AppleTalk Filing Protocol Specification

Version 2.1

Apple Computer, Inc.
February 22, 1991

A few extensions to AFP 2.0 (the version of AFP currently used in AppleShare 2.0.1) have been suggested to support extra features in AFP servers as well as new calls that have been added to the Hierarchical File System for System 7.0. These protocol extensions will be called AFP 2.1. The AFPVersion string for AFP 2.1 is `'AFPVersion 2.1'`.

The new calls that have been added to the protocol are:

- afpGetSrvrMsg enables an AFP client to get a string message from the server. Note: this call is optional. It does not have to be supported for the server to be called AFP 2.1 compliant. This document also defines the previously undocumented AFPUserBytes, the 2-byte attention code sent in an ASP Attention packet to an AFP client.

- Four calls to support File IDs. File IDs provide a mechanism for applications (and users) to keep track of a file regardless of whether it has been moved or if its name has been changed. These four calls are afpCreateID, afpDeleteID, afpResolveID, and afpExchangeFiles. All these calls are optional (see afpOpenVol and afpGetVolParms).

- Support for afpCatSearch, which allows searching of the catalog on almost any field that is returned by PBGetCatInfo. This call is optional (see afpOpenVol and afpGetVolParms).

Other changes in the behavior of the server to support enhanced security features are contained in AFP 2.1. These enhancements are optional in AFP 2.1, but support for them greatly improves the user experience when these new security features are utilized.

In order to accommodate some new features in AFP and HFS, the bitmaps of certain calls have been augmented. They are:

- New Directory Attributes and Access Rights in afpGetFlDrParms and any call that uses this bitmap.
- New bit definitions in the Flags word returned by afpGetSInfo.
- New Volume Attributes in afpGetVolParms.

A new User Authentication Method is available for use with AFP 2.1, known as Two-Way Scrambled. Using this method, the user is not only authenticated to the server, but the server is authenticated to the user.

In order to accommodate an environment on a local machine in which some portions of the hierarchical file system are shared (or "exported") for regular users, while at the same time the entire hierarchy is available for the local user (and the owner when connected remotely), the notion of "Blank Access Privileges" was added. A folder with blank access privileges "inherits" the privileges of the folder in which it is contained.

Furthermore, when a folder is created remotely, the default access privileges assigned to that folder are different than under AFP 2.0. Now when a user creates a new folder, the owner is still assigned full privileges, but the enclosing folder's Group and World privileges are copied to it.

Lastly, both "User" and "Group" names are now valid in either of the current "Owner" or "Group" fields. This allows two new situations that were not allowed in AFP 2.0:

1) a folder can now be owned by more than one user, and
2) a folder can be shared specifically with one user differently than with everyone else.

Blank Access Privileges

AFP 2.1 supports blank access privileges. When a folder has its blank access privilege bit set, then the other access privilege bits are ignored, and it uses the access privilege bits of its parent.

This paradigm is useful because folders' access privileges now behave in a way which users expect them to; when a folder with blank access privileges is moved around within a folder hierarchy, it always reflects the access privileges of its containing folder. However, once the blank access privileges bit has been cleared for a folder, its access privileges "stick" to that folder, and remain unchanged no matter where the folder is moved.

Therefore, although blank access privileges are an optional feature of AFP 2.1, it is highly recommended that this feature be included in your particular AFP 2.1 implementation since it has subtle human interface repercussions.

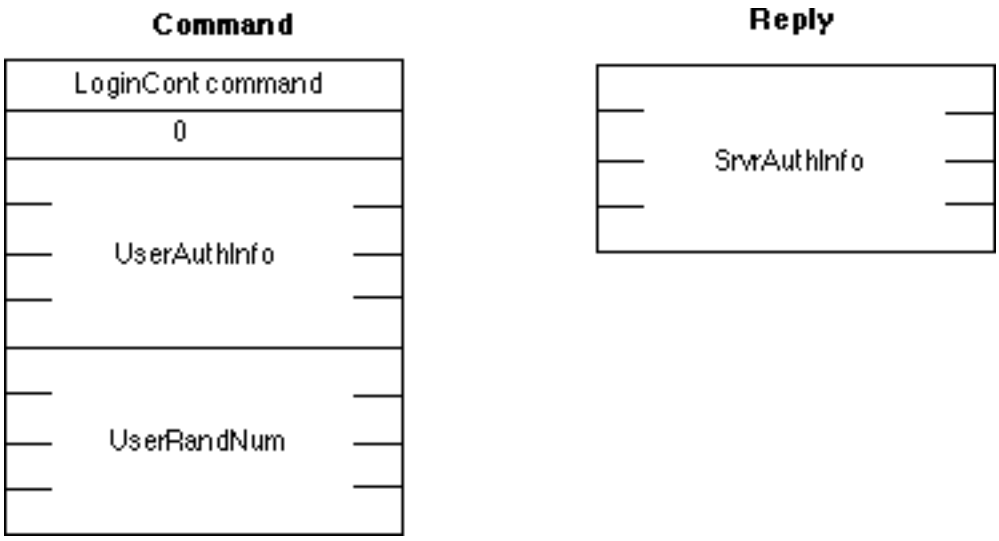
Two-Way Scrambled User Authentication Method

AFP 2.1 supports a new User Authentication Method in which the workstation is not only authenticated by the server, but the server is also authenticated by the workstation. This method uses the same initial steps as the Random Number Exchange UAM, with one additional last step. The corresponding UAM string is `'2-Way Randnum exchange'`.

Both the Random Number Exchange and the Two-Way Scrambled UAMs start with the workstation asking to login to the server. If the login is allowed, the server returns a random double-long word and an error of afpAuthContinue. The workstation then encodes the double-long word with its password and sends it back to the server in an afpLoginCont call. If the encoding was performed correctly, the workstation is authenticated and noErr is returned. However, for the Two-Way Scrambled method, the

workstation also sends a random double-long word along with its afpLoginCont call, which the server encodes with what it believes is the user's password, and returns that resulting double-long word in the afpLoginCont reply. The workstation compares this response to what resulted from its encoding of the original double-long word, and if they are the same, the server is then also authenticated. This avoids the potential for trojan-horse file servers.

Below are the command and reply block formats for the afpLoginCont call when using the Two-Way Scrambled user authentication method.



The Two-Way Scrambled UAM is not available for use with the afpPwdChange call, nor is it required. If the client is concerned about authenticating the server, they will have already logged in to the server with the Two-Way Scrambled UAM. Since the client must already be authenticated to call afpPwdChange, the client is assured that the server is the one they expect.

New Bitmap Definitions

Directory Attributes and Access Rights in afpGetFlDrParms

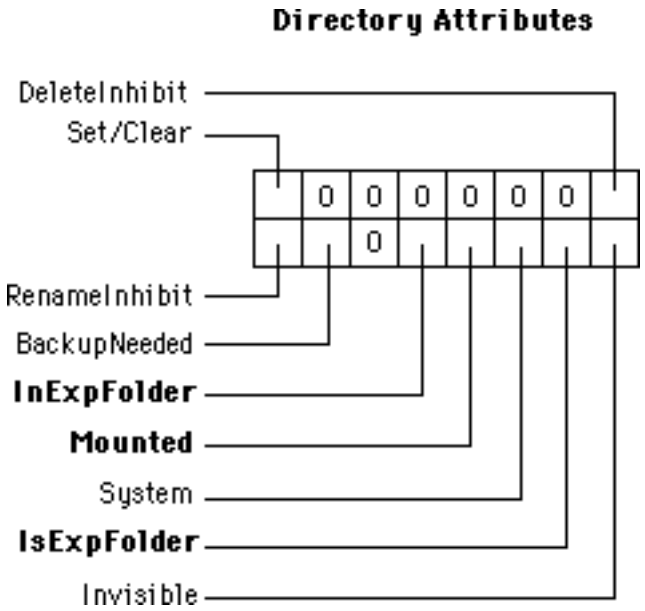
In order to accommodate the ability to export folders (as opposed to only entire volumes in AppleShare 2.0.1), new bit definitions have been added to the Directory Attributes word for afpGetFlDrParms. The entire directory bitmap is reproduced below, with the new bits for AFP 2.1 in bold. The new bits are:

IsExpFolder (bit 1)	This folder is a share point. This folder, and all folders under it, should give feedback to the local user that access privileges are valid (i.e., tabbed folders, drop box folder icons, enable the "Get Privileges" (System 6.0) or "Sharing..." (System 7.0) menu items). All folders outside of the shared area do not show access privileges on the local machine (although they may still possess

valid access privilege information, which only a super-user can see or modify).

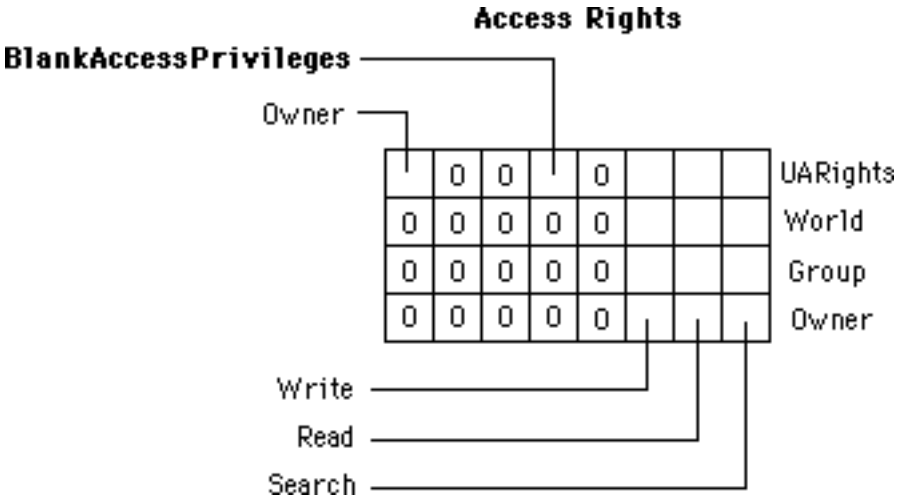
Mounted (bit 3) — This share point is mounted by some regular user (i.e., a user without "All Privileges"). The icon for such a folder will give feedback to the user of the local machine that this folder is not only a share point, but that a remote user currently has it mounted.

InExpFolder (bit 4) — This folder is in a shared (exported) area of the folder hierarchy. This folder, and all folders under it, should give feedback to the local user that access privileges are valid. In addition, it is not allowed for this folder to be shared, since it is not allowed to have a share point within another share point.



In order to accommodate blank access privileges, a new bit definition has been added to the Access Rights longword for afpGetFlDrParms. The entire access rights longword is reproduced below, with the new bit for AFP 2.1 in bold. The new bit is:

BlankAccessPrivileges (bit 28)
This folder has Blank access Privileges and will mirror the access privileges of its enclosing folder.


Access Rights
BlankAccessPrivileges
Owner
0 0 0 0
UARights
0 0 0 0 0
World
0 0 0 0 0
Group
0 0 0 0 0
Owner
Write
Read
Search

Flags word in afpGetSInfo

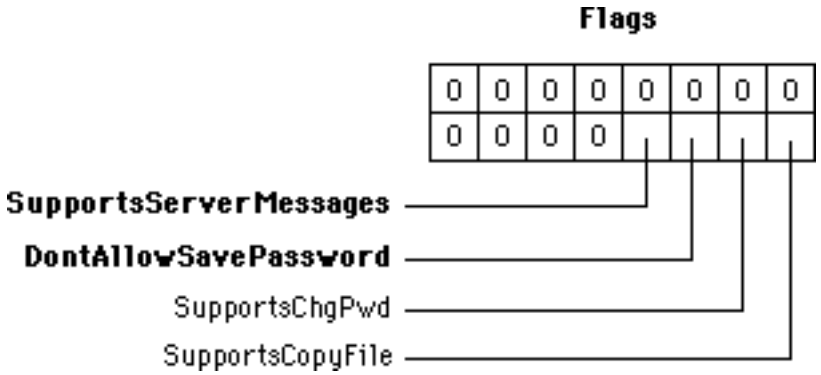
In order to accommodate the (optional) new features of AFP 2.1, some new bit definitions have been added to the Flags word for afpGetSInfo. The entire flags word is reproduced below, with the new bits for AFP 2.1 in bold. The new bits are:

DontAllowSavePassword (bit 2)

The workstation should not allow the user to save their password for boot mounting purposes. The item selection dialog may still allow the user to save their name, but when this bit is set, the button to allow the user to save their name and password will not be displayed.

SupportsServerMessages (bit 3)

Since server messages are an option in AFP 2.1, this allows servers to specify whether or not this optional feature is supported.



Volume Attributes in afpGetVolParms

In order to accommodate the new HFS calls in System 7.0, some new bit definitions have been added to the volume attributes word for afpGetVolParms. The entire attributes word is reproduced below, with the new bits for AFP 2.1 in bold. The new bits are:

HasVolumePasswords (bit 1)

This volume supports volume passwords. Volume passwords were supported in prior versions of AFP; now the volume attributes reflect this information.

SupportsFileIDs (bit 2)

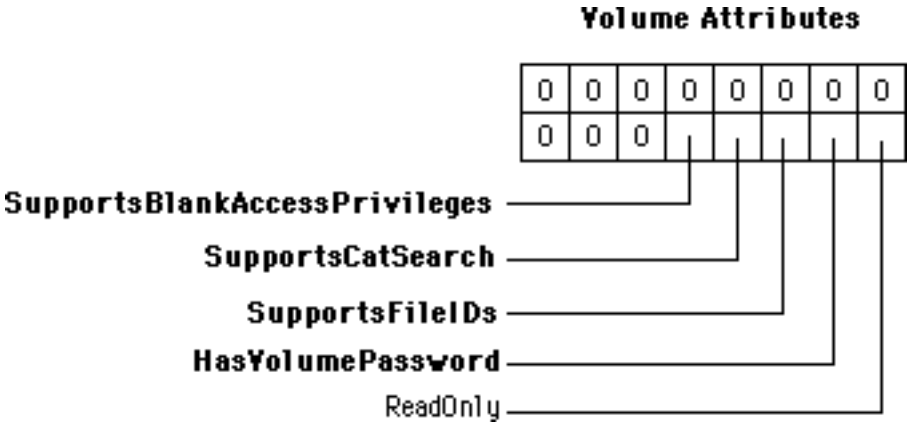
This volume supports File IDs. In general, if File IDs are supported on one volume, they will be supported on all volumes, but this allows the server to be more selective if necessary.

SupportsCatSearch (bit 3)

This volume supports afpCatSearch calls. Since afpCatSearch is optional in AFP 2.1, this allows the server to make this capability available on a per-volume basis.

SupportsBlankAccessPrivileges (bit 4)

This volume supports blank (mirrored) access privileges. Blank access privileges are discussed above.



New Security Features

AFP 2.1 has been augmented with a number of new security features. They are known as Minimum Password Length, Password Expiration, and Maximum Failed Login Attempts. These features are described below:

Minimum Password Length

It is now possible to specify the minimum length for a user's password. This length is specified using some administrative program. If the user's password is too short, they will get an afpPwdTooShort error upon logging in. The client code should put up an explanatory dialog, and then allow the user to change their password. The afpPwdChange call will continue to fail with an afpPwdTooShort error until a password of at least the specified length is submitted.

The administrative program should be intelligent enough to not allow the administrator to give passwords to users which are too short, otherwise these users' first login attempt will be dissatisfying, if not confusing. Whether or not the administrative program should alert the administrator when passwords for existing users are too short (as might happen when the administrator changes the minimum password length from 4 to 8) is up to the developer of such administrative programs.

The maximum password length is still 8.

Password Expiration

It is now possible to specify the period of time during which a user must change their password. This time can be specified using some administrative program. If the user changes their password before the password expiration time expires, the password expiration timer will be reset. However, if they do not change their password before the

password expiration time expires, their afpLoginCont call will return afpPwdExpired error. At this point they are logged in (i.e., their session is open), but they are not yet authenticated, and the only command they are allowed to perform is afpPwdChange. Once they successfully change their password, they are authenticated and can proceed as normal. Note that any other calls they make before they change their password will result in an afpUserNotAuth error.

If the administrator would like to give a user an account which becomes inactive after a certain period of time, they can set the password expiration time to that period of time, and then disallow the password to be changed. When the time expires, the user will no longer be able to connect to the server.

In order to keep the user from circumventing the spirit of this feature, there is a new error returned by the afpPwdChange call, afpPwdSameErr, which disallows the user from performing a change password when their new password is the same as their old password. The afpPwdChange call will return afpPwdSameErr only if the password expiration feature is enabled.

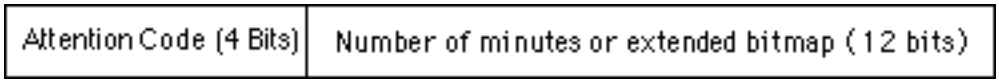
Maximum Failed Login Attempts

It is now possible to specify a maximum number of consecutive failed login attempts before the user's account is disabled. This count can be specified by some administrative program. The count is reset to zero after every successful login. Then, for every failed login attempt without an interceding successful login, the count is incremented. When the maximum failed login attempt is reached, the user's login is disabled. Any attempts to login after the user's login is disabled will result in an afpParmErr indicating that the user is unknown. The administrator will need to be notified to enable the user's account again. In this way, the administrator is alerted to potential intrusion attempts.

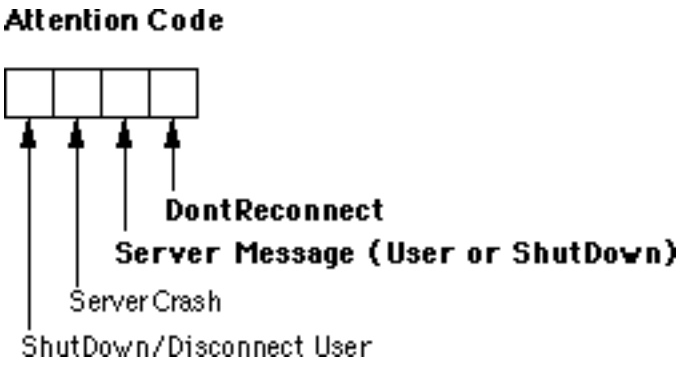
New AFPUserBytes definitions

The AFPUserBytes are the 2-byte attention code sent in an ASP Attention packet to the AFP client. In order to accommodate some new features in AFP 2.1 (such as server message) and new capabilities in the workstation code (auto-reconnect), the AFPUserBytes were augmented as described below.

The AFPUserBytes are defined as follows:



The Attention Code bits for the AFPUserBytes are defined as follows with the new bit definitions for AFP 2.1 in bold:



The bit numbers for the Attention Code bits are defined as follows:

bit 15	Shutdown or Attention bit. Used when either the server is being shutdown or when a user(s) are being disconnected.
bit 14	Server Crash bit. The server has detected some internal error, and the session will close immediately with minimal flushing of files. There may be some data loss. This is never accompanied by a server message. This condition is high unlikely.
bit 13	Server Message bit. There is a server message that the client should request using the afpGetSrvrMsg call with a MsgType of "Server" (see below). the client should request the message as soon as possible after receiving this attention, or else the server message it receives could be outdated.
bit 12	Don't Reconnect bit. This bit is set when a user is disconnected so that the client's reconnect code does not attempt to reconnect the session. This bit is not set for normal server shutdowns, nor when the server crashes. And it obviously is not set when the server loses power or where there is a break in the network cabling, because in these cases this attention is never sent out. This mechanism allows administrators to disconnect users (and not allow them to reconnect), but also allows them to shutdown the server for backup purposes, bring it back up again, and allow all those clients to transparently reconnect. This bit is ignored except for when the number of minutes is 0.

The valid bit combinations are the following:

1000	The server is shutting down or the user will be disconnected in the designated number of minutes. There is no message accompanying this shutdown. The workstation may reconnect if desired. This code may be used upon server shutdown (i.e., quitting file service).
1001	The server is shutting down or the user will be disconnected in the designated number of minutes. There is no message accompanying this shutdown. The workstation ought not reconnect. This is the code used upon user disconnection (i.e., selecting an intruder and disconnecting them).
1010	The server is shutting down or the user will be disconnected in the designated number of minutes. There is a message accompanying

	this shutdown. The workstation should immediately submit an afpGetSrvrMsg call to receive and display the message. The workstation may reconnect if desired. This code may be used upon server shutdown (i.e., quitting file service).
1011	The server is shutting down or the user will be disconnected in the designated number of minutes. There is a message accompanying this shutdown. The workstation should immediately submit an afpGetSrvrMsg call to receive and display the message. The workstation ought not reconnect. This is the code used upon user disconnection (i.e., selecting an intruder and disconnecting them).
0100	The server is going down immediately (possibly due to some internal error) and can only perform minimal flushing. Number of minutes is ignored. There is never a message accompanying such an attention code.
0010	The server has a server message available for this workstation. The workstation should immediately submit an afpGetSrvrMsg call to receive and display the message. The extended bitmap is reserved for Apple internal use only.
0011	Reserved. The extended bitmap is reserved for Apple internal use only.
0001	Reserved. The extended bitmap is reserved for Apple internal use only.
0000	Reserved. The extended bitmap is reserved for Apple internal use only.

afpGetSrvrMsg (38 or $26)

afpGetSrvrMsg enables an AFP client to get a string message from the server. This call is made by the client to receive shutdown, user, and login messages from the server. Normally, the server will send an attention to the client when these messages are available. However, the client may make the afpGetSrvrMsg call at any time. A nil string is returned if no message is available.

The afpGetSrvrMsg parameters are:

Inputs	*MsgType (int)*	Type of server message: 0 = login 1 = server
	MsgBitmap (int)	Bitmap indicating what information to pass with the server message (currently this is only the message string itself) See below for bitmap structure.
Outputs	*MsgType (int)*	Type of server message: 0 = login 1 = server
	MsgBitmap (int)	Bitmap indicating what information was passed.
	SrvrMessage (str)	String message from the server.
	FPError (long)	
Errors	*afpCallNotSupported*	afpGetSrvrMsg not implemented by Server or AFP version before 2.1.
	afpUserNotAuth	The user was not logged in.
	afpBitmapError	The bitmap specified has unrecognized bits set.
Rights	The client must be logged onto the server to receive server message notification and to issue this request. Other than that, there are no special access rights required to issue this call.	

The login (0) MsgType is only used for one kind of message and that is:

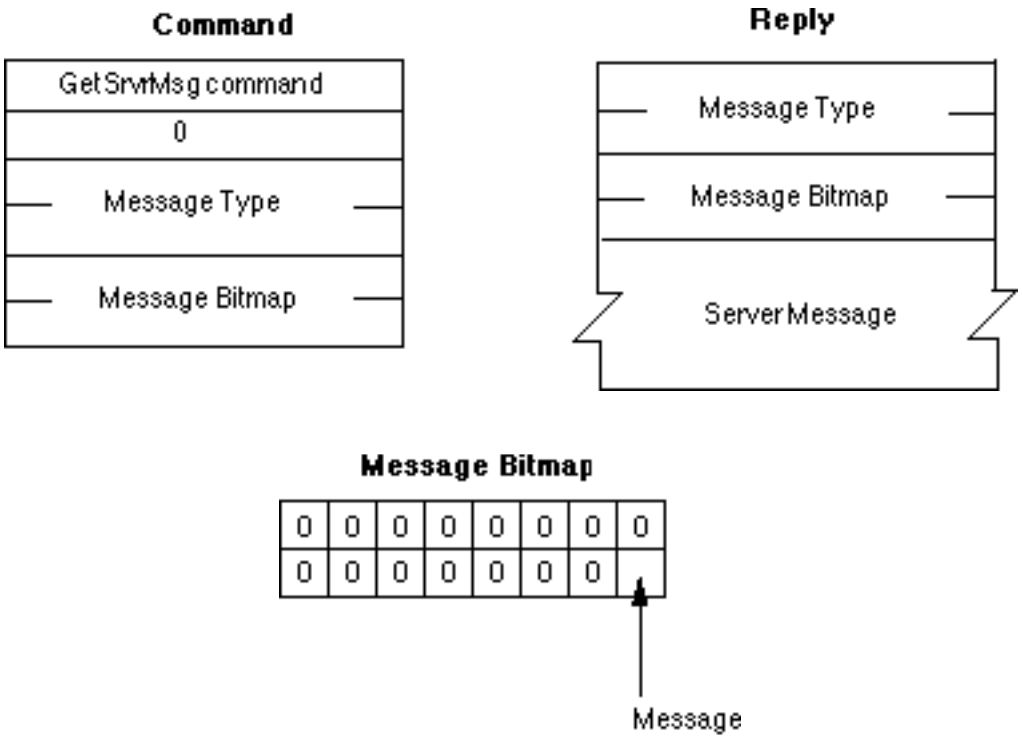
Login:	This allows for a message to be sent to a client at login time. The workstation may query the server for a login message at login time. If the workstation opts not to ask for the login message, it need not request it. If there is no login message, the afpGetSrvrMsg will simply return a zero-length string, and nothing need be displayed. By allowing the workstation to query for the login message at any time, the message may be displayed whenever it is convenient for the workstation to do so.

The server (1) MsgType is used for two kinds of messages and they are:

Shutdown: In addition to sending an attention when the server is going to shutdown, a message can be sent explaining, for example, why the server is going down, how long it will be down, etc. The workstation is made aware that a shutdown message is available by the server setting the "Server Message" bit in the Attention Code along with the "Shutdown" bit.

User: allows for a message to be sent to a specified user or users. The workstation is made aware that a user message is available by setting the "Server Message" bit in the Attention Code. Workstations implementing older AFP versions simply ignore this bit.

The maximum size of any of these messages is 200 bytes including the length byte (an Str199). The attention mechanism currently being used has been augmented to let the workstation know that there is a server message. The client then requests (via afpGetSrvrMsg) the message from the server.

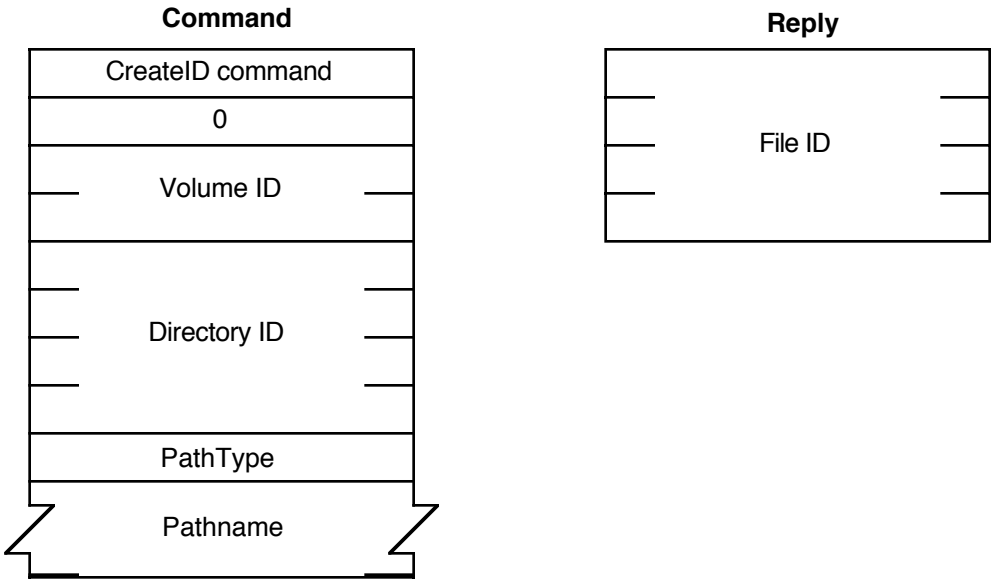


afpCreateID (39 or $27)

As stated previously, file IDs provide a means to keep track of a file even if its name or location changes. This call creates a unique file ID for a specified file. (Note: the scope of file IDs is limited to the files on a volume. They cannot be used across volumes.)

The afpCreateID parameters are:

Inputs	*VolumeID (int)*	The ID of the volume on which the file ID is to be created.
	DirectoryID (long)	The ID of the directory in which the file ID is to be created.
	PathType (byte)	Path type of the pathname: 1 = short name 2 = long name
	PathName (str)	String name of the file that is the target of the file ID (i.e., the filename of the file for which you want to create the file ID).
Outputs	*FileID (long)*	File ID that was created for the specified file.
	FPError (long)	
Errors	*afpCallNotSupported*	AFP version before 2.1.
	afpObjectNotFound	The target file does not exist.
	afpIDExists	A file ID already exists for this file. The FileID is returned in the FileID field.
	afpObjectTypeErr	Object defined was a directory not a file
	afpAccessDenied	User does not have the rights listed below or the volume has not been made available to the workstation with an afpOpenVol call.
Rights	The caller must have See Files rights to issue this call.	
Notes	The user must have previously called afpOpenVol for this volume.	


Command
CreateID command
0
Volume ID
Directory ID
PathType
Pathname
Reply
File ID

afpDeleteID (40 or $28)

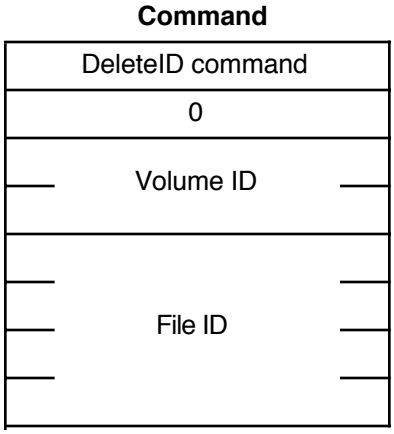
afpDeleteID invalidates all instances of the specified file ID.

The afpDeleteID parameters are:

Inputs	*VolumeID (int)*	The ID of the volume on which the file ID is to be invalidated.
	FileID (long)	File ID which is to be invalidated.
Outputs	*FPError (long)*	
Errors	*afpCallNotSupported*	AFP version before 2.1.
	afpObjectNotFound	The target file does not exist (file ID is deleted anyway).
	afpIDNotFound	File ID was not found.
	afpObjectTypeErr	Object defined was a directory not a file.
	afpAccessDenied	User does not have the rights listed below or the volume has not been made available to the workstation with an afpOpenVol call.

Rights The caller must have See Files and Make Changes rights to issue this call.

Notes The user must have previously called afpOpenVol for this volume.

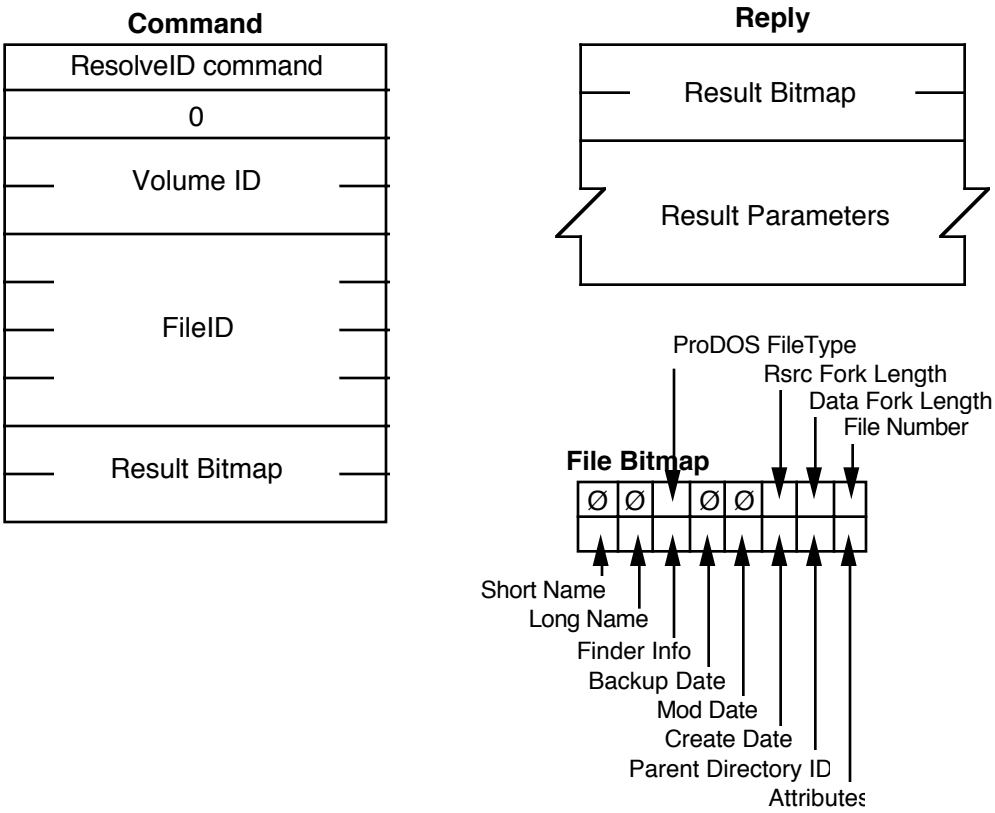


afpResolveID (41 or $29)

afpResolveID returns parameters for the file referred to by the specified file ID. These parameters can be any of those specified in the afpGetFlDrParms call.

The afpResolveID parameters are :

Inputs	*VolumeID (int)*	The ID of the volume on which the file ID is located.
	FileID (long)	File ID which is to be resolved.
	ResultBitmap (int)	Bitmap describing which parameters are to be returned (see below for bitmap structure)
Outputs	*ResultBitmap (int)*	Copy of input parameter.
	Parameters requested	
	FPError (long)	
Errors	*afpCallNotSupported*	AFP version before 2.1.
	afpIDNotFound	File ID was not found.
	afpObjectTypeErr	Object defined was a directory, not a file.
	afpBadIDErr	FileID number is not a defined FileID.
	afpAccessDenied	User does not have the rights listed below or the volume has not been made available to the workstation with an afpOpenVol call.
Rights	The caller must have See Files rights to issue this call.	
Notes	The user must have previously called afpOpenVol for this volume.	


Command
ResolveID command
0
Volume ID
FileID
Result Bitmap
Reply
Result Bitmap
Result Parameters
ProDOS FileType
Rsrc Fork Length
Data Fork Length
File Number
File Bitmap
Short Name
Long Name
Finder Info
Backup Date
Mod Date
Create Date
Parent Directory ID
Attributes

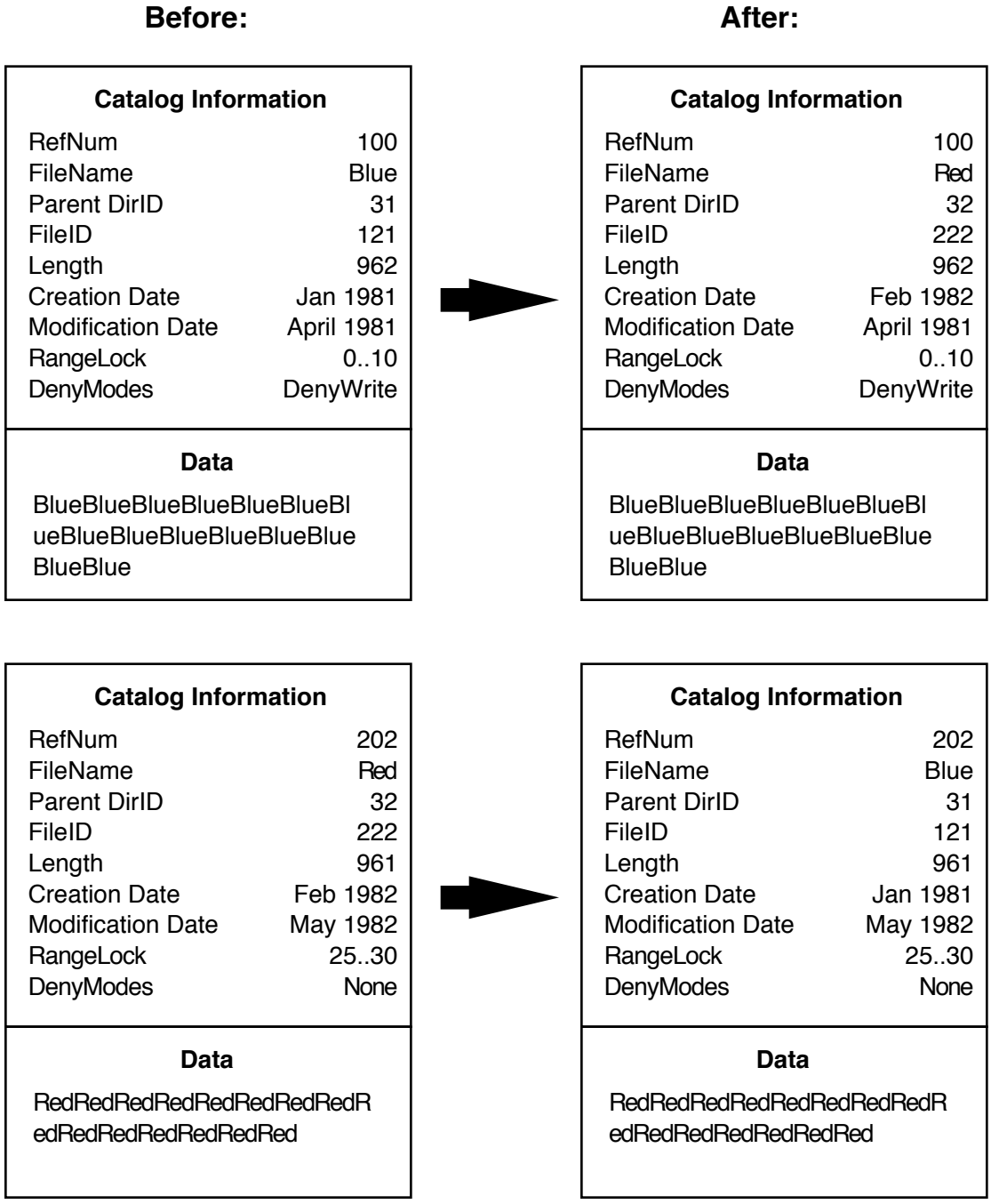
afpExchangeFiles (42 or $2A)

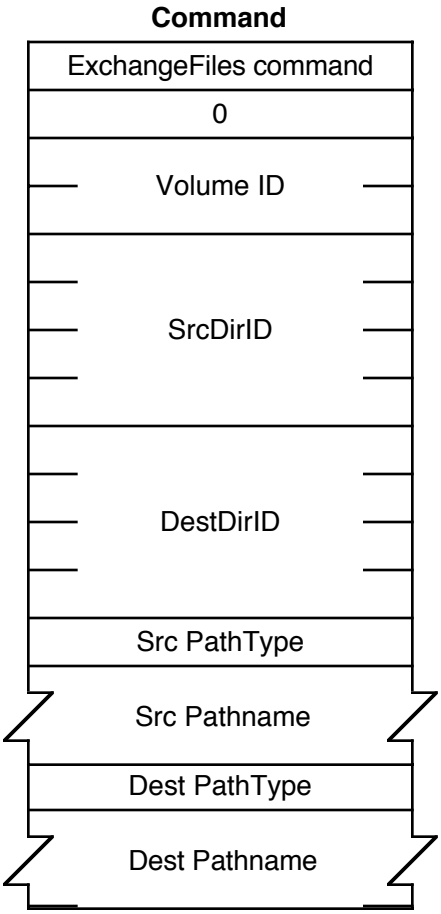
afpExchangeFiles is used to preserve existing file IDs when an application wishes to perform the "Save As..." or "Save" functions. Both files to be changed are specified. They must exist on the same volume. File IDs do not have to exist on the files to be exchanged. The files can be either opened or closed.

The afpExchangeFiles parameters are :

Inputs	*VolumeID (int)*	The ID of the volume on which the two files are located.
	SrcDirID (long)	The ID of the directory that contains the source file.
	DestDirID (long)	The ID of the directory that contains the destination file.
	SrcPathType (byte)	Path type of the source pathname: 1 = short name 2 = long name
	SrcPathName (str)	String name of the source file.
	DestPathType (byte)	Path type of the destination pathname: 1 = short name 2 = long name
	DestPathName (str)	String name of the destination file.
Outputs	*FPError (long)*	
Errors	*afpCallNotSupported*	AFP version before 2.1.
	afpObjectNotFound	The target file does not exist.
	afpDiffVol	The files that are specified exist on different volumes.
	afpObjectTypeErr	Object defined was a directory not a file.
	afpObjectLocked	The file was locked.
	afpSameObjectErr	The source file is the same as the destination file.
	afpAccessDenied	User does not have the rights listed below or the volume has not been made available to the workstation with an afpOpenVol call.
Rights	The caller must have See Files and Make Changes rights to both files to issue this call.	
Notes	The user must have previously called afpOpenVol for this volume.	

The example below shows the results of an afpExchangeFiles between the two files "Blue" and "Red". Notice that only the file name, parent directory ID, and FileID, and creation dates are exchanged. Byte range locks and deny modes still apply to the same file reference number and data.


Before:
After:
Catalog Information
RefNum 100
FileName Blue
Parent DirID 31
FileID 121
Length 962
Creation Date Jan 1981
Modification Date April 1981
RangeLock 0..10
DenyModes DenyWrite
Data
BlueBlueBlueBlueBlueBlueBl
ueBlueBlueBlueBlueBlueBlue
BlueBlue
Catalog Information
RefNum 100
FileName Red
Parent DirID 32
FileID 222
Length 962
Creation Date Feb 1982
Modification Date April 1981
RangeLock 0..10
DenyModes DenyWrite
Data
BlueBlueBlueBlueBlueBlueBl
ueBlueBlueBlueBlueBlueBlue
BlueBlue
Catalog Information
RefNum 202
FileName Red
Parent DirID 32
FileID 222
Length 961
Creation Date Feb 1982
Modification Date May 1982
RangeLock 25..30
DenyModes None
Data
RedRedRedRedRedRedRedRedR
edRedRedRedRedRedRed
Catalog Information
RefNum 202
FileName Blue
Parent DirID 31
FileID 121
Length 961
Creation Date Jan 1981
Modification Date May 1982
RangeLock 25..30
DenyModes None
Data
RedRedRedRedRedRedRedRedR
edRedRedRedRedRedRed


Command
ExchangeFiles command
0
Volume ID
SrcDirID
DestDirID
Src PathType
Src Pathname
Dest PathType
Dest Pathname

afpCatSearch (43 or $2B)

afpCatSearch allows applications to "efficiently" search an entire volume for files which match a specified criteria. This criteria includes any fields in the file and/or directory bitmaps that are defined for the afpGetFlDrParms call. Information parameters for the matching files and directories is returned. These parameters can also be any of those specified in the afpGetFlDrParms call.

The afpCatSearch call's parameters are:

Inputs	*VolumeID (int)*	The ID of the volume on which the file ID is located.
	ReqMatches (long)	The maximum number of matches to return.
	Reserved (long)	Reserved (must be zero).
	CatPosition (16 bytes)	Current position in the catalog
	FileRsltBitmap (int)	The fields in the File parameters that are to be returned; this field is the same as File Bitmap in the afpGetFlDrParms call (with some restrictions explained below).
	DirRsltBitmap (int)	The fields in the Dir parameters that are to be returned; this field is the same as Directory Bitmap in the afpGetFlDrParms call (with some restrictions explained below).
	RequestBitmap (long)	The fields in the File/Dir parameters that are to be searched on (bitmap below).
	Specification1	Search criteria lower bounds and values.
	Specification2	Search criteria upper bounds and masks.
Outputs	*CatPosition (16 bytes)*	Current position in the catalog.
	FileRsltBitmap (int)	Copy of the input bitmap.
	DirRsltBitmap (int)	Copy of the input bitmap.
	ActualCount (long)	How many matches were actually found.
	Results	An array of records that describe the matches that were found.
	FPError (long)	
Errors	*afpCallNotSupported*	AFP version before 2.1
	afpCatalogChanged	The catalog has changed and CatPosition may be invalid.
	afpParmErr	Input parameters are not valid.
	afpEofError	No more matches.
	afpAccessDenied	The volume has not been made available to the workstation with an afpOpenVol call.
Rights	No special access rights are needed to issue this call, however, to see all the files and/or folders that match the specified criteria, the caller must have See Files/See Folders rights to them. Folders without See Files/See Folders rights are skipped by the search.	
Notes	The user must have previously called afpOpenVol for this volume.	

CatPosition is a 16 byte field in which the first word signifies whether it denotes a "real" catalog position or hint. If the first word is zero, afpCatSearch should start from the

beginning. If non-zero, CatPosition is a “real” catalog position and afpCatSearch will pick up from this entry.

Specification1 and Specification2 are used together to specify the search parameters. These parameters are packed in the same order that the bits are set in the request bitmap. All variable length parameters (name, for example) are put at the end of each specification record. An offset is stored in the parameters to indicate where the actual variable length parameter is located. This offset is measured from the start of the specification parameters (not including the length and filler bytes). Results are packed in the same way.

The fields in Specification1 and Specification2 have different uses:

- In the name field, Specification1 holds the the target string and Specification2 must always have a nil name field.
- In all date and length fields, Specification1 holds the lowest value in the target range and Specification2 holds the highest value in the target range.
- In file attributes and Finder Info fields, Specification1 holds the target value, and Specification2 holds the bitwise mask that specifies which bits in that field in Specification1 are relevant to the current search.

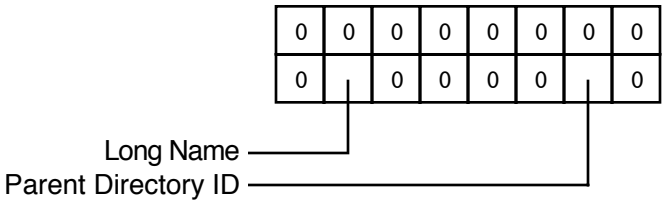
afpCatSearch returns the error afpEofError only when it has reached the end of the volume directory tree. For example, if the workstation requests 10 matches, the server may return only 4 matches with no error. The workstation should then make a request for 6 (10 - 4) more matches using the same CatPosition that was received in the previous reply. This process continues until the original requested matches are received or an afpEofError is returned.

afpCatSearch will return files and or directories depending on the FileRsltBitmap and DirRsltBitmap fields. If the FileRsltBitmap field is zero, afpCatSearch will assume that you are not searching for files. Likewise, if the DirRsltBitmap field is zero, afpCatSearch will assume that you are not searching for directories. If both fields are non-zero, afpCatSearch will return both files and directories. Note that if you are searching for both files and directories, certain restrictions apply on what fields afpCatSearch will search on (see below).

Valid Bitmaps For afpCatSearch:

The only valid bits for the FileRsltBitmap and DirRsltBitmap fields are the *LongName* and *Parent Directory ID* bits.

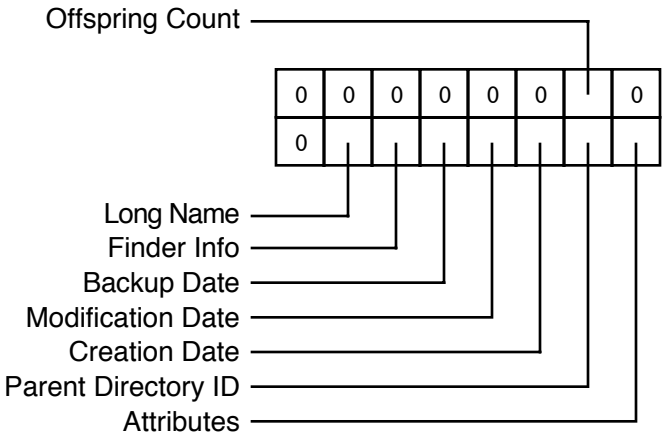
Valid ResultBitmap Bits



RequestBitmap:

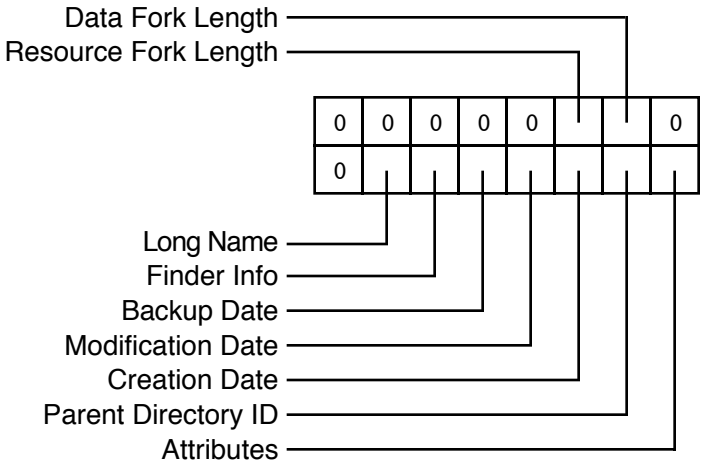
afpCatSearch can search for the following information when searching for directories only.

Valid Directory Bits



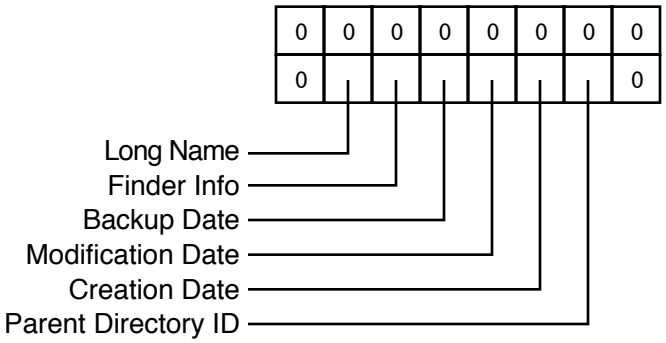
afpCatSearch can search for the following information when searching for files only.

Valid File Bits



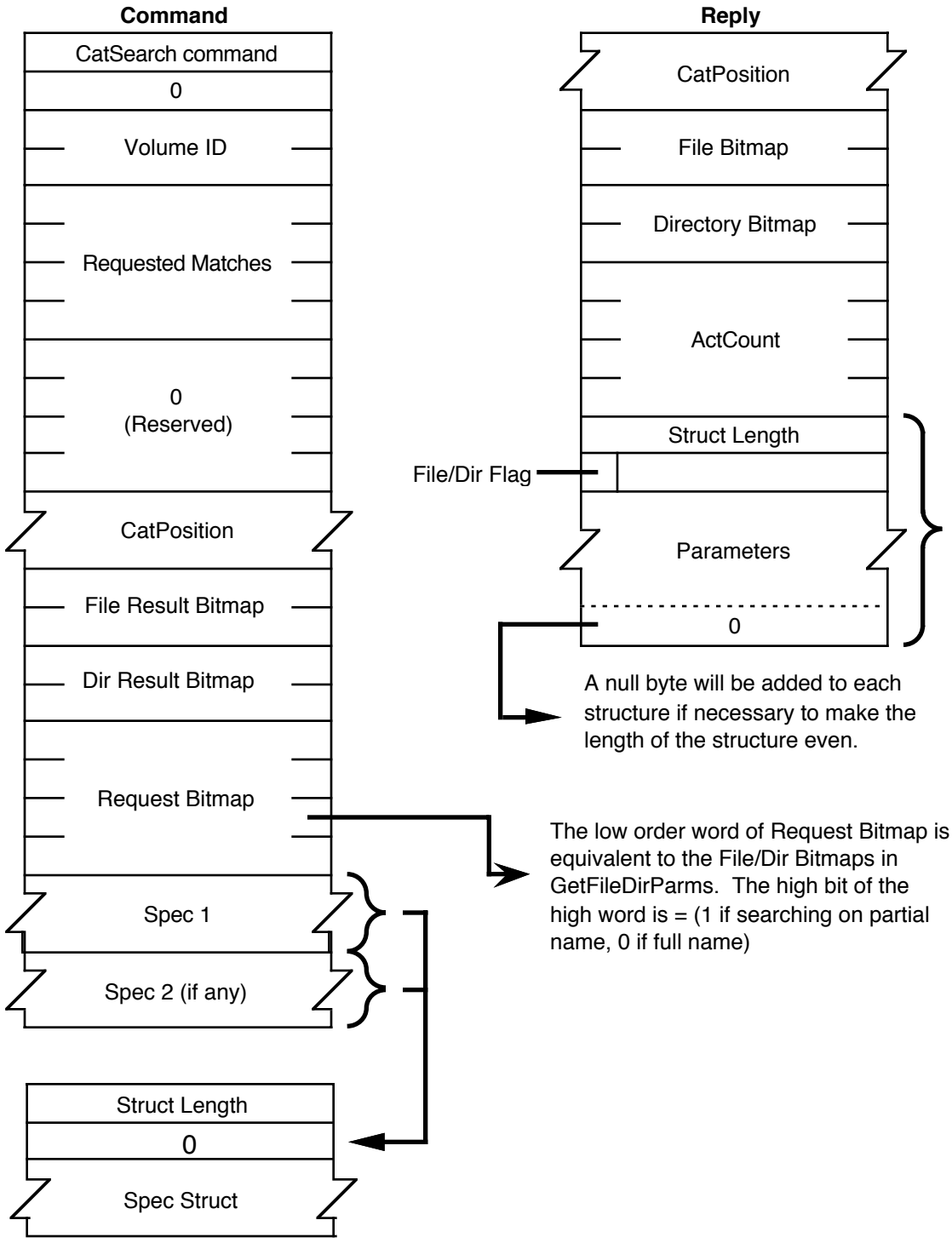
afpCatSearch can search for the following information when searching for directories and files.

Valid Directory & File Bits



Attributes Bits:

The only valid bits that can be searched for in the Attributes parameter are the inhibit bits. For files that's, DeleteInhibit, RenameInhibit, and WriteInhibit. For directories that's, DeleteInhibit and RenameInhibit. You cannot search on Attributes when searching for files and directories.


Command
CatSearch command
0
Volume ID
Requested Matches
0
(Reserved)
CatPosition
File Result Bitmap
Dir Result Bitmap
Request Bitmap
Spec 1
Spec 2 (if any)
Struct Length
0
Spec Struct
Reply
CatPosition
File Bitmap
Directory Bitmap
ActCount
Struct Length
File/Dir Flag
Parameters
0
A null byte will be added to each structure if necessary to make the length of the structure even.
The low order word of Request Bitmap is equivalent to the File/Dir Bitmaps in GetFileDirParms. The high bit of the high word is = (1 if searching on partial name, 0 if full name)

New Function Codes

The following new function codes have been defined for AFP 2.1. Each function code is a 16-bit integer sent in the packet high-byte first.

Decimal value	Hex value	AFP function
38	$0026	afpGetSrvrMsg
39	$0027	afpCreateID
40	$0028	afpDeleteID
41	$0029	afpResolveID
42	$002A	afpExchangeFiles
43	$002B	afpCatSearch

New Result Codes

The following new result codes have been defined for AFP 2.1. Each result codes is a 4-byte long word.

Decimal	Hex	FPError	Description
-5034	$FFFFEC56	afpIDNotFound	Returned when trying to delete a File ID that doesn't exist.
-5035	$FFFFEC55	afpIDExists	Returned when trying to create a File ID for a file that already has a File ID.
-5036	$FFFFEC54	afpDiffVol	Returned when the two files that are to be exchanged by afpExchangeFiles exist on different volumes.
-5037	$FFFFEC53	afpCatalogChanged	Returned when the catalog has changed while doing an afpCatSearch. The CatPosition may have become invalid.
-5038	$FFFFEC52	afpSameObjectErr	Returned when an afpExchangeFiles is on the same file.
-5039	$FFFFEC51	afpBadIDErr	Returned when an afpResolveID is performed on a non-existent FileID.
-5040	$FFFFEC50	afpPwdSameErr	Returned when the user attempts to change their password to the same password as they previously had.
-5041	$FFFFEC4F	afpPwdTooShort	Returned when the user attempts to change their password to a password which is shorter than the server's minimum password length.
-5042	$FFFFEC4E	afpPwdExpired	Returned when the user's password has expired, and the user is required to change their password. The user may login, but may only perform the afpPwdChange call.

Decimal	Hex	FPError	Description
-5043	$FFFFEC4D	afpInsideSharedErr	The folder being shared is inside a shared folder OR the folder contains a shared folder and is being moved into a shared folder OR the folder contains a shared folder and is being moved into the descendent of a shared folder.
-5044	$FFFFEC4C	afpInsideTrashErr	The folder being shared is inside the trash folder OR the shared folder is being moved into the trash folder OR the folder is being moved to the trash and it contains a shared folder.