write-protected disk under System 7

Date Written: 9/18/91

Last reviewed: 10/15/91

How do I boot a write-protected disk under System 7? It seems that the Macintosh operating system tries to write/modify the desktop file and I get an error message telling me to write-enable the disk.

___

Every Macintosh has been equipped with the ability for users to boot from both write-protected as well as writeable disks, all the way back to the first machine introduced in 1984. This didn't change with System 7; in fact, the operating system has only an indirect impact on this capability. The only thing you can't do from a write-protected boot disk is use the Chooser because the O/S needs to be able to write back the preferences for whatever you do in the Chooser.

You might be encountering the error you described because of desktop changes: Since the desktop file was obsoleted in System 7, which now uses the Desktop Manager instead, diskettes that are mastered on a System 7 machine then inserted into another machine running 6.0.x must have a desktop file created. (This does not apply to diskettes that you're trying to boot from; rather, when you insert the disk into a machine already running 6.0.x.) When this situation is encountered, the user is presented with the now familiar dialog to the effect that the disk needs minor repairs (in fact, it needs a desktop file). If you say OK to this dialog, the desktop file will be created and all will be well. All this leads up to how (not) to treat diskettes mastered under System 7: If you drag a series of files to a floppy disk (from a System 7 machine), immediately eject and lock it, then try to boot from it, the machine will try to update the directory information on the diskette as the machine is booting up. The way around this is to either A) eject the floppy after copying your files then reinsert it (or simply open and close the main window) to allow the directory information to be updated or B) use the installer to place files on the floppy as the installer automatically updates the directory information.

Another possible explanation for the anomolous behavior you're seeing is that you've got an INIT or application of some sort (such as a virus checker perhaps) on the floppy that wants to update itself after the system has finished starting up.

Changing a volume's modification date

Date Written: 12/9/91

Last reviewed: 6/14/93

Why can't I use `PBSetVInfo` to change a volume's modification date? The change shows up in the VCB list, but not in the Finder.

___

The modification date shown for a volume in its Get Info window in the Finder is not actually the mod date as recorded in the volume's VCB. Rather, it is the mod date of the root folder. The VCB mod date as returned by `PBGetVInfo` is the last time a change was written to the volume, but it cannot easily be forged, since calling `PBSetVInfo` will set the mod date to the time of the `PBSetVInfo` call.

An application should not need to change the modification date of a volume as it appears in the disk's Get Info window. However, it can be done by calling PBSetCatInfo to set the `ioDrModDat` field of the root folder. The root folder's dir ID is the constant `fsRtDirID` = 2. `PBSetCatInfo` is documented on page 156 of *Inside Macintosh* Volume IV.

System 7 DA UnmountVol bug

Date Written: 2/26/92

Last reviewed: 6/14/93

I am trying to unmount a volume from a desk accessory using the UnmountVol trap. Unmounting network volumes works fine. However, unmounting a floppy ends up giving me a stack/heap collision in the `_Unmountvol` trap. This only occurs in System 7 and only in a DA. If I move this to an application the code works fine. On System 6 the code also works (although I am getting unexpected results in the Finder.)

___

This is a bug in System 7, which should be fixed in the next major system release. It only happens with native (local) volumes since they are the only ones that use the operating system Unmount code. Network volumes dispatch to an external file system, whose code is likely to be correct.

How do I tell if a volume is a floppy or hard disk, or removable

Date Written: 11/17/89

Last reviewed: 6/14/93

How do I tell if a Macintosh volume is a floppy or hard disk?

___

What you probably want to know is whether a device in the drive queue is removable. Do this by examining the four bytes of flags proceeding the drive queue entry for the device. See pages 181-2 of *Inside Macintosh* Volume IV for details. C source is:

```
/* we assume that you get the drive number from some appropriate
   place, such as doing a PBHGetVInfo for the volume and grabbing the
   ioVDrvInfo field (Inside Mac IV-124)
*/
Boolean
IsEjectable(driveNumber)
short driveNumber
{
    DrvQElPtr   d;
    QHdrPtr     queueHeader;
    Ptr         p;

    queueHeader = GetDrvQHdr();
    d = (DrvQElPtr)queueHeader->qHead;

    while (d != nil)    /* find the appropriate drive # */
    {
        if (d->dQDrive == driveNumber)   /* is this the drive we want? */
        {
            p = (Ptr)d;
            p -= 3; /* to get to the byte with eject info */
            if ((*p) & 8)
                return false;           /* non ejectable disk in drive */
            else
                return true;
        }
        d = (DrvQElPtr)d->qLink;
    }
    return false;   /* you specified an invalid drive number */
}
```

If you actually want to know if a volume is a floppy or not, use `PBHGetVInfo`, and multiply the `ioVNmAlBlks` (remember, it's unsigned!) times `ioVAlBlkSiz`. Currently valid sizes include 400K, 800K, 720K, 1440K.

How will you handle other sizes in the future? Nasty. That's why I hope you are really asking to detect removable media.

Where can I find more information about HFS structures?

Date Written: 5/3/89

Last reviewed: 11/21/90

Where can I find more information about HFS structures other than in *Inside Macintosh?*

___

Another source of information besides *Inside Macintosh* is the MPW `FSPrivate.a` file, which contains some private equates used when building the system. None of the information found or used in this file, however, is supported. Don't ask MacDTS questions regarding any of this information. Using the information is very risky, and will certainly lead to compatibility problems. Proceed at your own risk!

File Manager backup volume info block bug & workaround

Date Written: 3/6/92

Last reviewed: 6/14/93

I found a situation where the Macintosh file system incorrectly updates the backup volume info block. If the driver for this volume is a qType 1 and the `dQDrvSz` is zero (let's say `dQDrvSz2` is $0100), the backup volume info block won't get updated properly when a new catalog extent is created. This is because the calculation dQDrvSz-2 (to get to the backup volume info block) is done as an integer rather than a longint. This means the block addressed is `$0100FFFE` instead of `$FFFFFE` which accesses a block past the end of the volume! I caught this condition by a test in my driver code for accesses outside the drive's size. It seems to be a problem with both System 6 and System 7. Is this a bug?

___

It looks like you're right. That bug is present in the File Manager ROM code on the Macintosh Plus, SE, II, IIx, IIcx, and Classic (everything with the pre-IIci ROMs). The two workarounds are:

* Make sure the low word of the logical number of blocks (stored in `dQDrvSz`) is never $0000 or $0001. This is the best workaround because it ensures that the alternate master directory block gets updated.

or

* Make sure your driver ignores accesses past the end of the volume.

Changing the volume control block modification date

Date Written: 3/11/91

Last reviewed: 6/14/93

How can I change the modification date of my Macintoshreg. external file system driver's volume control block? I've tried different values in both the vcbLsMod field of the volume control block and the `ioVLsMod` field of the `HParamBlock` without success.

___

Some of the volume information is getting magically cached somewhere, and that in order to update it, you must first put the volume offline! I know, it sounds weird, but I've seen the source code to disk copy utilities that do this in order to change that kind of information. So try putting the volume offline before writing the volume info out to it, and then call FlushVol on the volume.

Code for identifying vRefNum and DirID of MAC System Folder

Date Written: 4/10/91

Last reviewed: 6/14/93

I need to identify the startup volume as I index through all volumes with `PBHGetVInfo`, but `GetVol` and `SysEnvirons` are returning `WDIDs`. How can I get the true `vRefNum` from these calls?

___

The code below is general purpose code to identify the `vRefNum` and `DirID` of the System Folder. It is System 7.0 friendly in that it will use FindFolder if present; otherwise, it falls back to `SysEnvirons` and converts the `wdRefNum` into a vRefNum and DirID.

```
#define BTstQ(arg, bitnbr) (arg & (1 << bitnbr))

/* FindSysFolder returns the (real) vRefNum, and the DirID of the current
   system folder. It uses the Folder Manager if present, otherwise it falls
   back to SysEnvirons. It returns zero on success, otherwise a standard
   system error. */

OSErr FindSysFolder(short *foundVRefNum, long *foundDirID)
{
    long            gesResponse;
    SysEnvRec        envRec;
    WDPBRec            myWDPB;
    unsigned char    volName[34];
    OSErr            err;


    *foundVRefNum = 0;
    *foundDirID = 0;
    if (!Gestalt (gestaltFindFolderAttr, &gesResponse) &&
        BTstQ (gesResponse, gestaltFindFolderPresent)) {    /* Does Folder
Manager exist? */
            err = FindFolder (kOnSystemDisk, kSystemFolderType,
kDontCreateFolder,
                foundVRefNum, foundDirID);
    } else {
        /* Gestalt can't give us the answer, so we resort to SysEnvirons */
        if (!(err = SysEnvirons (curSysEnvVers, &envRec))) {
            myWDPB.ioVRefNum = envRec.sysVRefNum;
            volName[0] = '\000';                    /* Zero volume name */
            myWDPB.ioNamePtr = volName;
            myWDPB.ioWDIndex = 0;
            myWDPB.ioWDProcID = 0;
            if (!(err = PBGetWDInfo (&myWDPB, 0))) {
                *foundVRefNum = myWDPB.ioWDVRefNum;
                *foundDirID = myWDPB.ioWDDirID;
            }
        }
    }
    return (err);
}
```

System 7 UnmountVol and Eject calls return positive drive number

Date Written: 4/30/91

Last reviewed: 6/14/93

Why is my Macintosh driver receiving a positive drive number under System 7.0 upon notification of an `_Eject` call?

__

When the "driver wants a call on eject" bit is set in the flag bytes preceding a drive queue element, `_Eject` will issue a `_Control` call with a csCode of 7 to the driver. This `_Control` call is supposed to inform the driver which disk the OS is attempting to eject, by passing the drive number in the `ioVRefNum` field of the parameter block.

However, there's a bug in the ROM that only mainfests itself when `_Eject` is given a volume reference number for a disk that has both the "nonejectable" and "driver wants a call on eject" bits set in the drive flag bytes. This bug causes the driver to receive the negative of the drive number, rather than the positive drive number.

The System 7.0 Finder has reversed the order of its calls to `_UnmountVol` and `_Eject`, causing it to pass the drive number to `_Eject`, which then passes it on to the driver correctly. Unfortunately, under previous systems, the Finder passed the volume reference number to `_Eject`, forcing developers to work around the bug by accepting negative drive numbers; however, a problem could occur now under System 7.0 if positive drive numbers weren't accepted as well.

A number of driver writers have notified us of this problem, but few (so far) have been adversely affected. As it has always been possible for utilities or applications to make `_Eject` calls with either a volume reference number or a drive number, the proper workaround is to handle both positive and negative drive numbers.

X-Refs:

Inside Macintosh Volume II, page 214

Inside Macintosh Volume IV, page 181

How to determine if a Macintosh file is on a locked disk

Date Written: 1/1/91

Last reviewed: 6/14/93

How do I determine whether a Macintosh file is on a locked disk?

___

Call the function `PBGetVInfo` and check `ioVAtrb`, which it passes back in the parameter block. If the volume has a software lock, then bit 15 of ioVatrb will be set. If there is a hardware lock (such as the CD-ROM), bit 7 is set.

Working directories and unmounted volumes

Date Written: 5/4/92

Last reviewed: 6/14/93

When can I unmount a Macintosh volume? I know that if any files (other than Desktop, Desktop DB or Desktop DF) are open, I can't unmount the volume, but I'm not sure about working directories. Will open working directories for that volume prevent the unmount call from working?

___

Open working directories on a volume will not keep the volume from unmounting. When a volume is unmounted under either System 6 (Finder or MultiFinder) or System 7, all working directories for that volume are made invalid. Any attempts to use the working directory number after the volume is unmounted will result in a `rfNumErr` (-51).

Getting open file name on an MFS volume

Date Written: 4/30/92

Last reviewed: 6/14/93

`PBGetFCBInfo` works great for getting the name of an open file given the path reference number, except for on an MFS volume. In this case, the call returns success but the name is not filled in. Is it because the name of the file is not kept in the FCB of an MFS volume? Should this work or is there a better way to do what I am trying to do?

___

The code that deals with MFS volumes and the files on MFS volumes is the same code that has always dealt with MFS. The MFS code that fills in the FCB only knows about the first 30 bytes of the FCB (the 30 bytes documented in *Inside Macintosh* Volume II on page 127) which does not include the file name and several other pieces of information used by only by HFS.

So, there's really no good way to get the file name of an open file on an MFS volume.

How Macintosh Finder calculates free and used volume space

Date Written: 7/1/92

Last reviewed: 6/14/93

How does the Macintosh Finder calculate free, total, and used space on a given volume? The information derived from the fields in the VCB seems to give me the correct volume free space, but the volume used space sometimes is off.

___

The calculation the Finder uses for space used in the Get Info dialog is this:

```
    free = vcbFreeBks * vcbAlBlkSiz
    totalAlBlks = vcbNmAlBlk
    if the volume is not a MFS volume
        totalAlBlks = totalAlBlks - (vcbXTAlBlks + vcbCTAlBlks)
    used = (totalAlBlks * vcbAlBlkSiz) - free;
```

The Finder doesn't consider space used by the volume's catalog file and extents overflow file to be space used by the "user."

We don't normally recommend accessing the VCB directly if at all possible because it is a compatibility risk, but there is some information you can't get any other way, such as `vcbXTAlBlks` and `vcbCTAlBlks`. That is, if you can get everything you need from the `PBHGetVInfo` call, use `PBHGetVInfo`.

Back to top

Downloadables

	Acrobat version of this Note (K)	Download

Back to top