New Technical Notes

Macintosh ®

Developer Support

NW 510 - AppleTalk Filing Protocol Q&As

Networking

Revised by: Developer Support Center September 1993
Written by: Developer Support Center October 1990

This Technical Note contains a collection of Q&As relating to a specific topic—questions you've sent the Developer Support Center (DSC) along with answers from the DSC engineers. While DSC engineers have checked the Q&A content for accuracy, the Q&A Technical Notes don't have the editing and organization of other Technical Notes. The Q&A function is to get new technical information and updates to you quickly, saving the polish for when the information migrates into reference manuals.

Q&As are now included with Technical Notes to make access to technical updates easier for you. If you have comments or suggestions about Q&A content or distribution, please let us know by sending an AppleLink to DEVFEEDBACK. Apple Partners may send technical questions about Q&A content to DEVSUPPORT for resolution.

New Q&As in this Technical Note:
AFP error codes -5060,-5061, -5062, and -5063

AFP error codes -5060,-5061, -5062, and -5063

Date Written: 1/18/93
Last reviewed: 6/14/93

What's AppleTalk error -5062? It seems to be an AppleTalk Filing Protocol error of sorts but I can't find it documented anywhere.

—

Error -5062 is an afpAlreadyMounted error. You'll get it from the AppleShare external file system if you try to mount an AppleShare volume that's already mounted with PBVolumeMount. The three PBVolumeMount-related functions were added to System 7 very late during development, so afpAlreadyMounted (-5062), afpBadDirIDType (-5060), afpCantMountMoreSrvrs (-5061), and afpSameNodeEr (-5063) never made it into the public interface files.

Further Reference:
Inside Macintosh Volume VI, pages 25-49 and 25-50

How an AFP volume's allocation block size is calculated

Date Written: 8/15/91

Last reviewed: 8/16/91

The AppleTalk Filing Protocol (AFP) doesn't appear provide a way to learn about the allocation block size of a server volume. So, how is the allocation block size determined on AFP server volumes? I noticed that the Macintosh Finder seems to calculate "size on disk" erratically for files on some AFP server volumes. Is that a related problem?

—

All the workstation can get from an AFP server is the Bytes Total and Bytes Free (returned by FPGetVolParms and FPOpenVol), so the AFP workstation code estimates the value it uses in the allocation block size field (vcbAlBlkSiz) of the volume control block (VCB). The workstation code comes up with a value for vcbAlBlkSiz from the Bytes Total value returned to FPOpenVol when the volume is opened. The calculation used is:

```
vcbAlBlkSiz := round((BytesTotal / (512 * 65536)) + 0.5) * 512;
```

This is the same value a local HFS volume would use. If an AFP server is not running on a Macintosh (and then, probably not using HFS), then this value may not be what the server platform is actually using. And yes, the Finder uses the allocation block size in its calculations to decide how much space is used by a file, so the "size on disk" may not exactly reflect the amount of disk space actually used.

Inside Macintosh Volume VI PBGetVolMountInfoSize typo

Date Written: 1/22/92
Last reviewed: 6/14/93

When I call PBGetVolMountInfoSize to get the size of an AppleShare volume's mounting information record, the function returns with no errors but ioBuffer points to garbage instead of the size of an AFPVolMountInfo record.

—

The problem you're having with PBGetVolMountInfoSize is a typo in *Inside Macintosh* Volume VI on page 25-48. In that call, the ioBuffer field should be a pointer to a word (2-byte) size variable, not a long (4-byte) size variable.

AFP 2.0 FPRead NewLine Mask change

Date Written: 5/3/89
Last reviewed: 11/21/90

I'm confused about the changes to FPRead in AFP (AppleTalk Filing Protocol) version 2.0. How do I use the NewLine Mask?

—

The difference between AFP 1.1 and AFP 2.0 as far as the NewLine Mask is concerned is that, in AFP 1.1, the only legal values of NewLine Mask are $00 and $FF, whereas in AFP 2.0, all values of NewLine Mask are allowed. The NewLine Mask is logically ANDed with a copy of each byte read. If the result matches the NewLine character, the read terminates. The NewLine character is returned as the last byte of data that was read from the fork.