AppleShare IP 6.3
Developer's Kit

AppleTalk Filing Protocol

Update to *Inside Macintosh: Networking*, Chapter 9

Technical Publications
© Apple Computer, Inc. 1999

 Apple Computer, Inc.
© 1997-1999 Apple Computer, Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Computer, Inc., except to make a backup copy of any documentation provided on CD-ROM.

The Apple logo is a trademark of Apple Computer, Inc.
Use of the “keyboard” Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this book. Apple retains all intellectual property rights associated with the technology described in this book. This book is intended to assist application developers to develop applications only for Apple-labeled or Apple-licensed computers.

Every effort has been made to ensure that the information in this manual is accurate. Apple is not responsible for typographical errors.

Apple Computer, Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, and Macintosh are trademarks of Apple Computer, Inc., registered in the United States and other countries.

Adobe, Acrobat, and PostScript are trademarks of Adobe Systems Incorporated or its subsidiaries and may be registered in certain jurisdictions.

Helvetica and Palatino are registered trademarks of Linotype-Hell AG and / or its subsidiaries.

ITC Zapf Dingbats is a registered trademark of International Typeface Corporation.

QuickView™ is licensed from Altura Software, Inc.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this manual, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS MANUAL IS SOLD “AS IS,” AND YOU, THE PURCHASER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS MANUAL, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

Listings and Tables v

Preface About This Manual vii

Conventions Used in This Manual vii
For More Information viii

Chapter 1 AppleTalk Filing Protocol (AFP) 11

About AFP 12
AFP Reference 14
- Data Structures 14
  - DSParamBlock Structure 14
  - PB Control and PB Status Codes 17
  - AFPInsRemSMBParam Structure 18
  - AFPSrvrInfo Structure 19
  - DSGetStatusPB Structure 20
  - DSOpenPB Structure 21
  - DSWritePB Structure 22
  - DSXPortInfo Structure 22
  - GetVolSessInfoPB Structure 23
  - GetVolSessInfoRec Structure 24
  - User Authentication Constants 25
  - AFP Gestalt Constants 26
- Routines 27
  - NAFPCommandAsync Function 32
  - NAFPCommandSync Function 32
  - NAFPCommandImmediate Function 33
  - Completion Routine 34

Summary of AFP 34
- Pascal Summary 34
  - Constants 34
  - Data Types 35

Routines 40
C Summary 40
Constants 40
Data Types 41
Routines 47
Assembly-Language Summary 47
Constants 47
Miscellaneous 48
Data Structures 48
Result Codes 53

Index IN-1

Listings and Tables

Figure 1-1	The .AFPTranslator driver	13
Table 1-1	PB control codes	17
Table 1-2	PB status codes	18
Table 1-3	AFP command codes	28

About This Manual

This document describes the .AFPTranslater driver, which was implemented for AppleShare IP in order to provide transport independence for the AppleTalk Filing Protocol. The .AFPTranslator driver accepts Hierarchical File System (HFS) and AFP commands from applications and sends them to the data stream interface or the .XPP driver depending on the transport protocol that the command uses.

This document replaces Chapter 9 of *Inside Macintosh: Networking*.

Conventions Used in This Manual

The Courier font is used to indicate server control calls, code, and text that you type. Terms that are defined in the glossary appear in boldface at first mention in the text. This guide includes special text elements to highlight important or supplemental information:

Note
Text set off in this manner presents sidelights or interesting points of information. ◆

IMPORTANT
Text set off in this manner—with the word Important—presents important information or instructions. ▲

▲ **WARNING**
Text set off in this manner—with the word Warning—indicates potentially serious problems. ▲

For More Information

The following books provide information that is important for all AppleShare developers:

- *AppleShare IP Administrator's Manual*. Apple Computer, Inc.
- *Inside Macintosh*. Apple Computer, Inc.

For information on the programming interface for managing users and groups, see the following publication:

- *AppleShare IP 6.3 Developer's Kit*: *AppleShare Registry Library*. Apple Computer, Inc.

For additional information on the AppleTalk Filing Protocol (AFP), see the following publications:

- *AppleShare IP 6.3 Developer's Kit*: *AppleTalk Filing Protocol Version 2.1 and 2.2.* Apple Computer, Inc.
- *Inside AppleTalk*, Second Edition. Apple Computer, Inc.

For information on user authentication modules (UAMs), see the following publication:

- *AppleShare IP 6.3 Developer's Kit*: *User Authentication Modules.* Apple Computer, Inc.

For information on AppleShare IP Print Server security mechanisms, see the following publication:

- *AppleShare IP 6.3 Developer's Kit*: *AppleShare IP Print Server Security Protocol.* Apple Computer, Inc.

For information on controlling an AppleShare file server and handling server events, see the following publication:

- *AppleShare IP 6.3 Developer's Kit*: *Server Control Calls and Server Event Handlng.* Apple Computer, Inc.

For information on using the AppleShare IP File Server 6.3 and Macintosh File Sharing, see the following manuals:

- *AppleShare Client User's Manual.* Apple Computer, Inc.
- *Macintosh Networking Reference*. Apple Computer, Inc.

AppleTalk Filing Protocol (AFP)

This chapter describes the AppleTalk Filing Protocol (AFP) that allows a workstation on an AppleTalk network to access and manipulate files on an AFP file server, such as an AppleShare server.

Because you can use the native file system to access an AFP server from a workstation, in most cases you should not need to use AFP directly. For example, few application developers use AFP to access an AppleShare file server because the existing File Manager commands perform most of the functions needed to access and manipulate files on an AppleShare server.

However, if you want to provide functions that are not implemented by the native file system commands or you want to manipulate files on an AFP server other than an AppleShare server, your application can use the AFP programming interface to directly access AFP to send commands to the server. For example, you can use AFP to list the contents of a directory when you need to obtain ProDOS information. You can also use AFP to retrieve or set parameters for a specific file when ProDOS is used.

This chapter describes the programming interface to the workstation portion of AFP only. It does not describe how to implement an AFP server. For information on how to implement an AFP server, see *Inside AppleTalk*, second edition.

Because AFP is not widely used by application program developers, this chapter provides only the AFP basics. This chapter includes an “About” and “Reference” sections. It does not include a “Using” section, as do most of the other chapters in this book. This chapter is included in this book to complete the coverage of the AppleTalk protocol stack.

If you decide to use AFP, it is important to note that to implement an AFP command, you need information in addition to the information that this chapter provides. *Inside AppleTalk*, second edition, and the *AppleShare IP 6.3 Developer’s Kit*, provide information describing the AFP commands and the command block data structure required for each command. The *AppleShare IP*

IP 6.3 Developer's Kit includes extensions to AFP not described in *Inside AppleTalk*.

AFP is built on top of the AppleTalk Session Protocol (ASP) and uses the services of ASP. To use AFP, you should also be familiar with ASP, which is described in the chapter "AppleTalk Session Protocol (ASP)" in this book. For an overview of AFP and how it fits within the AppleTalk protocol stack, read the chapter "Introduction to AppleTalk," in *Inside Macintosh: Networking*.

About AFP

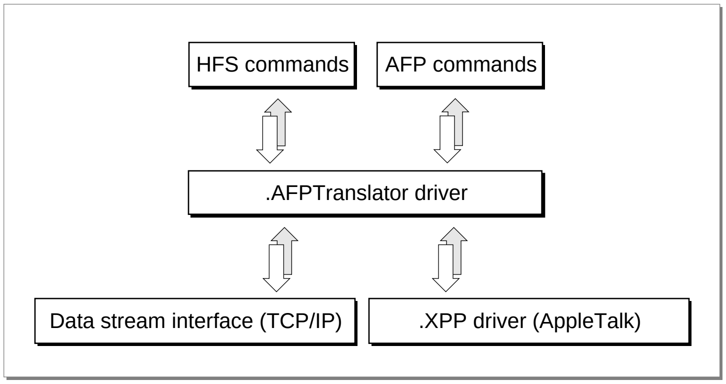
AFP is a remote filing system protocol that provides a workstation on a network with access to a server that is implemented according to the AFP file system structure. AFP also includes user authentication support and an access control mechanism that supports volume-level and folder-level access rights. AppleShare is the AFP file server that is implemented on Macintosh computers.

Through the native file system and AFP, your application running on one node can manipulate files on another node using the same file system commands on the remote node that it uses to manipulate files on its own node. You can use AFP commands to

- obtain and modify information about the file server and other parts of the file system structure
- create and delete files and directories
- read files or write to them
- retrieve and store information within individual files

The .AFPTranslator driver allows the workstation to access the server via AppleTalk or the Transmission Control Protocol / Internet Protocol (TCP / IP). Figure 9-1 shows the .AFPTranslator driver and its position in the flow of data between HFS and AFP commands and the data stream interface (for TCP / IP) and the .XPP driver (for AppleTalk).

Figure 1-1 The .AFPTranslator driver



For information on the data stream interface (DSI), see the *AppleTalk Filing Protocol Version 2.1 and 2.2* in the *AppleShare IP 6.3 Developer's Kit*.

Note
Prior to AppleShare Client 3.7, the .XPP driver was responsible for sending AFP commands from the client to the server. With AppleShare Client 3.7, the .AFPTranslator driver is responsible for sending AFP commands to the server. The .AFPTranslator driver and the .XPP driver do not use the same session reference number, so if AFP commands are sent to the .XPP driver when the AppleShare Client 3.7 is installed and the session is running over TCP/IP, errors will occur. If the session is running over AppleTalk and `afpSVolInfo` is used to get the session reference number, the session may be lost. ◆

The programming interface to the .AFPTranslator driver on the workstation consists of three functions:

- `NAFPCommandSync`, for sending commands to an AFP server synchronously
- `NAFPCommandAsync`, for sending commands to an AFP server asynchronously
- `NAFPCommandImmediate`, for sending a command to an AFP server without going through the Device Manager queue, such as a command to close all open sessions with all connected AFP servers

The three functions pass to the .AFPTranslator driver the command code and parameters for an AFP command.

Note
Because the native file system commands implement most of the functions that you need to access an AFP server, in most cases you will not need to use AFP directly. For this reason, this chapter does not include a “Using” section, as do most of the other chapters in this book. If the native file system implements a function you need, you should use the file system command. If you want to implement a function that is not part of the native file system, you can use AFP directly. In this case, you should continue to read this chapter. ◆

AFP Reference

This section describes the data structures and functions that are specific to the AppleTalk Filing Protocol (AFP).

The “Data Structures” section describes the constants and data structures used by the AFP functions.

The “Routines” section describes the AFP programming interface, which consists of three functions (`NAFPCommandAsync`, `NAFPCommandSync`, and `NAFPCommandImmediate`), which allow you to communicate with an AFP server and specify from within a `DSParamPB` structure a particular command and its parameters to send across the session to the server.

Data Structures

This section describes the data structures that you use to provide information to the AppleTalk Filing Protocol (AFP).

DSParamBlock Structure

You pass a `DSParamBlock` structure as a parameter to the three AFP functions: `AFPNAFPCommandSync`, `NAFPCommandAsync`, and `NAFPCommandImmediate`.

The first four fields of the DSParamBlock structure, qLink, qType, ioTrap, and ioCmdAddr, are used internally by the Device Manager.

You must specify the .AFPTranslator driver reference number as the input value of the ioCRefNum field. You can use the Device Manager's OpenDriver function to obtain the .AFPTranslator driver reference number.

```
struct DSParamBlock {
    QElem *             qLink;
    short               qType;
    short               ioTrap;
    Ptr                 ioCmdAddr;
    DSIOCompletionUPP   ioCompletion;
    OSErr               ioResult;
    long                cmdResult;
    short               ioVRefNum;
    short               ioCRefNum;
    short               csCode;
    short               dsTimeout;
    short               dsReserved1;
    long                dsRetry;
    UInt16              dsReserved2;
    short               dsSessRefNum;
    short               dsReserved3;
    short               dsCmdBufferSize;
    UInt8 *             dsCmdBuffer;
    UInt32              dsReplyBufferSize;
    UInt8 *             dsReplyBuffer;
    union {
                        DSOpenPB open;
                        DSWritePB write;
                        DSGetStatusPB status;
    } csParam;
};
typedef struct DSParamBlock DSParamBlock;
```

Field descriptions

qLink	Reserved.
qType	Reserved.
ioTrap	Reserved.
ioCmdAddr	Reserved.

`ioCompletion`	A pointer to a completion routine (page 1-34) if the structure is being passed as a parameter to `NAFPCommandAsync`.
`ioResult`	Contains the result when the `DSParamPB` structure is passed as an parameter to `NAFPCommandAsync`.
`cmdResult`	Four bytes of data returned from the server indicating the result of an AFP command.
`ioVRefNum`	Reserved.
`ioCRefNum`	Reference number from the .AFPTranslator driver.
`csCode`	The PB control or PB status code (page 1-17) for this operation.
`dsTimeout`	For sessions over AppleTalk, the interval in seconds that the .AFPTranslator driver waits between retries of the AFP command. For sessions over TCP/IP, `dsTimeout` is reserved.
`dsReserved`	Reserved.
`dsRetry`	For sessions over AppleTalk, the number of times to resend the request. For sessions over TCP/IP, `dsRetry` is reserved.
`dsReserved2`	Reserved.
`dsSessRefNum`	The session reference number, which is a unique number that the .AFPTranslator driver assigns to the session and returns in response to an `afpLogin` command.
`dsReserved3`	Reserved.
`dsCmdBufferSize`	The size in bytes of `dsCmdBuffer`. The size of the `dsCmdBuffer` must not exceed the value of `aspMaxCmdSize` (576 bytes) that the `ASPGetParms` function returns.
`dsCmdBuffer`	A pointer to the command buffer, which contains command parameters associated with the command code stored in `csCode`. When the value of `cscode` is `dsOpenSession`, `dsCmdBuffer` must be `kFPLogin`. When the value of `cscode` is `dsCloseSession`, the value of `dsCmdBuffer` must be `kFPLogout`.
`dsReplyBufferSize`	On input, the size in bytes of `dsReplyBuffer`, which is to hold the expected response to the AFP command. On return, `dsReplyBufferSize` contains the actual size of the reply pointed to by `dsReplyBuffer`.
`dsReplyBuffer`	A pointer to the reply buffer.

`csparam`	A union whose value can be a `PBOpenPB`, `DBWritePB`, or `DBGetStatus` structure for opening a session, writing, and getting the server's status, respectively.

PB Control and PB Status Codes

You provide a PB control code or a PB status code in the `csCode` field of the `DSParamPB` structure (page 1-22) to specify the type of operation for which the structure will be used when it is passed as a parameter to `NAFPCommandAsync`, `NAFPCommandSync`, or `NAFPCommandImmediate`. Table 1-1 lists the PB control codes.

Table 1-1 PB control codes

Constant	Code	Meaning
`dsIOCTL`	231	Makes an IOCTL call to the session encpoint if the transport protocol for the session is TCP / IP.
`dsCloseAll`	232	Closes all open sessions**.**
`dsCloseSession`	237	Closes the specified session.
`dsSendRequest`	240	Sends an AFP command to the server. For the list of AFP commands, see Table 1-3.
`dsGetStatus`	243	Sends a `GetSrvrInfo` request to the server. The value of `DSParamPB.csParam` must be a `DSGetStatusPB` structure.
`dsOpenSession`	244	Opens an AFP session with the specified server. The value of `DSParamPB.csParam` must be a `DSOpenPB` structure.
`AFPRemSessMemBlk`	245	Removes the session memory block from the .AFPTranslator driver's queue.
`AFPInsSessMemBlk`	246	Inserts the session memory block into the .AFPTranslator driver's queue.
`afpGetAttnRoutine`	252	Returns a pointer to the default AFP attention routine.

Table 1-2 lists the PB status codes.

Table 1-2 PB status codes

Constant	Code	Meaning
`afpGetSMBSize`	118	Gets the size of the session memory block.
`afpXGetVolInfo`	121	Gets extended information about the specified volume, such as server time offset and volume grade.
`afpSVolInfo`	124	Returns information in a `GetVolSessInfoRec` structure about the specified server, such as its name and OT address, and the name the user used to connect.
`afpGetFSID`	127	Gets the file system ID.
`dsGetXportInfo`	236	Gets information about the session's transport protocol.

AFPInsRemSMBParam Structure

You pass an `AFPInsRemSMBParam` structure as a parameter when you send a PB control code of `AFPInsSessMemBlk` to insert the session memory block into the .AFPTranslator driver's queue or a PB control code of `AFPRemSessMemBlk` to remove the session memory block from the .AFPTranslator driver's queue.

Note
You must insert the session memory block after you successfully open a session with a server. After you close a session with a server, you should remove the session memory block.

```
struct AFPInsRemSMBParam {
    QElemPtr        qLink;
    short           qType;
    short           ioTrap;
    Ptr             ioCmdAddr;
    IOCompletionUPP ioCompletion;
    OSErr           ioResult;
    StringPtr       ioNamePtr;
    short           ioVRefNum;
    short           ioRefNum;
    short           csCode;
    Ptr             smbPtr;
    };
```

Field descriptions

`qLink`	Reserved.
`qType`	Reserved.
`ioTrap`	Reserved.
`ioCmdAddr`	Reserved.
`ioCompletion`	On input, a pointer to an completion routine (page 1-34).
`ioResult`	On output, if the `AFPInsRemSMBParam` structure is passed as a parameter to `NAFPCommandAsync`, contains the result code.
`ioNamePtr`	Reserved.
`ioVRefNum`	Reserved.
`ioRefNum`	On input, the driver reference number provided by the .AFPTranslator driver.
`csCode`	On input, `AFPInsSessMemBlk` to insert the session memory block or `AFPRemSessMemBlk` to remove the session memory block.
`smbPtr`	On input and output, a pointer to the session memory block that is to be inserted or removed.

AFPSrvrInfo Structure

You receive a `AFPSrvrInfo` structure containing information about the server when you send a PB status code of `dsGetStatus` to the server.

Note
This section describes only the fixed portion of the `AFPSrvrInfo` structure. The variable-length portion of this structure is described in the *AppleTalk Filing Protocol Version 2.1 and 2.2* document in the *AppleShare IP 6.3 Developer's Kit*. ◆

```
struct AFPSrvrInfo {
    short           fMachineOffset;
    short           fVerCountOffset;
    short           fUAMCountOffset;
    short           fIconOffset;
    short           fFlags;
    unsigned char   fSrvrName[2];
};
typedef struct AFPSrvrInfo AFPSrvrInfo;
```

Field descriptions

`fMachineOffset`	Offset to the server's machine type.
`fVerCountOffset`	Offset to the number of versions of AFP that the server supports.
`fUAMCountOffset`	Offset to the number of UAMs that the server supports.
`fIconOffset`	Offset to the server's icon.
`fFlags`	Values that describe the server's capabilities. For details, see the enumeration later in this section.
`fSrvrName`	Offset to the server's name.

The following enumeration describes server capabilities returned in the `fFlags` field.

```
enum {
    srvSCopyFile    = 0, /* Server supports FPCopyFile call */
    srvSChangePswd  = 1, /* Server supports FPChangePassword call */
    srvNoPswdSave   = 2, /* Workstation should not save password */
    srvSServerMsgs  = 3, /* Server supports server messages */
    srvSSrvrSig     = 4, /* Server supports server signatures (AFP 2.2)*/
    srvSupportsTCP  = 5, /* Server supports TCP/IP (AFP 2.2) *
    srvSNotification= 6  /* Server will send notifications (AFP 2.2) */
};
```

DSGetStatusPB Structure

You use a `DSGetStatusPB` structure when you send PB control code of `dsGetStatus` command code to the server. The `DSGetStatusPB` structure identifies the address of the server that is to return status information in an `AFPSrvrInfo` structure (page 1-19).

```
struct DSGetStatusPB {
    OTAddress *     dsGSSrvrAddress;
    const char *    dsGSEpString;
};
typedef struct DSGetStatusPB DSGetStatusPB;
```

Field descriptions

`dsGSSrvrAddress`	The OT address of the server, which is either an `OTDDPAddress` (for AppleTalk) or an `InetAddress` (for TCP/IP).
`dsGSEpString`	The endpoint string for the connection. The default is `nil`. The endpoint string provides a way to specify streams configuration information on a per-connection basis. It is only used for TCP/IP connections and is ignored for AppleTalk connections.

You also need to fill in the `dsReplyBuffer` and `dsReplyBufferSize` fields in the `DSParamPB` structure.

Note

You should make the `dsReplyBuffer` field of the `DSParamBlock` structure at least 1024 bytes in size. ◆

DSOpenPB Structure

You use a `DSOpenPB` structure when you send a PB control code of `dsOpenSession` to the server.

```
struct DSOpenPB {
    AttnRoutineUPP  dsOSAttnRoutine;
    OTAddress *     dsOSSrvrAddress;
    Ptr             dsOSSessionBlock;
    const char      *dsOSEpString;
};
typedef struct DSOpenPB DSOpenPB;
```

Field descriptions

`dsOSAttnRoutine`	A custom attention routine. To use the default attention routine, set `dsOSAttnRoutine` to `nil`.

`dsOSSrvrAddress`	The OT address of the server, which is either an `OTDDPAddress` (for AppleTalk) or an `InetAddress` (for TCP / IP), with which a session is to be opened.
`dsOSSessionBlock`	A pointer to the block of memory reserved for this session. At minimum, the size of the block must be `SMBsize`.
`dsOSEpString`	The endpoint string for the connection. To use the default endpoint string, set `dsOSEpString` to `nil`. The endpoint string provides a way to specify streams configuration information on a per-connection basis. It is only used for TCP / IP connections and is ignored for AppleTalk connections.

DSWritePB Structure

You use a `DSWritePB` structure when you send the AFP command `afpWrite` to the server.

```
struct DSWritePB {
    UInt32  dsWriteDataOffset;
    UInt32  dsWriteBufferSize;
    Byte *  dsWriteBuffer;
};
typedef struct DSWritePB DSWritePB;
```

Field descriptions

`DSWriteDataOffset`	The offset at which the data is to be written.
`dsWriteBufferSize`	The size of the data that is to be written.
`dsWriteBuffer`	A pointer to the data that is to be written.

DSXPortInfo Structure

You use a `DSXPortInfo` structure when you call `NAFPCommandAsync` or `NAFPCommandSync` with a `DSParamPB` structure (page 1-22) whose `csCode` field is `dsGetXPortInfo`. The `dsGetXPortInfo` structure contains extended port information.

```
struct DSXPortInfo {
    long dsXPortType;       /* Transport Type (kASPXport, kTCPXport) */
    short dsXPortSessRefNum;/* Session reference number for ASP or TCP */
    union {
        InetAddress ipAddr;
        DDPAddress ddpAddr;
    }addr;
};
typedef struct DSXPortInfo DSXPortInfo;
```

Field descriptions

dsXPortType	On return, the transport type of the specified session (kASPXport for AppleTalk or kTCPXport for TCP/IP).
dsXPortSessRefNum	On input, the session reference number of the session for which you want to determine the transport type.
addr	On input, the IP address (for TCP/IP sessions) or the AppleTalk address (for AppleTalk sessions) of the server for which you want to determine the transport type.

GetVolSessInfoPB Structure

You use a GetVolSessInfoPB structure when you call NAFPCommandAsync or NAFPCommandSync with a DSParamPB structure (page 1-22) whose csCode field is dsGetSVolInfo.

The server returns in the GetVolSessionInfoPB structure information about the volume for which there is an open session, such as the AFP version number, session reference number, volume ID, server address, UAM type, and pointers to the user name string, volume icon, and Get Info “where” string in a GetVolSessInfoRec structure (page 1-24).

```
struct GetVolSessInfoPB {
    QElemPtr                qLink;
    short                   qType;
    short                   ioTrap;
    Ptr                     ioCmdAddr;
    IOCompletionUPP         ioCompletion;
    OSErr                   ioResult;
    StringPtr               ioNamePtr;
    short                   ioVRefNum;
    short                   ioRefNum;
    short                   csCode;
    Ptr                     vcbPtr;
    GetVolSessInfoRecPtr    sessInfoBuffer;
    long                    sessInfoSize;
    long                    actSessInfoSize;
};
```

Field descriptions

`qLink`	Reserved.
`qType`	Reserved.
`ioTrap`	Reserved.
`ioCmdAddr`	Reserved.
`ioCompletion`	A pointer to an I/O completion routine (page 1-34).
`ioResult`	On output, the result when the `DSParamPB` structure is passed as an parameter to `NAFPCommandAsync`.
`ioNamePtr`	Reserved.
`ioVRefNum`	Reserved.
`ioRefNum`	The driver reference number provided by the .AFPTranslator driver.
`csCode`	Always the `afpSVolInfo` command.
`vcbPtr`	On input, a pointer to the volume control block (VCB) for the volume for which you are getting volume information.
`sessInfoBuffer`	On input, a pointer to the `GetVolSessInfoRec` structure in which information about the volume is to be placed.
`sessInfoSize`	On input, the size in bytes of `sessInfoBuffer`.
`actSessInfoSize`	On output, the size in bytes of the data returned in `sessInfoBuffer`.

GetVolSessInfoRec Structure

You receive a `GetVolSessInfoRec` structure when you call `NAFPCommandAsync` or `NAFPCommandSync` with a `DSParamPB` structure whose `csCode` member is `afpSVolInfo`. The `GetVolSessInfoRec` structure contains basic information, such as the server's name and address and the name of the user who connected to the volume.

```
struct GetVolSessInfoRec {
    short       sessAFPVersion;
    short       sessReferenceNumber;
    short       sessAFPVolID;
    OTAddress * sessServerAddress;
    short       sessUAMType;
    StringPtr   sessUserNamePtr;
    Ptr         sessVolIconPtr;
    StringPtr   sessWhereStringPtr;
};
typedef struct GetVolSessInfoRec GetVolSessInfoRec;
```

Field descriptions

`sessAFPVersion`	On output, the version of AFP being used for the session as defined by the following enumeration: `enum { kAFPVersion11 = 1, kAFPVersion20 = 2, kAFPVersion21 = 3, kAFPVersion22 = 4 };`
`sessReferenceNumber`	On output, the AFP session reference number for this session.
`sessAFPVolID`	On output, the volume's AFP volume identifier.
`sessServerAddress`	On output, the server's address. For AppleTalk sessions, `sessServerAddress` is an AppleTalk address; for TCP/IP sessions, `sessServerAddress` is an IP address.
`sessUAMType`	On output, a constant that describes the user authentication method (page 1-25) that was used to authenticate the session.
`sessUserNamePtr`	A pointer to the user name string.
`sessVolIconPtr`	A pointer to the server volume icon mask for this volume.
`sessWhereStringPtr`	A pointer to the "where" information string, which the Finder displays in the Get Info window for this volume.

User Authentication Constants

Information about the user authentication method (UAM) that was used to authenticate a session is returned in the `UAMType` field of the `GetVolSessInfoRec` structure (page 1-24).

```
enum {
    kNoUserAuth         = 1,    /* No User Authentication UAM (Guest) */
    kCleartextAuth      = 2,    /* Cleartext Password UAM */
    kRandnumAuth        = 3,    /* Random Number Exchange UAM */
    k2WayRandnumAuth    = 6,    /* Two-Way Random Number Exchange UAM */
    kEncryptPassXport   = 7,    /* DHXCAST128 UAM */
    kMinCustomUAM       = 128   /* Minimum value for a custom UAM */
};
```

Note
Authentications that begin with `kCleartextAuth` or `kRandnumAuth` are automatically convered to `k2WayRandnumAuth` if `k2WayRandnumAuth` is available. ◆

AFP Gestalt Constants

The following AFP gestalt constants can be used to determine the capabilities of an AFP client.

```
enum {
    gestaltAFPClient            = FOUR_CHAR_CODE('afps'),
    gestaltAFPClientVersionMask = 0x0000FFFF, /* Low word is version */
    gestaltAFPClient3_5         = 1,
    gestaltAFPClient3_6         = 2,
    gestaltAFPClient3_6_1       = 3,
    gestaltAFPClient3_6_2       = 4,
    gestaltAFPClient3_6_3       = 5,  /* Includes 3.6.4, 3.6.5 */
    gestaltAFPClient3_7         = 6,  /* Includes 3.7.1 */
    gestaltAFPClient3_7_2       = 7,  /* Includes 3.7.3 */
    gestaltAFPClient3_8         = 8,
    gestaltAFPClientCfgMask     = (long)0xFFFF0000, /* high word is
                                        config */
    gestaltAFPClientCfgRsrc     = 16, /* Client uses config resources */
    gestaltAFPClientUAMv2       = 28, /* Client supports the 2.0 UAM
                                        interfaces */
    gestaltAFPClientSupportsIP  = 29, /* Client supports AFP over
                                        TCP/IP */
    gestaltAFPClientVMUI        = 30, /* Client can put up UI from the
                                        PBVolMount trap */
    gestaltAFPClientMultiReq    = 31  /* Client supports multiple
                                         outstanding requests */
};
```

Routines

The programming interface to AFP is different in form from the programming interfaces to the other AppleTalk protocols described in this book. For AFP, the programming interface consists of three functions:

- `NAFPCommandAsync`, which allows you to call AFP asynchronously and pass it the command code for a particular AFP command.
- `NAFPCommandSync`, which allows you to call AFP synchronously and pass it the command code for a particular AFP command.
- `NAFPCommandImmediate`, which you use to bypass the Device Manager and send an AFP command directly to the server so that the server can act on the command immediately.

All three functions require as a parameter a pointer to a data stream (DS) parameter block. The DS parameter block's `csCode` field contains a value that identifies the purpose for which the parameter block is intended. To send an AFP command, you specify a pointer to a command buffer, the first byte of which contains the AFP command, followed by any information required for the specified AFP command.

Before you send an AFP command code, you must use the Device Manager's `OpenDriver` function to open the .AFPTranslator driver.

In most circumstances, you should not close the .AFPTranslator driver because other applications and processes could be using the protocols implemented by the .AFPTranslator driver.

Table 1-3 lists the AFP command codes.

Table 1-3 AFP command codes

AFP command constant	Command code	Action
`kFPByteRangeLock`	1	Locks or unlocks a specified range of bytes within an open fork.
`kFPCloseVol`	2	Informs the server that the workstation no longer needs the volume.
`kFPCloseDir`	3	Closes a directory and invalidates its directory identifier.
`kFPCloseFork`	4	Closes a fork that was opened by `kFPOpenFork`.
`kFPCopyFile`	5	Copies a file from one location to another on the same file server.
`kFPCreateDir`	6	Creates a new directory.
`kFPCreateFile`	7	Creates a new file.
`kFPDelete`	8	Deletes a file or directory.
`kFPEnumerate`	9	Lists the contents of a directory.
`kFPFlush`	10	Writes to a disk any volume data that has been modified.
`kFPFlushFork`	11	Writes to a disk any data buffered from previous `kFPWrite` calls.
`kFPGetForkParms`	14	Retrieves parameters for a file associated with a particular open fork.

continued

Table 1-3 AFP command codes (continued)

AFP command constant	Command code	Action
kFPGetSrvrInfo	15	Obtains a block of descriptive information from the server, without requiring an open session. Use the ASPGetStatus function instead of this command code. See the chapter “AppleTalk Session Protocol (ASP)” in this book for information on ASPGetStatus. Making an kFPGetSInfo call using the AFPCommand results in an error.
kFPGetSrvrParms	16	Retrieves server parameters.
kFPGetVolParms	17	Retrieves parameters for a particular volume.
kFPLogin	18	Establishes an AFP session with a server.
kFPLoginCont	19	Continues the login and user authentication process started by the kFPLogin command.
kFPLogout	20	Terminates a session with a server.
kFPMapID	21	Maps a user ID to a user name, or a group ID to a group name.
kFPMapName	22	Maps a user name to a user ID, or a group name to a group ID.
kFPMoveAndRename	23	Moves a directory or file to another location on the same volume.
kFPOpenVol	24	Makes a volume available to the workstation.
kFPOpenDir	25	Opens a directory on a variable directory ID volume and returns its directory ID.
kFPOpenFork	26	Opens the data or resource fork of an existing file to read from it or write to it.
kFPRead	27	Reads a block of data from an open fork.

Table 1-3 AFP command codes (continued)

AFP command constant	Command code	Action
`kFPRename`	28	Renames a directory or file.
`kFPSetDirParms`	29	Sets parameters for a specified directory.
`kFPSetFileParms`	30	Sets parameters for a specified file.
`kFPSetForkParms`	31	Sets the fork length for a specified open fork.
`kFPSetVolParms`	32	Sets the backup date for a specified volume.
`kFPWrite`	33	Writes a block of data to an open fork.
`kFPGetFlDrParms`	34	Retrieves parameters for either a file or a directory.
`kFPChangePassword`	36	Changes a user's password.
`kFPSetFlDrParms`	35	Sets parameters for a file or directory.
`kFPGetUserInfo`	37	Gets user information.
`kFPGetSrvrMsg`	38	Gets a string message from the server, such as shutdown, user, and login messages.
`kFPCreateID`	39	Creates a unique file ID for a specified file.
`kFPDeleteID`	40	Invalidates all instances of a specified file ID.
`kFPResolveID`	41	Returns parameters for the file referred to by the specified file ID.
`kFPExchangeFiles`	42	Preserves an existing file ID when an application performs a “Save” or “Save As” operation.
`kFPCatSearch`	43	Allows an application to efficiently search an entire volume for files that match specified criteria.

continued

Table 1-3 AFP command codes (continued)

AFP command constant	Command code	Action
`kFPOpenDT`	48	Opens the Desktop database on a particular volume.
`kFPCloseDT`	49	Informs the server that the workstation no longer needs the volume’s Desktop database.
`kFPGetIcon`	51	Retrieves an icon from the volume’s Desktop database.
`kFPGtIconInfo`	52	Retrieves icon information from the volume’s Desktop database.
`kFPAddAPPL`	53	Adds an APPL mapping to the Desktop database.
`kFPRemoveAPPL`	54	Removes an APPL mapping from the volume’s Desktop database.
`kFPGetAPPL`	55	Retrieves an APPL mapping from the volume’s Desktop database.
`kFPAddComment`	56	Adds a comment for a file or directory to the volume’s Desktop database.
`kFPRemoveComment`	57	Removes a comment from the volume’s Desktop database.
`kFPGetComment`	58	Retrieves a comment associated with a specified file or directory from the volume’s Desktop database.
`kFPAddIcon`	192	Adds an icon bitmap to the volume’s Desktop database.

For a description of the commands and their required command block formats and parameters, see the following documents:

- For command codes 38 through 43, inclusive, see the *AppleTalk Filing Protocol Version 2.1 and 2.2* document in the *AppleShare IP 6.3 Developer's Kit*.
- For all other AFP command codes, see *Inside AppleTalk*, second edition.

NAFPCommandAsync Function

Communicate asynchronously with an AFP server.

```
OSErr NAFPCommandAsync (DSParamBlockPtr paramblock);
```

`paramBlock`	A pointer to a `DSParamBlock` structure (page 1-22).

DESCRIPTION

You use the `NAFPCommandAsync` function to communicate asynchronously with an AFP server.

To prepare `paramblock` for sending to a server, set the `csCode` field to an appropriate PB control (page 1-17) or PB status code (page 1-18) and set the `dsCmdBuffer` field so that it contains the data structure that is appropriate for the value of `csCode`.

To send an AFP command to the server, set the `csCode` field to `dsSendRequest`, and prepare the `dsCmdBuffer` field so that it contains the appropriate AFP command code (listed in Table 1-3 (page 28)), followed by the command block for the specified code. For information on command block formats for command codes 38 through 43, inclusive, see the *AppleTalk Filing Protocol Version 2.1 and 2.2* in the *AppleShare IP 6.3 Developer's Kit* . For information on command block formats for all other AFP command codes, see *Inside AppleTalk*, second edition.

RESULT CODES

The `NAFPCommandSync` function can return any of the result codes listed in “Result Codes” (page 53).

NAFPCommandSync Function

Communicate synchronously with an AFP server.

```
OSErr NAFPCommandSync (DSParamBlockPtr paramBlock);
```

`paramBlock`	A pointer to a `DSParamBlock` structure (page 1-22).

DESCRIPTION

The `NAFPCommandSync` function provides a way to send an AFP command to the server and receive a reply synchronously.

To prepare `paramblock` for sending to a server, set the `csCode` field to an appropriate PB control (page 1-17) or PB status code (page 1-18) and set the `dsCmdBuffer` field so that it contains the data structure that is appropriate for the value of `csCode`.

To prepare `paramblock` for sending an AFP command, set the `csCode` field to `dsSendRequest`, and prepare the `dsCmdBuffer` field so that it contains the appropriate AFP command code (listed in Table 1-3 (page 28)), followed by the command block for the specified code. For information on command block formats for command codes 38 through 43, inclusive, see the *AppleTalk Filing Protocol Version 2.1 and 2.2* in the *AppleShare IP 6.3 Developer's Kit*. For information on command block formats for all other AFP command codes, see *Inside AppleTalk*, second edition.

RESULT CODES

The `NAFPCommandSync` function can return any of the result codes listed in “Result Codes” (page 53).

NAFPCommandImmediate Function

Communicate directly with an AFP server.

```
OSErr NAFPCommandImmediate (DSParamBlockPtr paramBlock);
```

`paramblock`	A pointer to a `DSParamBlock` structure (page 1-22).

DESCRIPTION

You use the `NAFPCommandImmediate` function to bypass the Device Manager and send commands directly to a server for immediate response. You typically use the `NAFPCommandImmediate` function to send a command that requires immediate attention, such as `dsCloseAll` to close all open sessions immediately.

RESULT CODES

The NAFPCommandImmediate function can return any of the result codes listed in "Result Codes" (page 53).

Completion Routine

The DSParamPB, GetVolSessInfoPB, and AFPInsRemSMBParam structures each include a pointer to an I/O completion routine that uses register-based parameters under classic 68k and cannot be written in a high-level language without the help of mixed mode or assembly glue.

Summary of AFP

Pascal Summary

Constants

```
CONST                              { PBControl calls }
    afpGetAttnRoutine   = 252;
    dsOpenSession       = 244;
    dsGetStatus         = 243;
    dsSendRequest       = 240;
    dsCloseSession      = 237;
    dsCloseAll          = 232;
    AFPInsSessMemBlk    = 246;
    AFPRemSessMemBlk    = 245
                                   ( PBStatus calls }
    afpGetFSID          = 127,
    afpSVolInfo         = 124,
    afpXGetVolInfo      = 121,
    dsGetXPortInfo      = 236

    SMBSize             =  2560; {size of session block memory}
```

Data Types

Send Request Parameter Block

```
{ csCode = dsSendRequest }
DSWritePBPtr = ^DSWritePB;
DSWritePB = RECORD
dsWriteDataOffset:UInt32;{ Specifies the write offset in the data }
dsWriteBufferSize:UInt32;{ Size of the data to be written }
dsWriteBuffer:Ptr;{ Pointer to data to be written }
END;
```

Get Status Parameter Block

```
{  csCode = dsGetStatus }
    DSGetStatusPBPtr = ^DSGetStatusPB;
    DSGetStatusPB = RECORD
        dsGSSrvrAddress:OTAddressPtr;   { OT Address of server to GetStatus() from }
                                        { (you also need to fill in the reply buffer }
                                        { and size) }
        dsGSEpString:ConstCStringPtr;   { Endpoint string for the connection (nil }
                                        { == default) }
    END;
```

Open Session Parameter Block

```
{  csCode = dsOpenSession }
    DSOpenPBPtr = ^DSOpenPB;
    DSOpenPB = RECORD
        dsOSAttnRoutine:AttnRoutineUPP; { Custom attention routine; nil == default) }
        dsOSSrvrAddress:OTAddressPtr;   { OT Address of server to open a session with }
        dsOSSessionBlock:Ptr;           { Pointer to the SMB reserved for the }
                                        { session }

        dsOSEpString:ConstCStringPtr;   { Endpoint string for the connection;
                                        { (nil == default) }
    END;
```

DSParamBlock Record

```
TYPE
    DSParamBlockPtr = ^DSParamBlock;
    DSParamBlock = RECORD
        qLink:              QElemPtr;       { Reserved }
        qType:              INTEGER;        { Reserved }
        ioTrap:             INTEGER;        { Reserved }
        ioCmdAddr:          Ptr;            { Reserved }
        ioCompletion:       DSIOCompletionUPP;  { Completion routine }
        ioResult:           OSErr;          { Result code }
        cmdResult:          LONGINT;        { Command result }
        ioVRefNum:          INTEGER;        { Reserved }
        ioCRefNum:          INTEGER;        { .AFPTranslator driver reference number }
        csCode:             INTEGER;        { DS Command code }
        dsTimeout:          INTEGER;        { Timeout (AppleTalk only) }
        dsReserved1:        INTEGER;        { Reserved }
        dsRetry:            LONGINT;        { Retry (AppleTalk only) }
        dsReserved2:        UInt16;         { Reserved }
        dsSessRefNum:       INTEGER;        { AFP session number }
        dsReserved3:        INTEGER;        { Reserved }
        dsCmdBufferSize:    INTEGER;        { Size of the command buffer }
        dsCmdBuffer:        Ptr;            { Pointer to the command buffer }
        dsReplyBufferSize:  UInt32;         { Size of the reply buffer }
        dsReplyBuffer:Ptr;                  { Pointer to the reply buffer }
        CASE INTEGER OF
        0: (
            open:           DSOpenPB;
            );
        1: (
            write:          DSWritePB;
            );
        2: (
            status:         DSGetStatusPB;
            );
    END;

{ definitions for dsXPortType }
CONST
    kASPXport          = $00;
    kTCPXport          = $01;
```

GetVolSessInfoRec Record

```
TYPE
    GetVolSessInfoRecPtr = ^GetVolSessInfoRec;
    GetVolSessInfoRec = RECORD
        sessAFPVersion:         INTEGER;        { AFP version number }
        sessReferenceNumber:    INTEGER;        { AFP session reference number }
        sessAFPVolID:           INTEGER;        { AFP volume identifier }
        sessServerAddress:      OTAddressPtr;   { Server internet address }
        sessUAMType:            INTEGER;        { User authentication method }
        sessUserNamePtr:        StringPtr;      { Pointer to user name string }
        sessVolIconPtr:         Ptr;            { Pointer to server volume icon/mask }
        sessWhereStringPtr:     StringPtr;      { Pointer to "where" information
                                                { string shown in the Get Info window }
    END;

CONST
    kAFPVersion11   = 1;
    kAFPVersion20   = 2;
    kAFPVersion21   = 3;
    kAFPVersion22   = 4;

    kNoUserAuth     = 1;        { 'No User Authent' UAM (Guest) }

    kCleartextAuth  = 2;        { 'Cleartxt Passwrd' UAM (types 2 & 3 will be
                                { automatically upgraded to 6) }
    kRandnumAuth    = 3;        { 'Randnum Exchange' UAM  }
    k2WayRandnumAuth= 6;        { '2-Way Randnum exchange'  }
    kMinCustomUAM   = 128;      { Minimum type value for a custom UAM }
```

GetVolSessInfoPB Record

```
TYPE
    GetVolSessInfoPBPtr = ^GetVolSessInfoPB;
    GetVolSessInfoPB = RECORD
        qLink:          QElemPtr;           { Standard header information }
        qType:          INTEGER;            { Standard header information }
        ioTrap:         INTEGER;            { Standard header information }
        ioCmdAddr:      Ptr;                { Standard header information }
        ioCompletion:   IOCompletionUPP;    { Completion routine pointer }
        ioResult:       OSErr;              { Result from async call }
        ioNamePtr:      StringPtr;          { Standard header information }
        ioVRefNum:      INTEGER;            { Standard header information }
        ioRefNum:       INTEGER;            { RefNum of the ".AFPTranslator" }
        csCode:         INTEGER;            { Always afpSVolInfo }
        vcbPtr:         Ptr;                { Pointer to the VCB that you want
                                            { information about }
        sessInfoBuffer: GetVolSessInfoRecPtr;   { Pointer to the GetVolSessInfoRec to
                                            { be filled }
        sessInfoSize:LONGINT;               { Size of the GetVolSessInfoRec }
        actSessInfoSize:LONGINT;            { Actual size of the data returned }
    END;
```

AFPInsRemSMBParam Record

```
{ The AFPInsSessMemBlk and AFPRemSessMemBlk calls are currently
  required when opening or closing a session. Make the AFPInsSessMemBlk call after the
  dsOpenSession call succeeds (or returns afpAuthContinue), with the same
  dsOSSessionBlock that you sent into dsOpenSession. You need to call
  AFPRemSessMemBlk with that same pointer after calling dsCloseSession or dsCloseAll.
}

    AFPInsRemSMBParamPtr = ^AFPInsRemSMBParam;
    AFPInsRemSMBParam = RECORD
        qLink:          QElemPtr;           { Standard header information }
        qType:          INTEGER;            { Standard header information }
        ioTrap:         INTEGER;            { Standard header information }
        ioCmdAddr:      Ptr;                { Standard header information }
        ioCompletion:   IOCompletionUPP;    { Completion rtn pointer }
        ioResult:       OSErr;              { Result from Async call }
        ioNamePtr:      StringPtr;          { Standard header information }
        ioVRefNum:      INTEGER;            { Standard header information }
        ioRefNum:       INTEGER;            { RefNum of the ".AFPTranslator" }

        csCode:         INTEGER;            { AFPInsSessMemBlk or AFPRemSessMemBlk  }
        smbPtr:         Ptr;                { Pointer to the SMB to insert or remove }
    END;

    AFPInsRemSMBPBPtr   = ^AFPInsRemSMBParam;
```

AFPSrvrInfo Record

```
{ Server Info Buffer returned from the dsGetStatus call }
{ you should make your buffer at least 1024 bytes in size.}
{ a partial definition of the AFPSrvrInfo data structure (the fixed portion) }

    AFPSrvrInfoPtr = ^AFPSrvrInfo;
    AFPSrvrInfo = RECORD
        fMachineOffset:     INTEGER;
        fVerCountOffset:    INTEGER;
        fUAMCountOffset:    INTEGER;
        fIconOffset:        INTEGER;
        fFlags:             INTEGER;
        fSrvrName:          PACKED ARRAY [0..1] OF UInt8;
    END;

{ definitions for the fFlags word}
CONST
    srvSCopyFile        = 0;    { Server supports FPCopyFile call }
    srvSChangePswd      = 1;    { Server supports FPChangePassword call }
    srvNoPswdSave       = 2;    { Workstation should not save password }
    srvSServerMsgs      = 3;    { Server supports server messages }
    srvSSrvrSig         = 4;    { Server supports Server Signatures  (AFP 2.2) }
    srvSupportsTCP      = 5;    { Server may be connected to via TCP/IP (AFP 2.2) }
    srvSNotification    = 6;    { Server will send notifications (AFP 2.2) }
```

Gestalt Selectors and Definitions

```
gestaltAFPClient            = 'afps';
    gestaltAFPClientVersionMask = $0000FFFF;    {  low word is version }
    gestaltAFPClient3_5         = 1;
    gestaltAFPClient3_6         = 2;
    gestaltAFPClient3_6_1       = 3;
    gestaltAFPClient3_6_2       = 4;
    gestaltAFPClient3_6_3       = 5;            {  including 3.6.4, 3.6.5 }
    gestaltAFPClient3_7         = 6;            {  including 3.7.1 }
    gestaltAFPClient3_7_2       = 7;            {  including 3.7.3 }
    gestaltAFPClient3_8         = 8;
    gestaltAFPClientCfgMask     = $FFFF0000;    { high word is config }
    gestaltAFPClientCfgRsrc     = 16;           { Client uses config resources }
    gestaltAFPClientUAMv2       = 28;   { Client supports the 2.0 UAM interfaces }
    gestaltAFPClientSupportsIP  = 29;   { Client supports AFP over TCP/IP }
    gestaltAFPClientVMUI        = 30;   { Client can put up UI from the PBVolMount }
                                        { trap }
    gestaltAFPClientMultiReq    = 31;   { Client supports multiple outstanding
                                        { requests }
```

Routines

```
FUNCTION NAFPCommandASync(paramBlock: DSParamBlockPtr): OSErr;

FUNCTION NAFPCommandImmediate(paramBlock: DSParamBlockPtr): OSErr;

FUNCTION NAFPCommandSync(paramBlock: DSParamBlockPtr): OSErr;
```

I/O Completion Routine

```
TYPE
    DSIOCompletionProcPtr = Register68kProcPtr;
    { PROCEDURE DSIOCompletion(pb: UNIV Ptr); }

    DSIOCompletionUPP = UniversalProcPtr;
```

C Summary

Constants

```
enum {                              /* PBControl calls */
    afpGetAttnRoutine   = 252,
    dsOpenSession       = 244,
    dsGetStatus         = 243,
    dsSendRequest       = 240,
    dsCloseSession      = 237,
    dsCloseAll          = 232,
    AFPInsSessMemBlk    = 246,
    AFPRemSessMemBlk    = 245
};


enum {                          /* PBStatus calls */
    afpGetFSID          = 127,
    afpSVolInfo         = 124,
    afpXGetVolInfo      = 121,
    dsGetXPortInfo      = 236
};

enum {
    SMBSize = 2560  /* size of the session memory block*/
};
```

Data Types

Send Request Parameter Block Structure

```
/* csCode = dsSendRequest*/
struct DSWritePB {
    UInt32      dsWriteDataOffset;/* Specifies the write offset in the data */
    UInt32      dsWriteBufferSize;/* Size of the data to be written */
    Byte *      dsWriteBuffer;  /* Pointer to data to be written */

};
typedef struct DSWritePB DSWritePB;
```

Get Status Parameter Block Structure

```
/* csCode = dsGetStatus*/
struct DSGetStatusPB {
    OTAddress * dsGSSrvrAddress;    /* OT address of server to GetStatus() from */
                                    /* (you also need to fill in the reply buffer */
                                    /* and size) */
    const char *    dsGSEpString;   /* Endpoint string for the connection;
                                    /* (nil == default) */
};
typedef struct DSGetStatusPB DSGetStatusPB;
```

Open Session Parameter Block

```
/* csCode = dsOpenSession*/
struct DSOpenPB {
    AttnRoutineUPP  dsOSAttnRoutine;    /* Custom attention routine (nil == default) */
    OTAddress *     dsOSSrvrAddress;    /* OT address of server to open a session to */
    Ptr             dsOSSessionBlock;   /* Pointer to the SMB reserved for */
                                        /* the session */
    const char *    dsOSEpString;       /* Endpoint string for the connection: */
                                        /* (nil == default) */
};
typedef struct DSOpenPB DSOpenPB;

enum {
    SMBSize     = 2560                  /* Size of the session memory block */
};
```

DSParamBlock Structure

```
struct DSParamBlock {
    QElem *             qLink;              /* Reserved */
    short               qType;              /* Reserved */
    short               ioTrap;             /* Reserved */
    Ptr                 ioCmdAddr;          /* Reserved */
    DSIOCompletionUPP   ioCompletion;       /* Completion routine */
    OSErr               ioResult;           /* Result code */
    long                cmdResult;          /* Command result */
    short               ioVRefNum;          /* Reserved */
    short               ioCRefNum;          /* .AFPTranslator driver reference number*/
    short               csCode;             /* DS Command code */
    short               dsTimeout;          /* Timeout (AppleTalk only) */
    short               dsReserved1;        /* Reserved */
    long                dsRetry;            /* Retry count (AppleTalk only) */
    UInt16              dsReserved2;        /* Reserved */
    short               dsSessRefNum;       /* AFP session number*/
    short               dsReserved3;        /* Reserved */
    short               dsCmdBufferSize;    /* Size of the command buffer */
    UInt8 *             dsCmdBuffer;        /* Pointer to the command buffer */
    UInt32              dsReplyBufferSize;  /* Size of the reply buffer */
    UInt8 *             dsReplyBuffer;      /* Pointer to the reply buffer */
    union {
        DSOpenPB open;                      /* csCode is dsOpenSession */
        DSWritePB write;
        DSGetStatusPB status;               /* csCode is dsGetStatus */
    }   csParam;
};
typedef struct DSParamBlock DSParamBlock;
typedef DSParamBlock *DSParamBlockPtr;
```

DSXPortInfo Structure

```
struct DSXPortInfo {
    long            dsXPortType;        /* Transport type (kASPXport, kTCPXport) */
    short           dsXPortSessRefNum;  /* Session reference number for ASP or TCP */
    union {
        InetAddress ipAddr;
        DDPAddress  ddpAddr;
    } addr;
};
typedef struct DSXPortInfo DSXPortInfo;
typedef DSXPortInfo *DSXPortInfoPtr;

/* definitions for dsXPortType */
enum {
    kASPXport           = 0x00,
    kTCPXport           = 0x01
};
```

GetVolSessInfoRec Structure

```
struct GetVolSessInfoRec {
    short           sessAFPVersion;         /* AFP version number */
    short           sessReferenceNumber;    /* AFP session reference number */
    short           sessAFPVolID;           /* AFP volume identifier */
    OTAddress *     sessServerAddress;      /* Server internet address */
    short           sessUAMType;            /* User authentication method */
    StringPtr       sessUserNamePtr;        /* Pointer to user name string */
    Ptr             sessVolIconPtr;         /* Pointer to server volume icon/mask */
    StringPtr       sessWhereStringPtr;     /* Pointer to "where" information */
                                            /* string shown in the Get Info window */
};
typedef struct GetVolSessInfoRec GetVolSessInfoRec;
typedef GetVolSessInfoRec *GetVolSessInfoRecPtr;

enum {
    kAFPVersion11   = 1,
    kAFPVersion20   = 2,
    kAFPVersion21   = 3,
    kAFPVersion22   = 4
};

enum {
    kNoUserAuth         = 1,    /* 'No User Authent' UAM (Guest)*/
    kCleartextAuth      = 2,    /* 'Cleartxt Passwrd' UAM (types 2 & 3 will be */
                                /* automatically upgraded to 6)*/
    kRandnumAuth        = 3,    /* 'Randnum Exchange' UAM */
    k2WayRandnumAuth    = 6,    /* '2-Way Randnum exchange' */
    kMinCustomUAM       = 128   /* Minimum type value for a Custom UAM*/
};
```

GetVolSessInfoPB Structure

```
struct GetVolSessInfoPB {
    QElemPtr            qLink;              /* Standard header information */
    short               qType;              /* Standard header information */
    short               ioTrap;             /* Standard header information */
    Ptr                 ioCmdAddr;          /* Standard header information */
    IOCompletionUPP     ioCompletion;       /* Completion routine pointer */
    OSErr               ioResult;           /* Result from Async call */
    StringPtr           ioNamePtr;          /* Standard header information */
    short               ioVRefNum;          /* Standard header information */
    short               ioRefNum;           /* RefNum of the ".AFPTranslator" */
    short               csCode;             /* Always afpSVolInfo */
    Ptr                 vcbPtr;             /* Pointer to the VCB that you want */
                                            /* information about */
    GetVolSessInfoRecPtr sessInfoBuffer;    /* Pointer to the GetVolSessInfoRec to */
                                            /* be filled */
    long                sessInfoSize;       /* Size of the GetVolSessInfoRec */
    long                actSessInfoSize;    /* Actual size of the data returned */
};
typedef struct GetVolSessInfoPB GetVolSessInfoPB;
typedef GetVolSessInfoPB *GetVolSessInfoPBPtr;

/* the AFPInsSessMemBlk & AFPRemSessMemBlk calls are currently (pre Client 3.8)
   required when opening or closing a session. Make the AFPInsSessMemBlk call after
   the dsOpenSession call succeeds (or returns afpAuthContinue), with the same
   dsOSSessionBlock that you sent into dsOpenSession. You need to call
   AFPRemSessMemBlk with that same pointer after calling dsCloseSession or dsCloseAll.
   In Client 3.8 these will be called for you during the dsOpenSession &
   dsCloseSession calls.
*/
```

AFPInsRemSMBParam Structure

```
struct AFPInsRemSMBParam {
    QElemPtr        qLink;          /* Standard header information */
    short           qType;          /* Standard header information */
    short           ioTrap;         /* Standard header information */
    Ptr             ioCmdAddr;      /* Standard header information */
    IOCompletionUPP ioCompletion;   /* Completion routine pointer*/
    OSErr           ioResult;       /* Result from Async call*/
    StringPtr       ioNamePtr;      /* Standard header information */
    short           ioVRefNum;      /* Standard header information */
    short           ioRefNum;       /* RefNum of the ".AFPTranslator" */
    short           csCode;         /* AFPInsSessMemBlk or AFPRemSessMemBlk */
    Ptr             smbPtr;         /* Pointer to the SMB to insert or remove */
};
typedef struct AFPInsRemSMBParam AFPInsRemSMBParam;
typedef AFPInsRemSMBParam *AFPInsRemSMBPBPtr;
```

AFPSrvrInfo Structure

```
/* Server Info Buffer returned from the dsGetStatus call */
/* you should make your buffer at least 1024 bytes in size.*/
/* a partial definition of the AFPSrvrInfo data structure (the fixed portion) */

struct AFPSrvrInfo {
    short           fMachineOffset;
    short           fVerCountOffset;
    short           fUAMCountOffset;
    short           fIconOffset;
    short           fFlags;
    unsigned char   fSrvrName[2];
};
typedef struct AFPSrvrInfo AFPSrvrInfo;

/* Definitions for the fFlags word*/
enum {
    srvSCopyFile        = 0,    /* Server supports FPCopyFile call */
    srvSChangePswd      = 1,    /* Server supports FPChangePassword call */
    srvNoPswdSave       = 2,    /* Workstation should not save password */
    srvSServerMsgs      = 3,    /* Server supports server messages */
    srvSSrvrSig         = 4,    /* Server supports Server Signatures  (AFP 2.2) */
    srvSupportsTCP      = 5,    /* Server may be connected to via TCP/IP (AFP 2.2) */
    srvSNotification    = 6     /* Server will send notifications (AFP 2.2) */
};
```

Gestalt Selectors and Definitions

```
enum {
    gestaltAFPClient            = FOUR_CHAR_CODE('afps'),
    gestaltAFPClientVersionMask = 0x0000FFFF,       /* Low word is version*/
    gestaltAFPClient3_5         = 1,
    gestaltAFPClient3_6         = 2,
    gestaltAFPClient3_6_1       = 3,
    gestaltAFPClient3_6_2       = 4,
    gestaltAFPClient3_6_3       = 5,    /* Including 3.6.4, 3.6.5 */
    gestaltAFPClient3_7         = 6,    /* Including 3.7.1 */
    gestaltAFPClient3_7_2       = 7,    /* Including 3.7.3 */
    gestaltAFPClient3_8         = 8,
    gestaltAFPClientCfgMask     = (long)0xFFFF0000,/* high word is config */
    gestaltAFPClientCfgRsrc     = 16,   /* Client uses config resources */
    gestaltAFPClientUAMv2       = 28,   /* Client supports the 2.0 UAM interfaces */
    gestaltAFPClientSupportsIP  = 29,   /* Client supports AFP over TCP/IP */
    gestaltAFPClientVMUI        = 30,   /* Client can put up UI from the */
                                        /* PBVolMount trap */
    gestaltAFPClientMultiReq    = 31    /* Client supports multiple outstanding */
                                        /* requests */
};
```

Routines

```
OSErr NAFPCommandAsync(DSParamBlockPtr paramBlock);

OSErr NAFPCommandImmediate(DSParamBlockPtr paramBlock);

OSErr NAFPCommandSync(DSParamBlockPtr paramBlock);
```

I/O Completion Routine

```
typedef CALLBACK_API( void , DSIOCompletionProcPtr )(void *pb);

/*
    WARNING: DSIOCompletionProcPtr uses register based parameters under classic 68k
             and cannot be written in a high-level language without
             the help of mixed mode or assembly glue.
*/

typedef REGISTER_UPP_TYPE(DSIOCompletionProcPtr) DSIOCompletionUPP;
```

Assembly-Language Summary

Constants

DS Control Codes

```
afpXGetVolInfo      EQU     121
afpSVolInfo         EQU     124
afpGetFSID          EQU     127
dsCloseAll          EQU     232
dsGetXPortInfo      EQU     236
dsCloseSession      EQU     237
dsSendRequest       EQU     240
dsGetStatus         EQU     243
dsOpenSession       EQU     244
```

```
AFPRemSessMemBlk    EQU     245
AFPInsSessMemBlk    EQU     246
afpGetAttnRoutine   EQU     252
```

Miscellaneous

```
SMBSize             EQU         2650    ;size of the session memory block
```

Data Structures

Send Request Parameter Block

```
; csCode = dsSendRequest
DSWritePB       RECORD 0
dsWriteDataOffset   ds.l    1   ; offset: $0 (0); Specifies the write offset in the
                                                ; data
dsWriteBufferSize   ds.l    1   ; offset: $4 (4); Size of the data to be written
dsWriteBuffer       ds.l    1   ; offset: $8 (8); Pointer to data to be written
sizeof              EQU *       ; size:   $C (12)
ENDR
```

Get Status Parameter Block

```
;  csCode = dsGetStatus
DSGetStatus     PB          RECORD 0
dsGSSrvrAddress     ds.l    1   ; offset: $0 (0); OT address of server to GetStatus()
                                ; from (you also need to fill in the reply buffer and
                                ; size)
dsGSEpString        ds.l    1   ; offset: $4 (4); Endpoint string for the connection
                                ; (nil == default)
sizeof              EQU *       ; size:   $8 (8)
ENDR
```

Open Session Parameter Block

```
; csCode = dsOpenSession
DSOpenPB        RECORD 0
dsOSAttnRoutine     ds.     1   ; offset: $0 (0)    ; Custom attention routine
                                                    ; (nil == default)
dsOSSrvrAddress     ds.l    1   ; offset: $4 (4)    ; OT address of server to open a
                                                    ; session to
dsOSSessionBlock    ds.l    1   ; offset: $8 (8)    ; Pointer to the SMB reserved
                                                    ; for the session
dsOSEpString        ds.l    1   ; offset: $C (12)   ; Endpoint string for the
                                                    ; connection (nil == default)
sizeof               EQU *              ; size:   $10 (16)
```

DSParamBlock Parameter Block

```
DSParamBlock        RECORD 0
qLink               ds.l    ; offset: $0 (0)    ; Standard header information
qType               ds.w    ; offset: $4 (4)    ; Standard header information
ioTrap              ds.w    ; offset: $6 (6)    ; Standard header information
ioCmdAddr           ds.l    ; offset: $8 (8)    ; Standard header information
ioCompletion        ds.l    ; offset: $C (12)   ; Completion routine pointer
ioResult            ds.w    ; offset: $10 (16)  ; Result from Async call
cmdResult           ds.l    ; offset: $12 (18)  ; Result from the server for the AFP
                                                ; command
ioVRefNum           ds.w    ; offset: $16 (22)  ; Standard header information
ioCRefNum           ds.w    ; offset: $18 (24)  ; RefNum of the ".AFPTranslator"
csCode              ds.w    ; offset: $1A (26)  ; DS command code
dsTimeout           ds.w    ; offset: $1C (28)  ; Reserved for TCP; for ASP, how long
                                                ; to wait before retrying the request
dsReserved1         ds.w    ; offset: $1E (30)  ; Reserved
dsRetry             ds.l    ; offset: $20 (32)  ; Unused for TCP; for ASP, how many
                                                ; times to retry the request
dsReserved2         ds.w    ; offset: $24 (36)  ; Reserved
dsSessRefNum        ds.w    ; offset: $26 (38)  ; AFP session number
dsReserved3         ds.w    ; offset: $28 (40)  ; Reserverd
dsCmdBufferSize     ds.w    ; offset: $2A (42)  ; Size of the command buffer
dsCmdBuffer         ds.l    ; offset: $2C (44)  ; Pointer to the command buffer
dsReplyBufferSize   ds.l    ; offset: $30 (48)  ; Size of the reply buffer
dsReplyBuffer       ds.l    ; offset: $34 (52)  ; Pointer to the reply buffer
open                ds      DSOpenPB; offset: $38 (56)      ORG 56
```

```
write               ds      DSWritePB; offset: $38 (56)     ORG 56
status              ds      DSGetStatusPB; offset: $38 (56) ORG 72
sizeof              EQU *   ; size:   $48 (72)
ENDR
```

DSXPortInfo Record

```
DSXPortInfo     RECORD 0
dsXPortType         ds.l    1   ; offset: $0 (0); Transport Type (kASPXport,
                                                ; kTCPXport)
dsXPortSessRefNum   ds.w    1   ; offset    : $4 (4); Session ref number for ASP or TCP
ipAddr              ds      InetAddress     ; offset: $6 (6)
                    ORG 6
ddpAddr             ds      DDPAddress      ; offset: $6 (6)
                    ORG 22
sizeof              EQU *               ; size:   $16 (22)
ENDR

; definitions for dsXPortType
kASPXport               EQU     $00
kTCPXport               EQU     $01
```

GetVolSessInfo Record

```
GetVolSessInfoRecRECORD 0
sessAFPVersion      ds.w    1   ; offset: $0 (0)    ; AFP version number:
sessReferenceNumber ds.w    1   ; offset: $2 (2)    ; AFP session reference number
sessAFPVolID        ds.w    1   ; offset: $4 (4)    ; AFP volume identifier
sessServerAddress   ds.l    1   ; offset: $6 (6)    ; Server internet address
sessUAMType         ds.w    1   ; offset: $A (10)   ; User authentication method
sessUserNamePtr     ds.l    1   ; offset: $C (12)   ; Pointer to user name string
sessVolIconPtr      ds.l    1   ; offset: $10 (16)  ; Pointer to server volume icon/
                                                    ; mask
sessWhereStringPtr  ds.l1       ; offset: $14 (20)  ; ptr to "where" information
                                                    ; string shown in the Get Info
                                                    ; window
sizeof               EQU *      ; size:   $18 (24)
ENDR
```

```
kAFPVersion11       EQU     1
kAFPVersion20       EQU     2
kAFPVersion21       EQU     3
kAFPVersion22       EQU     4
kNoUserAuth         EQU     1   ; 'No User Authent' UAM (Guest)
kCleartextAuth      EQU     2   ; 'Cleartxt Passwrd' UAM (types 2 & 3 will be
                                ; automatically upgraded to 6)
kRandnumAuth        EQU     3   ; 'Randnum Exchange' UAM
k2WayRandnumAuth    EQU     6   ; '2-Way Randnum exchange'
kMinCustomUAM       EQU     128 ; Minimum type value for a Custom UAM
```

GetVolSessionPB Record

```
GetVolSessInfoPBRECORD 0
qLink           ds.l    1   ; offset: $0 (0)    ; Standard header stuff
qType           ds.w    1   ; offset: $4 (4)    ; Standard header stuff
ioTrap          ds.w    1   ; offset: $6 (6)    ; Standard header stuff
ioCmdAddr       ds.l    1   ; offset: $8 (8)    ; Standard header stuff
ioCompletion    ds.l    1   ; offset: $C (12)   ; Completion rtn pointer
ioResult        ds.w    1   ; offset: $10 (16)  ; Result from Async call
ioNamePtr       ds.l    1   ; offset: $12 (18)  ; Standard header stuff
ioVRefNum       ds.w    1   ; offset: $16 (22)  ; Standard header stuff
ioRefNum        ds.w    1   ; offset: $18 (24)  ; RefNum of the ".AFPTranslator"
csCode          ds.w    1   ; offset: $1A (26)  ; Always afpSVolInfo
vcbPtr          ds.l    1   ; offset: $1C (28)  ; Pointer to the VCB that you want info
                                                ; about
sessInfoBuffer  ds.l    1   ; offset: $20 (32)  ; Pointer to the GetVolSessInfoRec to
                                                ; be filled
sessInfoSize    ds.l    1   ; offset: $24 (36)  ; Size of the GetVolSessInfoRec
actSessInfoSize ds.l    1   ; offset: $28 (40)  ; Actual size of the data returned
sizeof          EQU *       ; size:   $2C (44)
ENDR
```

AFPInsRemSMBParam Record

```
; The AFPInsSessMemBlk & AFPRemSessMemBlk calls are currently
; required when opening or closing a session. Make the AFPInsSessMemBlk call after the
; dsOpenSession call succeeds (or returns afpAuthContinue), with the same
; dsOSSessionBlock that you sent into dsOpenSession. You need to call AFPRemSessMemBlk
; with that same pointer after calling dsCloseSession or dsCloseAll.
```

```
AFPInsRemSMBParamRECORD 0
qLink           ds.l    1   ; offset: $0 (0)    ; Standard header stuff
qType           ds.w    1   ; offset: $4 (4)    ; Standard header stuff
ioTrap          ds.w    1   ; offset: $6 (6)    ; Standard header stuff
ioCmdAddr       ds.l    1   ; offset: $8 (8)    ; Standard header stuff
ioCompletion    ds.l    1   ; offset: $C (12)   ; Completion rtn pointer
ioResult        ds.w    1   ; offset: $10 (16)  ; Result from Async call
ioNamePtr       ds.l    1   ; offset: $12 (18)  ; Standard header stuff
ioVRefNum       ds.w    1   ; offset: $16 (22)  ; Standard header stuff
ioRefNum        ds.w    1   ; offset: $18 (24)  ; RefNum of the ".AFPTranslator"
csCode          ds.w    1   ; offset: $1A (26)  ; AFPInsSessMemBlk or AFPRemSessMemBlk
smbPtr          ds.l    1   ; offset: $1C (28)  ; Pointer to the SMB to insert or
                                                ; remove
sizeof          EQU *       ; size:   $20 (32)
ENDR
```

AFPSrvrInfo Record

```
; Server Info Buffer returned from the dsGetStatus call
; you should make your buffer at least 1024 bytes in size.
; a partial definition of the AFPSrvrInfo data structure (the fixed portion)

AFPSrvrInfo     RECORD 0
fMachineOffset  ds.w    1   ; offset: $0 (0)
fVerCountOffset ds.w    1   ; offset: $2 (2)
fUAMCountOffset ds.w    1   ; offset: $4 (4)
fIconOffset     ds.w    1   ; offset: $6 (6)
fFlags          ds.w    1   ; offset: $8 (8)
fSrvrName       ds.b    2   ; offset: $A (10)
sizeof          EQU *       ; size:   $C (12)
ENDR

;  definitions for the fFlags word
srvSCopyFile        EQU     0   ; Server supports FPCopyFile call
srvSChangePswd      EQU     1   ; Server supports FPChangePassword call
srvNoPswdSave       EQU     2   ; Workstation should not save password
srvSServerMsgs      EQU     3   ; Server supports server message
srvSSrvrSig         EQU     4   ; Server supports Server Signatures  (AFP 2.2)
srvSupportsTCP      EQU     5   ; Server may be connected to via TCPIP (AFP 2.2)
srvSNotificationEQU     6       ; Server will send notifications (AFP 2.2)
```

Gestalt Selectors and Definitions

```
gestaltAFPClient            EQU    'afps'
gestaltAFPClientVersionMask EQU    $0000FFFF   ; Low word is version
gestaltAFPClient3_5         EQU    1
gestaltAFPClient3_6         EQU    2
gestaltAFPClient3_6_1       EQU    3
gestaltAFPClient3_6_2       EQU    4
gestaltAFPClient3_6_3       EQU    5           ; Including 3.6.4, 3.6.5
gestaltAFPClient3_7         EQU    6           ; Including 3.7.1
gestaltAFPClient3_7_2       EQU    7           ; Including 3.7.3
gestaltAFPClient3_8         EQU    8
gestaltAFPClientCfgMask     EQU    $FFFF0000   ; High word is config
gestaltAFPClientCfgRsrc     EQU    16          ; Client uses config resources
gestaltAFPClientUAMv2       EQU    28          ; Client supports 2.0 UAM interfaces
gestaltAFPClientSupportsIP  EQU    29          ; Client supports AFP over TCP/IP
gestaltAFPClientVMUI        EQU    30  ; Client can put up UI from the PBVolMount trap
gestaltAFPClientMultiReq    EQU    31  ; Client supports multiple outstanding requests
```

Result Codes

`kFPAccessDenied`	–5000	The client does not have permission to perform the specified command.
`kFPAuthContinue`	–5001	The client should perform the next authentication step.
`kFPBadUAM`	–5002	The specified UAM does not exist.
`kFPBadVersNum`	–5003	The server does not support the specified version number.
`kFPBitmapErr`	–5004	The specified bitmap specifies an invalid value.
`kFPCantMoveErr`	–5005	Can’t move a file or folder from one directory to another. Superuser trying to move one share point into another.
`kFPDenyConflict`	–5006	User opens file and denies write, another opens to write.
`kFPDirNotEmpty`	–5007	The command to remove a directory could not be completed because the directory is not empty.
`kFPDiskFull`	–5008	The command could not be completed because the disk is full.

`kFPEOFErr`	–5009	Returned by the `FPCatSearch` command when there are no more matches.
`kFPFileBusy`	–5010	The specified file is in use.
`kFPFlatVol`	–5011	Obsolete as of AppleShare IP 6.0.
`kFPItemNotFound`	–5012	The specified file or directory could not be found.
`kFPLockErr`	–5013	The specified file is locked.
`kFPMiscErr`	–5014	A unspecified error occurred.
`kFPNoMoreLocks`	–5015	All of the available locks are in use.
`kFPNoServer`	–5016	There is not a server at the specified server address, or the server did not respond to the request.
`kFPObjectExists`	–5017	The specified object already exists.
`kFPObjectNotFound`	–5018	The specified object could not be found.
`kFPParamErr`	–5019	The parameter block contains data that is invalid for the specified AFP command.
`kFPRangeNotLocked`	–5020	The specified range could not be locked.
`kFPRangeOverLap`	–5021	The specified range contains overlapping values.
`kFPSessClosed`	–5022	The specified command could not be completed because the session is closed.
`kFPUserNotAuth`	–5023	The command could not be performed because the client has not been authenticated.
`kFPCallNotSupported`	–5024	The specified command is not supported.
`kFPObjectTypeErr`	–5025	The specified object is the wrong type.
`kFPTooManyFilesOpen`	–5026	The specified command could not be completed because too many files are open.
`kFPServerGoingDown`	–5027	The server is shutting down.
`kFPCantRename`	–5028	The specified file or directory cannot be renamed.
`kFPDirNotFound`	–5029	The specified directory could not be found.
`kFPIconTypeErr`	–5030	The specified icon is of the wrong type.
`kFPVolLocked`	–5031	The specified command could not be completed because the volume is locked.
`kFPObjectLocked`	–5032	The specified command could not be completed because the object is locked.
`kFPContainsSharedErr`	–5033	The specified share point contains a share point.
`kFPIDNotFound`	–5034	The specified ID could not be found.
`kFPIDExists`	–5035	The specified ID already exists.
`kFPDiffVolErr`	–5036	Equivalent to the Mac OS error code.

`kFPCatalogChanged`	–5037	The catalog has changed and *CatPosition* may be invalid. No matches were returned.
`kFPSameObjectErr`	–5038	Equivalent to the Mac OS error code.
`kFPBadIDErr`	–5039	The specified ID is invalid.
`kFPPwdSameErr`	–5040	The new password is the same as the old password.
`kFPPwdTooShortErr`	–5041	The specified password is too short.
`kFPPwdExpiredErr`	–5042	The password has expired.
`kFPInsideSharedErr`	–5043	The specified directory is inside a share point.
`kFPInsideTrashErr`	–5044	The specified directory is in the Trash.
`kFPPwdNeedsChangeErr`	–5045	The password needs to be changed the first time the user logs on.
`kFPPwdPolicyErr`	–5046	The specified password violates a UAM's policy.

Index

A

AFP
- attention routine 17
- command codes 28
- versions 20, 25

AFPInsRemSNBParam structure 18–19
afpLogin command 16
AFPSRVRInfo structure 19–20
afpSVolInfo command 13, 24
.AFPTranslator driver 12
AppleShare Client 3.7 13
AppleTalk Session Protocol 12
AppleTalk sessions
- retries 16
- timeouts 16

ASP 12
asynchronous .AFPTranslator call 13, 32
attention routine 17, 21
authentication 29
authentication, user 25–26

C

closing
- directories 28
- forks 28
- sessions 17

command codes, AFP 28
completion routine 16, 34
control codes, PB 17
copying files 28
creating
- directories 28
- files 28

D

data stream interface 12
Desktop database 31
Device Manager
- DSParamBlock fields 15
- queue 13

directories
- closing 28
- creating 28
- listing contents 28
- renaming 30

driver reference number 15, 16
DSGetStatusPB structure 20–21
DSI 12
DSOpenSessionPB structure 21–22
DSParamBlock structure 14–17
DSWritePB structure 22
DSXPortInfo structure 22–23

E

endpoint string 21
extended volume information 18

F

files
- copying 28
- creating 28
- moving 29
- renaming 30
- setting parameters 30

forks
- closing 28

getting parameters 28
opening 29
functions
NAFPCommandAsync 13, 32
NAFPCommandImmediate 13, 33
NAFPCommandSync 13, 32

G

Gestalt constants, AFP 26
Get Info "where" string 23
getting
fork parameters 28
server information 29
session information 23
status information 17
GetVolSessInfoPB structure 23–24
GetVolSessInfoRec structure 24–25

I

icon, server 20
immediate .AFPTranslator call 13, 33

L

listing contents of directories 28
locking a range 28
logging on 29

M

machine type, obtaining 20
moving files and directories 29

N

NAFPCommandAsync function 13, 32
NAFPCommandImmediate function 13, 33
NAFPCommandSync function 13, 32
native file system 11, 12

O

OpenDriver function 15
opening sessions 17, 18
OT address 21

P

passwords, changing 30
PB control codes 17
PB status codes 18

Q

queues
.AFPTranslator driver 18
Device Manager 13

R

range locking 28
reading data 29
renaming files and directories 30
retries 16

S

sending AFP commands 17
server information, getting 20, 29
session information, getting 23

session memory block
  inserting and removing 17, 18
  opening a session 22
session reference number 13, 16
sessions
  closing 17
  closing all 17
  opening 17, 18
  starting 29
  terminating 29
setting parameters 30
status codes, PB 18
structures
  `AFPInsRemSNBParam` structure 18–19
  `AFPSrvrInfo` structure 19–20
  `DSGetStatusPB` structure 20–21
  `DSOpenSessionPB` structure 21–22
  `DSParamBlock` structure 14–17
  `DSWritePB` structure 22
  `DSXPortInfo` structure 22–23
  `GetVolSessInfoPB` structure 23–24
  `GetVolSessRec` structure 24–25
synchronous .AFPTranslator call 13, 32

T

TCP/IP 12
timeouts 16
Transmission Control Protocol 12
transport type 23

U

UAMs 20, 25
user authentication 25–26

V

volume control block 24
volume data, writing 28
volumes
  closing 28
  getting information 18
  getting parameters 29

W

writing
  buffered data 28
  volume data 28

X

.XPP driver 12, 13