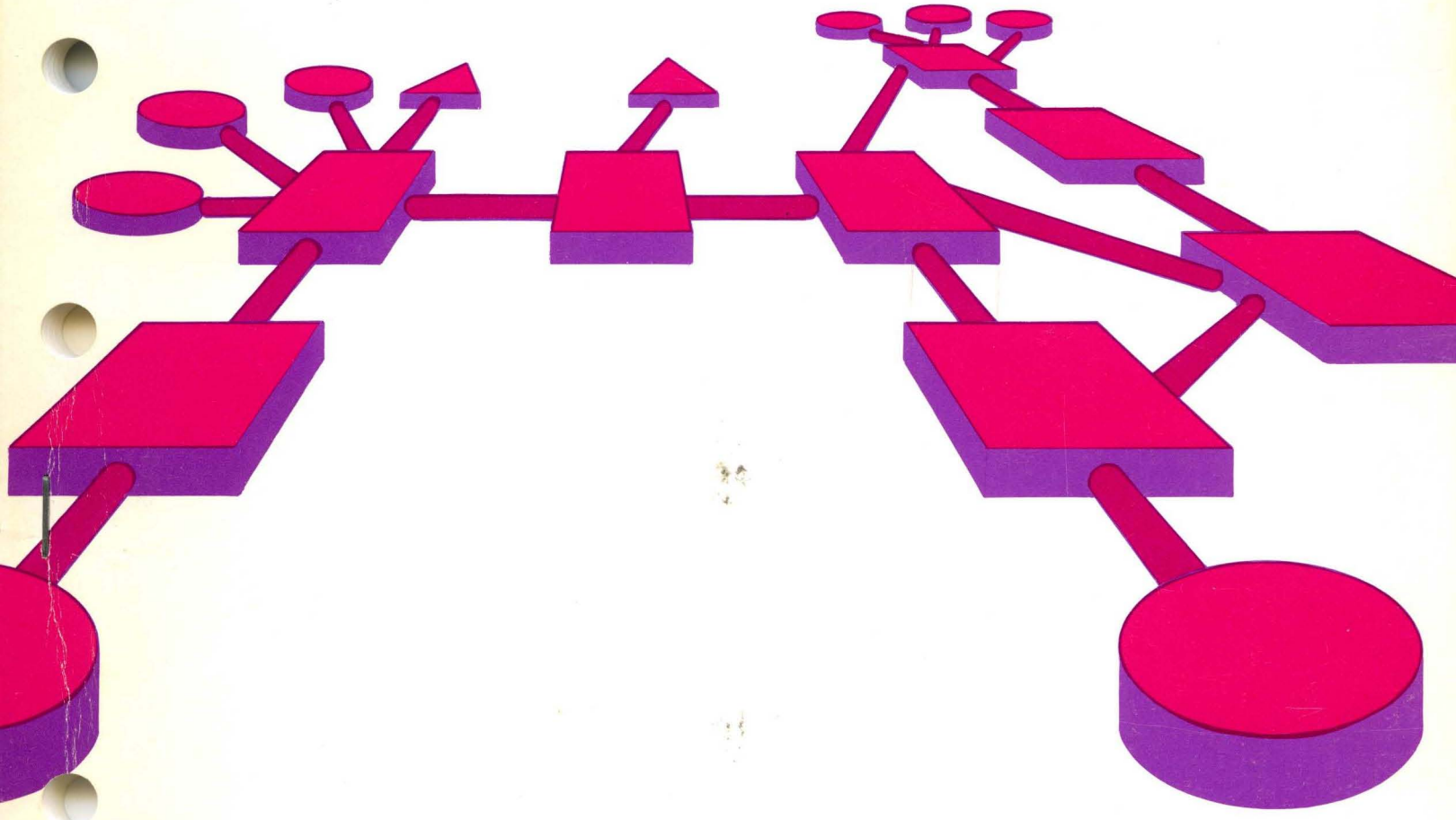
Order No. AA-K177A-TK

DECnet
DIGITAL Network Architecture

Data Access Protocol
Functional Specification
(DAP)

Version 5.6.0



DECnet
DIGITAL Network Architecture
(Phase III)

Data Access Protocol
Functional Specification
(DAP)

Order No. AA-K177A-TK
Version 5.6.0

October 1980

This document describes the features, message formats, and operation of the Data Access Protocol (DAP). DAP provides standardized formats and procedures for accessing and passing data between a user process and a file system existing in a network environment. It assumes a controlled conversation path provided by the network system. In DECnet, this path is created by the Session Control and Network Services Protocols and their associated interfaces.

To order additional copies of this document, contact your local Digital Equipment Corporation Sales Office.

digital equipment corporation · maynard, massachusetts

First Printing, October 1980

This material may be copied, in whole or in part, provided that the copyright notice below is included in each copy along with an acknowledgment that the copy describes the Data Access Protocol developed by Digital Equipment Corporation.

This material may be changed without notice by Digital Equipment Corporation, and Digital Equipment Corporation is not responsible for any errors which may appear herein.

Copyright (C) 1980 by Digital Equipment Corporation

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10
VAX	VMS	SBI
DECnet	IAS	PDT
DATATRIEVE	TRAX	

CONTENTS

		Page
1.0	INTRODUCTION	1
2.0	FUNCTIONAL DESCRIPTION	2
2.1	DAP Functions	2
2.2	Relationship to the DIGITAL Network Architecture	2
2.3	Generic Model	4
3.0	MESSAGE FORMATS	6
3.1	Notation	6
3.2	General Message Format	7
3.3	Configuration Message	11
3.4	Attributes Message	14
3.5	Access Message	21
3.6	Control Message	25
3.7	Continue Transfer Message	29
3.8	Acknowledge Message	29
3.9	Access Complete Message	29
3.10	Data Message	31
3.11	Status Message	31
3.12	Key Definition Attributes Extension Message	45
3.13	Allocation Attributes Extension Message	46
3.14	Summary Attributes Extension Message	48
3.15	Date and Time Attributes Extension Message	48
3.16	Protection Attributes Extension Message	49
3.17	Name Message	51
3.18	Access Control List Attributes Extension Message	51
4.0	FILE ORGANIZATION	52
4.1	Types of Files	52
4.2	Record Formats and Attributes	52
4.2.1	Handling Stream ASCII	53
4.2.2	Conversions	53
4.3	Data Formats	54
4.3.1	Fixed-Length Records	54
4.3.2	Variable-Length Records	54
4.3.3	Variable with Fixed-Control Format Records	54
4.3.4	ASCII Stream	55
4.4	Supported Data Types	55
4.4.1	ASCII	55
4.4.2	EBCDIC	55
4.4.3	Image	55
4.4.4	Compression	56
5.0	OPERATION	58
5.1	Setting up the Link	58
5.1.1	Errors in the Setup Sequence	60
5.1.2	Setup Sequence	61
5.2	Transferring Data over the Link	67
5.2.1	Sequential File Retrieval	68
5.2.2	Sequential File Storage/Append	70
5.2.3	Record Retrieval	73
5.2.4	Record Store	74
5.2.5	Append to Existing File	75
5.2.6	Deleting a File	75
5.2.7	Command/Batch Execution Files	76
5.2.8	Renaming a File	76

		Page
5.2.9	Extending Files	76
5.2.10	Display Attributes	77
5.2.11	Directory List	78
5.2.12	Rewind Data Stream	81
5.2.13	Truncate File	81
5.2.14	Free Buckets	81
5.2.15	Space Forward or Backward	82
5.2.16	Flush I/O Buffers	82
5.2.17	Deleting a Record	82
5.2.18	Find	83
5.2.19	Update	83
5.2.20	Wildcard Operations	84
5.2.20.1	Wildcard Sequential File Retrieval	85
5.2.20.2	Wildcard File Deletion	87
5.2.20.3	Wilcard File Rename	89
5.2.20.4	Wildcard Command File Execution	90
5.3	Closing a File and Terminating Data Streams	90
5.4	Terminating a Logical Link	90
5.5	File Security, Integrity and Protection	90
5.5.1	Access Control	90
5.5.2	Data Integrity	91
5.6	DAP-Based Applications Written by DIGITAL	91
5.6.1	Access Control and Accounting for DAP-Based Applications	91
APPENDIX A	USER IDENTIFICATION MESSAGE	93
APPENDIX B	REVISION HISTORY	95
B.1	Version 1.0 to Version 4.1	95
B.2	Version 4.1 to Version 5.6	95
GLOSSARY		97

FIGURES

FIGURE	1	Interrelationship of DNA Layers	3
	2	Typical DAP Message Exchange (Sequential File Retrieval)	5

TABLES

TABLE	1	DAP Messages	4
	2	MACCODE Field Values	32
	3	MICCODE Field Values for Use with MACCODE Values of 2, 10, and 11 Octal	33
	4	MICCODE Field Values for Use with MACCODE Values of 0, 1, 4, 5, 6, and 7 Octal	39
	5	MICCODE Field Values (with MACCODE Value of 12 Octal)	44
	6	Responses to Setup Message Errors	61

1.0 INTRODUCTION

This document describes the functions, operation, and message formats of the Data Access Protocol (DAP). The document is primarily intended to assist those who implement DAP in understanding how DAP functions within a system. The document does not address specific implementations.

DAP performs remote file access via a file system such as Record Management Services (RMS). Unit Record Devices and terminals can be accessed if supported by a file system. When Unit Record Devices and terminals are supported by a file system in a device-dependent manner, the device control features peculiar to the individual devices are not supported by DAP.

DAP is one of the protocols of the DIGITAL Network Architecture (DNA). DNA models the software (or hardware) for DECnet implementations. DECnet is a family of software modules, data bases, and hardware components typically used to tie DIGITAL systems together for resource sharing, distributed computation, or remote system communication.

DNA is a layered structure. Modules in each layer perform distinct functions. Equivalent modules within the same layer in both the same and different nodes communicate using protocols. A node is an implementation of the Session Control layer (Section 2.2). Usually a single computer is associated with one node. Protocols are the messages exchanged by modules and the rules governing the message exchanges. Modules in functionally different layers of DNA interface using either subroutine calls or a system-dependent method.

A glossary at the end of this document defines many DAP terms.

Other DNA Phase III functional specifications are:

> DNA Digital Data Communications Message Protocol (DDCMP) Functional Specification, Version 4.1.0, Order No. AA-K175A-TK
>
> DNA Maintenance Operations Protocol (MOP) Functional Specification, Version 2.1.0, Order No. AA-K178A-TK
>
> DNA Network Management Functional Specification, Version 2.0.0, Order No. AA-K181A-TK
>
> DNA Network Services (NSP) Functional Specification, Version 3.2.0, Order No. AA-K176A-TK
>
> DNA Session Control Functional Specification, Version 1.0.0, Order No. AA-K182A-TK
>
> DNA Transport Functional Specification, Version 1.3.0, Order No. AA-K180A-TK

The following overview document for these specifications is an introduction to the structure and functions of DNA.

> DNA General Description, Order No. AA-K179A-TK

2.Ø FUNCTIONAL DESCRIPTION

The Data Access Protocol is an application level protocol. Its primary purpose is to permit remote file access within the DECnet environment independently of the I/O structure of the operating system being accessed.

2.1 DAP Functions

Within DECnet, DIGITAL operating systems can employ DAP to provide the following remote file access functions:

- Retrieve a file from an input device.
- Store a file on an output device.
- Provide ASCII file transportability between nodes.
- Provide error recovery.
- Allow multiple data streams to be sent over a logical link.
- Command file execution and submission.
- Provide for random access of records in a file.
- Provide for file deletion.
- Rename files.
- List directories.

2.2 Relationship to the DIGITAL Network Architecture

The DIGITAL Network Architecture provides a modular design for DECnet. Its functional components are defined within eight layers which are described below. Each layer performs a well-defined set of network functions (via network protocols) and presents services to the layer above it:

User layer. The User layer contains all the user-supplied functions. It is the highest layer in the DNA structure.

Network Management layer. The Network Management layer contains the modules that provide user control over and access to network parameters and counters. Network Management modules also furnish down-line loading, up-line dumping, and testing functions.

Network Application layer. The Application layer supports the various user services and programs that utilize the network facilities. These services and programs must utilize the network communication mechanism provided by the Session Control and Network Services layers. DAP resides within the Application layer.

Session Control layer. The Session Control layer defines the system-dependent aspects of logical link communication.

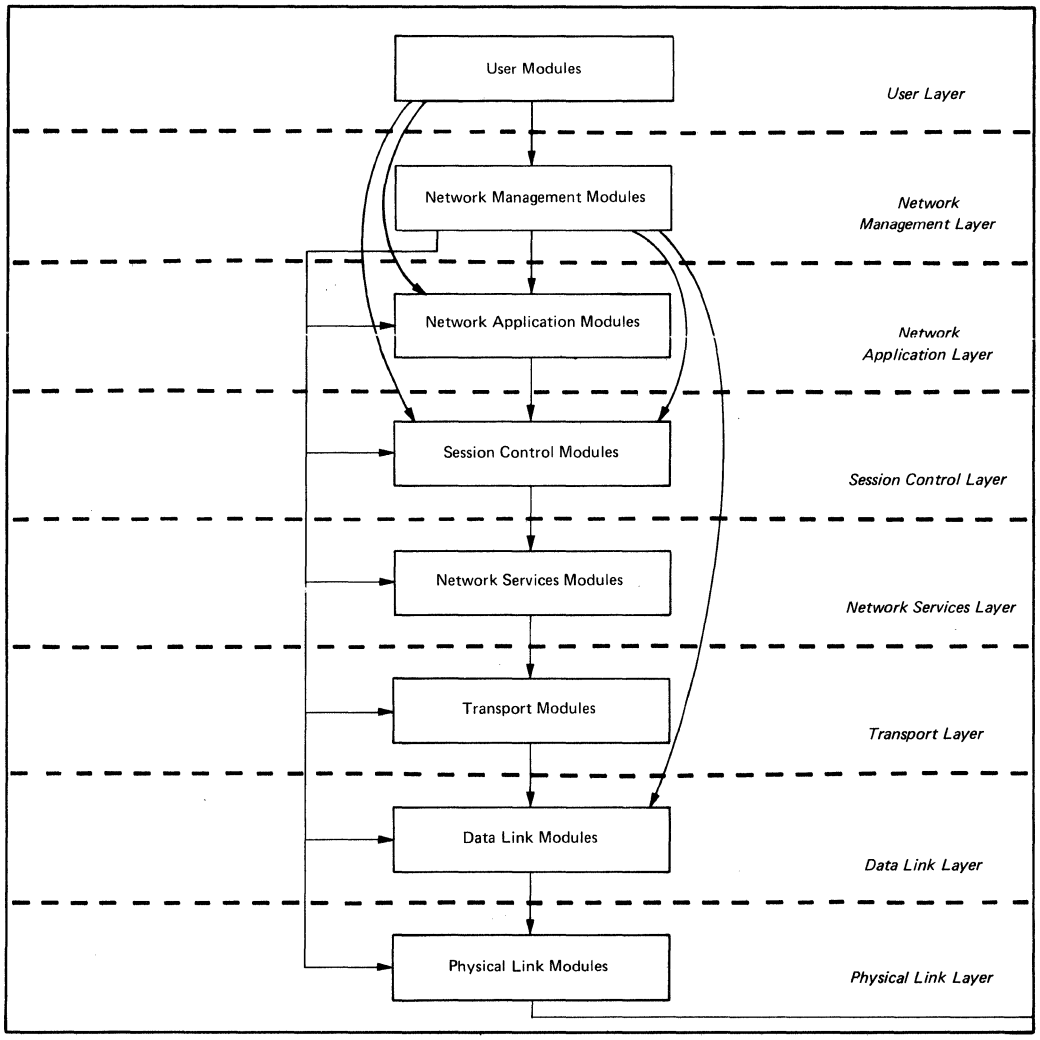
Network Services layer. The Network Services layer provides the mechanism that permits node-to-node communications and process-to-process communications between processes in the same or different nodes. It provides a logical link service and a datagram service to the network.

Transport layer. The Transport layer provides a mechanism for the network service layer to send a unit of data (a packet) from any node in the network to any other node in the network.

Data Link layer. The Data Link layer controls the physical link operation to ensure both data integrity and sequentiality.

Physical Link layer. The Physical Link layer, the lowest layer, consists of device specific modules that provide the interface to the communication hardware.

Figure 1 shows the interrelationship of the DNA layers.



Horizontal arrows show direct access for control and examination of parameters, counters, etc. Vertical and curved arrows show interfaces between layers for normal user operations such as file access, down-line load, up-line dump, end-to-end looping, and logical link usage.

Figure 1 Interrelationship of DNA Layers

2.3 Generic Model

As an aid toward understanding the Data Access Protocol, this section contains a generic model. This model consists of a summary of the DAP messages (Table 1) and a typical DAP message exchange sequence illustrating how DECnet sequential file retrieval is accomplished between two dialogue processes (Figure 2).

For a more detailed description of the DAP message formats and the protocol operation, refer to Sections 3.0 and 5.0, respectively.

Table 1
DAP Messages

Message	Function
Configuration	Exchanges system capability and configuration information between DAP-speaking processes. This message is sent immediately after a link is established. It contains information about the operating system, the file system, protocol version, and buffering ability.
Attributes	Provides information on how data is structured in the file being accessed. The message contains information on file organization, data type, format, record attributes, record length, size, and device characteristics.
Access	Specifies the file name and type of access requested.
Control	Sends control information to a file system and to establish data streams.
Continue-Transfer	Allows recovery from errors. Used for retry, skip, and abort after an error is reported.
Acknowledge	Acknowledges access commands and control connect messages used to establish data streams.
Access Complete	Controls termination of file and stream access.
Data Message	Transfers the file data over the link.
Status	Returns status and information on error conditions.
Key Definition Attributes Extension	Specifies key definitions for indexed files.
Allocation Attributes Extension	Used when creating or explicitly extending a file to specify the character of the allocation.

(continued on next page)

Table 1 (Cont.)
DAP Messages

Message	Function
Summary Attributes Extension	Returns summary information about file.
Date Time Attributes Extension	Specifies time-related information about the file.
Protection Attributes Extension	Specifies the file protection code.
Name	Sends name information when renaming a file or obtaining a directory listing.
Access Control List Attributes Extension	When creating a file, this message is used to specify the access rights users are granted for access to this file.

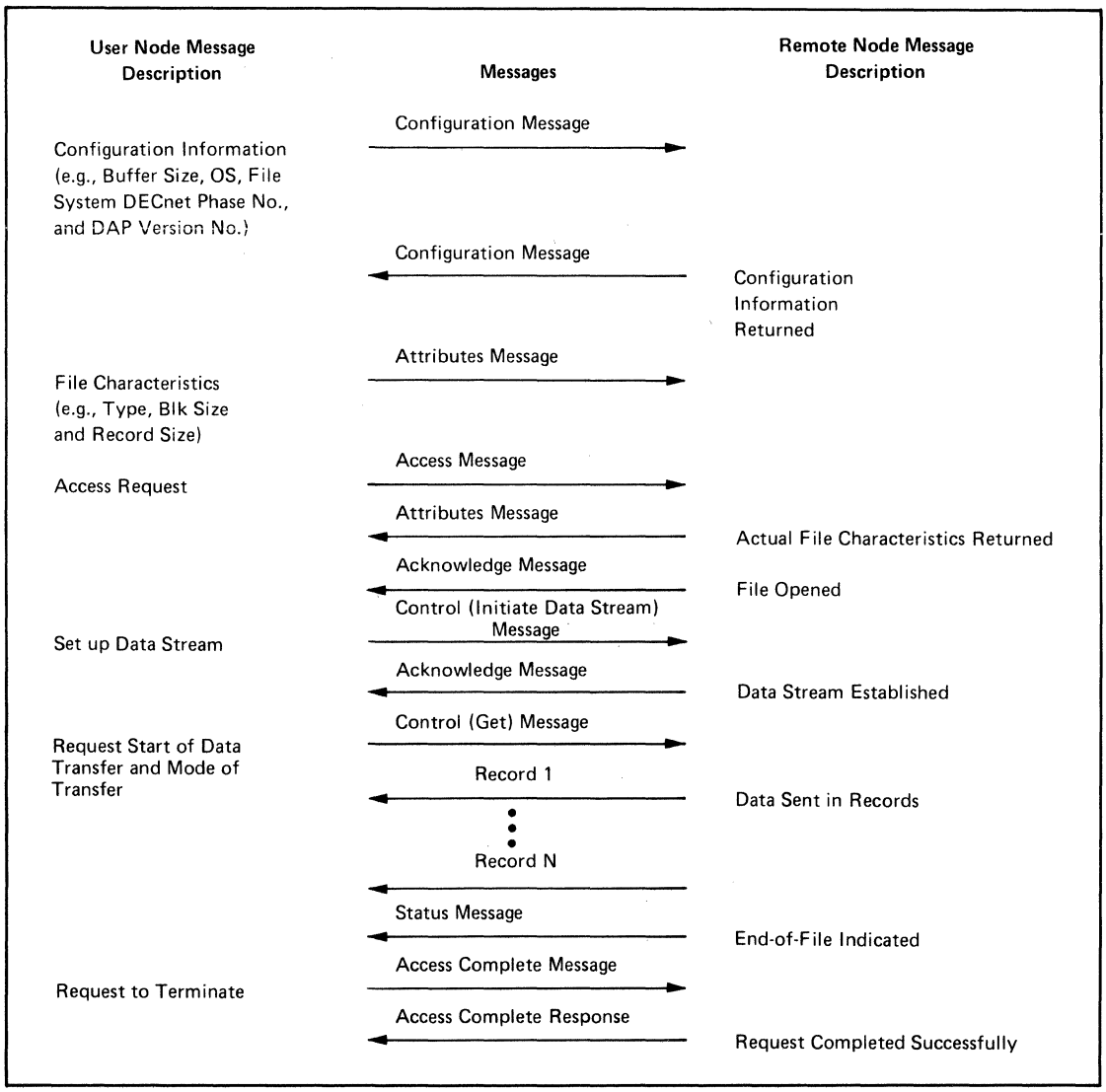


Figure 2 Typical DAP Message Exchange (Sequential File Retrieval)

3.Ø MESSAGE FORMATS

3.1 Notation

The following notation is used to describe the DAP messages:

FIELD (LENGTH) : CODING Description of field.

where:

FIELD — Is the name of the field being described.

LENGTH — Is the length of the field, which can be indicated in one of four ways:

1. A number meaning number of 8-bit bytes (octet).

2. A number followed by a "B" meaning number of bits.

3. The letters "EX" meaning extensible field. Extensible fields are of variable length consisting of 8-bit bytes in which the high-order bit of each byte denotes whether the next byte is part of the same field. A 1 means the next byte is part of this field and a Ø denotes the last byte. Extensible fields are for bit maps only. Seven bits from each octet are used as information bits. The notation EX-n means an extensible field where the maximum length of the field is n bytes.

   NOTE

   The bit definitions define the information bits after removing the extension bits and compressing the remaining bits.

4. The letters "I-n" means this is an image field, with n being a number that specifies the maximum length of the field in 8-bit bytes. The image is preceded by a 1-byte count of the length of the remainder of the field. Image fields are variable in length and may be null (count=Ø). All 8-bits of each byte are used as information bits. The meaning and interpretation of each image field is as defined with that specific field.

CODING — Is the representation type used, where:

A = 7-bit ASCII
B = binary
BM = bit map (in which each bit has a specific meaning)

The following rules apply to the notation:

1. If length and coding are omitted, field represents a number of subfields specified in the description.

2. Any bit or field described as "reserved" shall be zero unless otherwise specified.

3. All fields are presented to the Session Control Protocol with the least-significant byte first. In an ASCII field, the left-most character is contained in the low-order byte.

4. All numbers are in decimal unless otherwise specified.

5. When default values are defined for fields in DAP messages, the values will be used only if that field is absent from the message. There are two ways in which fields within DAP messages can be omitted so the default can be used:

   a. A field that appears under a MENU may be omitted by setting the corresponding MENU bit to zero.

   b. Trailing fields in DAP messages may be omitted if they are not needed or if the default value can be used. If a MENU field is truncated in this way, its value is zero (which means all the fields controlled by the MENU are absent, too).

   If a field is present with a zero value, do not use the default value. Use the value zero.

6. Brackets [ ] denote optional fields.

3.2 General Message Format

All DAP messages have the following form:

OPERATOR	OPERAND

where:

OPERATOR — This field describes the characteristics and type of message. It is divided into seven subfields: TYPE, FLAGS, STREAMID, LENGTH, LEN256, BITCNT, and SYSPEC.

TYPE(1) : B — The type of DAP message. These numbers are given with each DAP message description. Types 128-191 are reserved for DIGITAL applications based on DAP. Types 192-255 are reserved for user extensions to DAP.

FLAGS(EX-5) : BM — The DAP message flags. Bits in this extensible field are currently defined as follows:

Bit(s)	Meaning (When Set)
Ø	Stream identification field present.
1	Length field present.
2	If bit 1 (length field present) is set, field LEN256 is present and the length field is in effect 2 bytes long. Illegal if bit 1 not set.
3	The BITCNT field is present.
4	Reserved (Ø).
5	SYSPEC field present.
6	If set, this is a segmented message and this is not the last segment of the message. The next message will contain the next segment of the full message. The next message must be of the same type as this message. Refer to the SYSCAP field of the Configuration message to see if this facility is supported.

STREAMID(1) : B — The stream identification field. This field is included only if bit Ø of the FLAGS field is set. This field is used to allow a single user to have multiple data streams in use for a single open file. All data streams use the same logical link (they multiplex on the STREAMID number).

If the STREAMID number is omitted, it is assumed to be zero. Not all file systems are capable of supporting multiple data streams from a single file.

LENGTH(1) : B — Denotes the length of the OPERAND field (number of 8-bit bytes). This field is optional. It is included only if bit 1 of the FLAGS field is set. Two or more DAP messages may be blocked together into one Session Control buffer (usually for reasons of efficiency). If DAP messages are blocked, the LENGTH field must be present so the end of one message and start of the next can be found.

Messages between Ø and 255 bytes long may be blocked using only the LENGTH field. DAP messages whose operand length is between 256 and 64K bytes may be blocked only if both bits 1 and 2 of FLAGS are set. Lengths greater than 64K bytes are sent unblocked or as the last part of a blocked message.

LEN256(1) : B — Contains the most significant byte of a two byte OPERAND length if bit 2 of FLAGS is set. LENGTH contains the least significant byte. If LEN256 is present, LENGTH must also be present.

BITCNT(1) : B — This field is valid only with the Data message. If present, it contains a number in the range Ø-7 which is the number of unused bits in the final 8-bit byte of the message. It is required only when transmitting data which does not completely fill the final 8-bit frame of a Data message. This is useful when accessing files whose byte size is not a multiple of eight. See Section 4.4.3.

SYSPEC(I-255) : B — This optional field is the system specific field. It can only be used for file access between two homogeneous systems. This field is included only if bit 5 of the FLAGS field is set. If it is used between hetrogeneous systems, it will produce a fatal error and the access will be aborted. Between homogeneous systems, this field can be used for passing system specific information as defined by the system. The SYSPEC field can not be used with the Configuration message. This field must not be used for passing information common to more than one system.

NOTE

If this field is used, all system specific functions must be registered with the architecture group within DECnet in advance of certification.

OPERAND The information field for DAP messages. It is dependent on the TYPE field.

NOTE

Typically, one DAP-speaking process sends one or more DAP messages to a cooperating process which then responds with one or more messages. Section 5 contains examples of message sequences. This pattern repeats until the access is complete.

Blocking DAP messages. DAP messages can be blocked in one of two ways:

1. The first process blocks and sends DAP messages up to the point where it expects a response from the cooperating process. It then waits for the response.

2. The first process blocks and sends DAP messages without regard for response from the cooperating process. The first process may be way ahead of the cooperating process. This means the cooperating process must be able to receive a buffer full of DAP messages, process some of them, send a response and then continue processing DAP messages from the same buffer from where it left off before sending the response. If the response was an error, the unprocessed messages in the buffer may no longer be valid and must be discarded.

   The first method of blocking is the more commonly used. The type of blocking a system supports is specified in the Configuration message. Some small systems may not support blocking at all.

Truncating DAP messages. DAP messages can be truncated up to the point of leaving only the TYPE field provided the fields truncated are not needed or can be defaulted. This is particularly useful with the Acknowledge message which reduces the ACK to a one byte message.

3.3 Configuration Message

The Configuration message passes system configuration information between the cooperating processes involved in DAP remote file access. This message is sent immediately following link establishment. The accessed process may wait to receive a Configuration message from the accessing process before it sends a Configuration message itself. However, this is not necessary. The Configuration message should not be sent blocked with any other DAP message. The reason for this is that buffers of the appropriate size can be allocated for receiving subsequent DAP messages. The Configuration message format is:

CONFIG	BUFSIZ	OSTYPE	FILESYS	VERSION	SYSCAP

where:

CONFIG : The OPERATOR field with TYPE = 1.

BUFSIZ(2) : B The maximum buffer size (in bytes) of the sending system for message exchange. The two cooperating DAP speaking processes will use the lesser of the two buffer sizes as the maximum size. If one of the two systems has an unlimited buffer size, it sends a Ø and the two systems will use the nonzero buffer size as the maximum. If both systems send Ø, there is no limit on the length of messages sent.

OSTYPE(1) : B Operating system type (the sending system). Values in the range 1-191 are reserved for DIGITAL use; 192-255 are reserved for user-specified operating systems.

Value	OS Type
Ø	Illegal
1	RT-11
2	RSTS/E
3	RSX-11S
4	RSX-11M
5	RSX-11D
6	IAS
7	VAX/VMS
8	TOPS-2Ø
9	TOPS-1Ø
1Ø	RTS-8
11	OS-8
12	RSX-11M+
13	COPOS/11 (TOPS-2Ø front-end)

FILESYS(1) : B

File system type (of the file system being used by the process sending this message). Values in the range 1-191 are reserved for DIGITAL use; 192-255 are reserved for user-specified file systems.

Value	System
Ø	Illegal
1	RMS-11
2	RMS-2Ø
3	RMS-32
4	FCS-11
5	RT-11
6	No file system supported
7	TOPS-2Ø
8	TOPS-1Ø
9	OS-8

VERSION :

A field identifying the protocol and software version numbers. This field is subdivided as follows:

VERNUM	ECONUM	USRNUM	SOFTVER	USRSOFT

where:

VERNUM(1) : B — DAP version number. This is the same as the first digit of the protocol version number.

ECONUM(1) : B — DAP ECO (Modification) number. This is the same as the second digit of the protocol version number.

USRNUM(1) : B — Customer modification level of DAP. Set to Ø by DIGITAL.

SOFTVER(1) : B — DAP software version number in binary. This is the DIGITAL release number. If the software is completely user written, this field should be Ø.

USRSOFT(1) : B — User software version number in binary. If the user modifies DIGITAL software, he should increment this byte to reflect his modification number. Set to Ø by DIGITAL.

SYSCAP(EX-12) : BM Generic system capabilities. These are defined as follows:

Bit	Meaning (When Set)
Ø	Supports file preallocation.
1	Supports sequential file organization.
2	Supports relative file organization.
3	Intended to support direct file organization (reserved).
4	Supports single keyed indexed file organization (reserved).
5	Supports sequential file transfer.
6	Supports random access by record number.
7	Supports random access by Virtual Block Number.
8	Supports random access by key.
9	Intended to support random access by user generated hash code (reserved).
1Ø	Supports random access by Record File Address (RFA).
11	Supports multi-keyed indexed file organization.
12	Supports switching access mode. (See RAC field in Section 3.6.)
13	Supports append to file access.
14	Supports command file submission and/or execution as specified in the Access message.
15	Supports data compression (reserved).
16	Supports multiple data streams.
17	Supports status return (reserved).
18	Supports blocking of DAP messages up to response. (See note Section 3.2.)
19	Supports unrestricted blocking of DAP messages. (See note Section 3.2.)
2Ø	Supports the use of two byte operand length in the DAP message header, i.e., LENGTH and LEN256.
21	Supports the file checksum option (See ACCOPT field in the Access message and also Section 5.5.2).
22	Supports the Key Definition Extended Attributes message.
23	Supports the Allocation Extended Attributes message.
24	Supports the Summary Extended Attributes message.
25	Supports directory list.
26	Supports the Date and Time Extended Attributes message.
27	Supports the File Protection Extended Attributes message.

Bit	Meaning (When Set)
28	Supports the Access Control List Extended Attributes message (reserved).
29	Supports spooling as specified by bit 20 of the FOP field (see Attributes and Access Complete messages).
30	Supports command file submission as specified by bit 21 of the FOP field.
31	Supports file deletion as specified by bit 22 of the FOP field.
32	Supports the default file specification (see Section 3.17) (reserved).
33	Supports sequential record access.
34	Supports the recovery option for file transfer (reserved).
35	Supports the use of the BITCNT field in the Data message.
36	Supports the warning Status message (MACCODE = 6).
37	Supports the file rename operation.
38	Supports wildcard operations (see Section 5.2.19).
39	Supports the Go/No-Go option (see Section 3.5, ACCOPT field). (reserved)
40	Supports the Name message.
41	Supports segmented DAP messages (see bit 6 of FLAGS field). (reserved)

NOTE

Bits 5 and 33 differentiate between the use of file transfer (see Sections 5.2.1 and 5.2.2) and record access on sequential files (see Section 5.2.3 and 5.2.4) where file transfer reduces overhead by eliminating the need for Control messages.

3.4 Attributes Message

The Attributes message describes how data is being represented in a file being accessed. It is sent as a part of the initial set up. The Attributes message format is as follows:

ATTRIB	ATTMENU	DATATYPE	ORG	RFM	RAT	BLS	MRS	ALQ	BKS	FSZ

MRN	RUNSYS	DEQ	FOP	BSZ	DEV	SDC	LRL	HBK	EBK	FFB	SBN

NOTES

1. Symbolic names, where supplied, refer to the corresponding RMS names. They are included here for ease of reference only; they have no meaning for DAP.

2. Values passed to the accessed system's file system may not be used literally in some cases. For example if a value of Ø is passed in the DEQ field, RMS systems ignore the Ø and use the local default rather than return an error as might be expected. The application of defaults, as in this example, is a function of individual file systems and is in no way related to defaults as specified in DAP. If a field is present in a DAP message, no DAP specified default will be substituted in place of the value it contains.

3. Sometimes fields will be ignored if they are not applicable to the particular operation or are not supported and their omission will not affect the operation. For example, RMS ignores MRS as input to the OPEN operation. Also, with bit map fields, sometimes bits will be ignored where they are not applicable to the operation or not supported by that particular file system.

where:

ATTRIB : The OPERATOR field with TYPE = 2.

ATTMENU(EX-6) : BM The following bit map specifies which of the attributes fields will be present in the main attributes message when the corresponding bit is set. These fields and only these fields may appear in the message and they must be in the order specified.

Bit	Meaning (When Set)
0	DATATYPE
1	ORG
2	RFM
3	RAT
4	BLS
5	MRS
6	ALQ
7	BKS
8	FSZ
9	MRN
10	RUNSYS
11	DEQ
12	FOP
13	BSZ
14	DEV
15	SDC (reserved)
16	LRL
17	HBK
18	EBK
19	FFB
20	SBN

DATATYPE(EX-2) : BM

The type of data being transferred. The default is Image. Unless a file has attributes specifying whether the file contains ASCII or Image data, the accessed process returns the value (ASCII or Image) sent by the accessing process when opening a file.

Bit	Meaning (When Set)
0	ASCII (see Note 1).
1	Image (default) (see Note 2).
2	EBCDIC (reserved).
3	Compressed format.
4	Executable code.
5	Privileged code.
6	Reserved (set 0 when sending Attributes message; ignore when receiving Attributes message).
7	Sensitive data -- zero on delete. Data in file is set to zero when the file is deleted for data security.

NOTES

1. This is the 7-bit ASCII code set as defined in the 1968 ANSI Standard. When transmitting or receiving 7-bit ASCII in 8-bit frames, the high order bit is ignored except when using 7-bit compression.

2. Image is the mode where no code-set is specified. It is a format for sending 8-bit quantities in DAP without specifying any code representation. The actual data may be ASCII, binary or anything else. The user process determines how to use the data. See Section 4.4.3.

ORG(1) : B

Attributes of the file being accessed. These attributes are as follows:

Value (octal)	Meaning
0	FB$SEQ; Sequential (default).
20	FB$REL; Relative.
40	FB$IDX; Indexed.
60	FB$HSH; Hashed (reserved).

RFM(1) : B

Format of the records being transferred. These formats are as follows:

Value	Meaning
0	FB$UDF; Undefined record format.
1	FB$FIX; Fixed-length records (default).
2	FB$VAR; Variable-length records.
3	FB$VFC; Variable with fixed control format.
4	FB$STM; ASCII Stream Format.

RAT(EX-3) : BM

Information about the attributes of individual records. The default is all bits set 0.

Bit	Meaning (When Set)
0	FB$FTN; Records contain FORTRAN carriage control (see Note 1).
1	FB$CR; Records have an implied LF/CR envelope.
2	FB$PRN; Print file carriage control where pre- and post-fix carriage control information is stored in the fixed header field of files in variable with fixed control (VFC) format.
3	FB$BLK; Records that do not span blocks (see Note 2).
4	Records have embedded format control (see Note 3).
5	Intended for COBOL carriage control (reserved).
6	FB$LSA; Line-sequenced ASCII Format.
7	MACY11 format (note 4).

NOTES

1. FORTRAN Carriage Control. For line printers and some terminals, treat the first character of each record as a carriage control character.

2. This bit, when set, informs the system that the record length should not exceed the physical device blocking size. With some systems and on some I/O devices (for example, disk and magnetic tape) this may be a factor in determining the actual format used on the device.

3. This bit, when set, informs the system that records in the file may contain format control characters (LF, VT, and so on).

4. MACY11 format is a standard used on 10's and 20's to store files destined for PDP-11's. These files usually contain 8-bit data such as object code output by a cross compiler.

BLS(2) : B

Physical block size in bytes on media. The default value is 512. The actual byte size is as specified by field BSZ.

MRS(2) : B

The length of each file record in number of bytes. For variable-length records, this field specifies the maximum record size. When the accessed process receives the MRS (maximum record size), it must check it against the length of its buffer. If the buffer will not accommodate this size record, the accessed process should return an error. A zero value means no checking is performed on record size. MRS is used by the various file systems in a system-dependent manner. The default is 0.

ALQ(I-5) : B

This field specifies the allocation quantity in blocks. For file creation, it specifies the initial size of the new file. The actual size of the new file is returned in this field. The default is 0.

NOTE

Ignore this value on opening existing files. Use this field only to return the file size.

BKS(1) : B

Bucket size in blocks. Used for access to relative (not RMS-20), hashed and indexed files with RMS. The default is 0 (see Note 2 at the top of Section 3.4).

FSZ(1) : B

Size in bytes of fixed part of variable length record with fixed control format. The default is Ø (see Note 2 at the top of Section 3.4).

MRN(I-5) : B

Maximum record number for file (for relative files only). If set to Ø, checking is suppressed. The default is Ø.

RUNSYS(I-4Ø) : A

Name of the Run-Time System environment required to execute the code contained in the file. This field is useful to operating systems that can emulate other operating system environments. The default value is accessed-operating-system-dependent.

DEQ(2) : B

File extension quantum size in virtual blocks. This size is the amount of space, in blocks, added to the file each time the file is implicitly extended. The default is Ø (see Note 2 at the top of Section 3.4).

FOP(EX-6) : BM

The file access options a user requires. The default is all bits set to Ø.

Bit	Meaning (When Set)
Ø	FB$RWO; Rewind on open.
1	FB$RWC; Rewind on close.
2	Reserved (Ø).
3	FB$POS; Position magnetic tape just past the most recently created file.
4	FB$DLK; Do not lock file if not properly closed.
5	Reserved (Ø).
6	File Locked.
7	FB$CTG; A contiguous file creation or extension required.
8	FB$SUP; Supersede existing file on create.
9	FB$NEF; Do not position to EOF on opening magnetic tape file for PUT.
1Ø	FB$TMP; Create temporary file.
11	FB$MKD; Create temporary file and mark for delete on close.
12	Reserved (Ø).
13	FB$DMO; Rewind and dismount magnetic tape on close.
14	FB$WCK; Enable Write checking.
15	FB$RCK; Enable Read checking.
16	FB$CIF; Create new file if one by the same name does not exist. If one does exist, open the highest version of the file.
17	FB$LKO; Override file lock on open (reserved).
18	FB$SQO; Sequential access only.
19	FB$MXV; Maximize version number.
2Ø	FB$SPL; Spool file to line printer (one copy only) on close.

Bit	Meaning (When Set)
21	FB$SCF; Submit as command file on close.
22	FB$DLT; Delete file on close. May be used as a sub-option with submit or spool.
23	FB$CBT; Contiguous best try. The file will be created but it will be contiguous only when it is possible.
24	FB$WAT; Wait for file if it is locked by another process (reserved).
25	FB$DFW; Deferred write (REL and IDX files).
26	FB$TEF; Truncate at EOF on close (write accessed SEQ files).
27	FB$OFP; Output file parse (the name type will be preserved until overridden).

BSZ(1) : B

Number of bits per byte for data stored in the file on the accessed node. Data is always transmitted in 8-bit frames (see Section 4.4).

When a DAP Attributes message is sent from the accessing to the accessed system, BSZ is required only when dealing with systems capable of supporting a variable byte size, such as TOPS-20. The default value for BSZ is the normal default for the accessed system's file system, for example, 8 bits per byte for RSX.

When the Attributes message is returned from the accessed to the accessing system, BSZ should always be sent unless the default applies. The default for BSZ in a returned Attributes message is 8 bits.

DEV(EX-6) : BM

For attributes sent to the accessing node, this field contains the generic characteristics of the device on which a file resides. The default is all bits set to 0.

Bit	Meaning (When Set)
0	FB$REC; Record oriented.
1	FB$CCL; Carriage control device.
2	FB$TRM; Terminal.
3	FB$MDI; Directory structured.
4	FB$SDI; Single directory only.
5	FB$SQD; Sequential, block oriented (for example, magnetic tape).
6	; Null device.
7	FB$FOD; A file-oriented device (for example, a disk or magnetic tape).
8	; Device can be shared.
9	FB$SPL; Device is being spooled.
10	FB$MNT; Device is currently mounted.
11	FB$DMT; Device is marked for dismount.

Bit	Meaning (When Set)
12	FB$ALL; Device is allocated.
13	FB$IDV; Device is capable of providing input.
14	FB$ODV; Device is capable of providing output.
15	FB$SWL; Device is software write-locked.
16	FB$AVL; Device is available for use.
17	FB$ELG; Device has error logging enabled.
18	FB$MBX; Device is a mailbox.
19	FB$RTM; Device is realtime in nature, not suitable for RMS use.
20	FB$RAD; A random access device.
21	; Device has read checking enabled.
22	; Device has write checking enabled.
23	; Device is foreign.
24	; Network device.
25	; Generic device.

SDC(EX-6) : BM (Reserved for future use)
: Spooling device characteristics. SDC uses the same bit definitions as in the DEV field. If the file is spooled, SDC contains the characteristics of the spooling device. The characteristics of the ultimate device are contained in DEV. The default is all bits set to 0.

LRL(2) : B
: Longest record length. Length of the longest record in the file.

HBK(I-5) : B
: Highest virtual block allocated to the file.

EBK(I-5) : B
: End of file virtual block number.

FFB(2) : B
: First free byte in end of file block--byte size as defined in BSZ.

SBN(I-5) : B
: Starting logical block number for file if contiguous; else 0.

3.5 Access Message

The Access message specifies the file name and type of access requested. The format for the Access message is as follows:

ACCESS	ACCFUNC	ACCOPT	FILESPEC	FAC	SHR	DISPLAY	PASSWORD

NOTE

Symbolic names, where supplied, refer to the corresponding RMS names. They are included here for ease of reference only. They have no meaning for DAP.

where:

ACCESS : The OPERATOR field with TYPE = 3.

ACCFUNC(1) : B The request code specifying the operation to be performed is as follows:

1 $OPEN; Open existing file.
2 $CREATE; Open new file.
3 $RENAME; Rename a file.
4 $ERASE; Delete a file.
5 Reserved.
6 Directory List.
7 Submit as a command (batch) file.
8 Execute command (batch) file.

If $RENAME is specified, the FILESPEC field below contains the old file specification and the new file specification is contained in the Name message which follows the Access message.

If $CREATE is specified, but a file of that name already exists, follow the rules of the accessed node for file creation. For example, the file system may create a new file whose version number is one greater than the current highest version number.

NOTE

The Data Access Protocol (DAP) does not perform functions beyond remote data access. DAP should not be extended to include RJE, spooling, or other similar functions which, while they involve file transfer, also require command processing, parameter passing, and job queueing. The two command file submission commands are here for historical reasons: they were implemented in the first release of DAP-based software.

ACCOPT(EX-5) : BM The access options are as follows:

Bit	Meaning (When Set)
Ø	I/O errors are non-fatal. A record may be skipped or repeated as specified by the Continue Transfer message. If not set, I/O errors are fatal and terminate the access.
1	A status message will be returned following each record sent to the accessed process in the record access mode (reserved).

Bit	Meaning (When Set)
2	A status message is returned with each record retrieved from an accessed system. The status message should precede the Data message so that it is always possible to block the two into one Session Control buffer. When a user requires a Record File Address (RFA) to be returned, this option is used (reserved).
3	A 16 bit file checksum will be generated by both the transmitting and receiving nodes. When closing the file, the accessing process sends the checksum it generated to the accessed process in the Access Complete (Close) message. The accessed process closes the file if the checksums agree. If they do not agree, it returns a status message. See Section 5.5.2.
4	The Go/No-Go option is to be used with this operation. Go/No-Go is valid only for the Delete, Rename, and Execute Command File functions and wildcard operations using these functions. Go/No-Go operation causes the accessed process to return the name of the file to the accessing process before executing the operation. The accessing process can then choose whether or not to perform the operation on the file by sending either a Control(Resume) or Control(Skip) See Section 5.2.11 for state operation.

FILESPEC (I-255) : A

The file specification in the format required by the remote node. A null file specification assumes the meaning of a null file specification on the target node.

FAC(EX-3) : BM

The file access operations a user requires:

Bit	Meaning (When Set)
Ø	FB$PUT; Put access.
1	FB$GET; Get access (default).
2	FB$DEL; Delete access.
3	FB$UPD; Update access.
4	FB$TRN; Truncate access.
5	FB$BIO; Block I/O acess (see Note).
6	FB$BRO; Support switching between block and record I/O.

NOTE

```
FB$REA = FB$BIO!FB$GET;  Block I/O Read access.
FB$WRT = FB$BIO!FB$PUT;  Block I/O Write access.
```

SHR(EX-3) : BM Operations shared with other users:

Bit	Meaning (When Set)
Ø	FB$PUT; Put access.
1	FB$GET; Get access (default).
2	FB$DEL; Delete access.
3	FB$UPD; Update access.
4	FB$MSE; Enable multi-stream access.
5	FB$UPI; User provided interlocking (allows multiple writers to SEQ files).
6	FB$NIL; No access by other users.

DISPLAY(EX-4): BM Attributes and Extended Attributes messages which are to be returned in response to this Access message. See note 2 Section 3.6.

Bit	Meaning (When Set)
Ø	Main Attributes message (see Note 1, Section 5.1.2).
1	Key definition Attributes message.
2	Allocation Attributes message.
3	Summary Attributes message.
4	Date and Time Attributes message.
5	File Protection Attributes message.
6	Reserved (Ø).
7	Access Control List Attributes message (reserved).
8	Name message containing resultant file specification. If the file was opened using logical name(s), this will return the file specification of file opened without logical names. NOTE When opening a file and DISPLAY requests key definition and allocation attributes for that file, the Attributes message must be followed by Key Definition and Allocation Attributes Extension messages specifiying for which keys and which areas this information is to be returned. If no keys or areas are specified, no key or area information will be returned. See Section 5.1.2 for the setup message sequence.

PASSWORD(I-4Ø): B Password required to obtain access to file.

3.6 Control Message

The Control message sends control type information to a file system. The Control message format is as follows:

CONTROL	CTLFUNC	CTLMENU	RAC	KEY	KRF	ROP	HSH	DISPLAY

where:

CONTROL : The OPERATOR field with TYPE = 4.

CTLFUNC(1) : B Specific control information:

Value	Meaning
1	$GET (or $READ for block I/O); Get record (or block). If random access to a relative file is made, the KEY field contains the record number. If a random access to an indexed file is made, KEY contains the key. If sequential access is employed, get the next record (default).
2	$CONNECT; Initiate data stream. If multiple data streams are used, they are multiplexed on the STREAMID number. The STREAMID number in the Control message initiates a data stream. If the STREAMID number is omitted, assume a default of zero.
3	$UPDATE; Update current record. Indicates to the accessed system the intent of the accessing system to update the currently positioned record with the next data transmission.
4	$PUT (or $WRITE for block I/O); Indicates to the accessed system, that the information to follow should be written into the file.
5	$DELETE; Delete current record.
6	$REWIND; Rewind file.
7	$TRUNCATE; Truncate file. Writes end-of-file at current position. Used with sequential files only.
8	$MODIFY; Change file attributes (reserved).
9	$RELEASE; Unlock record specified by Record File Address in KEY field (reserved).
10	$FREE; Unlock all locked records for this data stream.
11	Reserved.

Value	Meaning
12	$FLUSH; Write out all modified I/O buffers and attributes for this data stream.
13	$NXTVOL; Perform end-of-volume and start-of-next-volume processing (reserved).
14	$FIND; Find record. Same as 1, but the data is not transferred.
15	$EXTEND; Extend this file by the amount specified in the following Allocation Attributes Extension message (reserved).
16	$DISPLAY; Retrieve this file's attributes as defined by the field DISPLAY (reserved).
17	SPACE FORWARD; Forward space the file by the number of blocks specified by KEY below. Block I/O only.
18	SPACE BACKWARD; Backward space the file by the number of blocks specified by KEY below. Block I/O only.

CTLMENU(EX-4):BM

The following bits when set, indicate the following optional fields are present. These fields and only these fields may appear in the message and they must be in the order specified.

Bit	Field
Ø	RAC
1	KEY
2	KRF
3	ROP
4	HSH (reserved)
5	DISPLAY (reserved)

RAC(1) : B

Sets the access mode. If this field is not present, retain the option in force for the last access. The default is Ø if not set previously.

Value	Meaning
Ø	RB$SEQ; Sequential record access.
1	RB$KEY; Keyed access.
2	RB$RFA; Access by Record File Address (RFA--an RMS specific access mode).
3	Sequential file access (the remainder of the file is transferred sequentially from the current file position).
4	Block mode; Access by Virtual Block Number. For retrieval, a Control message must request each block as in the record access mode.
5	Block mode file transfer. Blocks are transferred sequentially to end-of-file without need for a Control message preceding each block transferred. An explicit Control (Get) or (Put) is required to start data moving.

KEY(I-255) : B

File or Mode	Key
Relative Files	Record Number
Indexed Files	Record Key
Hashed Files	Record Key
Record File Address	
Access Mode	Record File Address
Block Mode Access	Virtual Block Number (binary, range 1 to n)
Recovery	Input File Checkpoint Locater

Right justify non 8-bit quantities in the KEY field with the high order and unused bits set to zero. If the key consists of 7-bit ASCII characters, right justify each 7-bit character in an 8-bit frame as is usual for the transmission of ASCII characters.

KRF(1) : B

Key of reference. If this field is not present, do not change the key of reference. Default is primary if never set.

Value	Meaning
Ø	Primary key
1-254	Secondary key indicator

ROP(EX-6) : BM

Optional record processing features. If this field is not present, retain the options in force for the last access.

Bit	Meaning (When Set)
Ø	RB$EOF; Position to EOF.
1	RB$FDL; Fast delete--mark record deleted but do not remove pointers from index.
2	RB$UIF; $PUT's update existing records in relative files.
3	RB$HSH; Use hash code in HSH (reserved).
4	RB$LOA; Follow fill quantities.
5	RB$ULK; Manual locking/unlocking (reserved).
6	RB$TPT; Truncate Put--writes EOF at current position. SEQ files only (Put also occurs).
7	RB$RAH; Read ahead.
8	RB$WBH; Write behind.
9	RB$KGE; Key is >=.
1Ø	RB$KGT; Key is >.
11	RB$NLK; Do not lock record.
12	RB$RLK; Read a locked record (read only).
13	RB$BIO; Block I/O.
14	RB$LIM; Compare for key limit reached (reserved).
15	RB$NXR; Non-existent record processing (reserved).

HSH(I-5) : B
(Reserved for future use)

Hash code if keyed access on direct file is employed and the user is doing hashing.

DISPLAY(EX-4) : BM (Reserved for future use)

Attributes messages to be returned in response to a request to retrieve the file's attributes are:

Bit	Meaning (When Set)
Ø	Main Attributes message.
1	Key definition attributes.
2	Allocation attributes.
3	Summary Attributes message.
4	Date and Time Attributes message.
5	File Protection Attributes message.
6	Reserved (Ø).
7	Access Control List Attributes message (reserved).
8	Name message containing resultant file specification--if file opened using logical name(s), this returns the file specification of the file opened without logical names.

NOTE

1. When a file's attributes and Key definition and/or allocation attributes have been requested in the DISPLAY field of a Control message, the Control message must have been preceded by Key Definition and/or Allocation Attributes Extension messages specifying for which Keys and/or areas this information is to be returned. If no keys or areas are specified, no Key or area information will be returned. See Section 5.2.1Ø for the message sequence.

2. An accessing process seeing the accessed process does not support a particular message type (as indicated in the Configuration message from the accessed process) does not request that message type in the DISPLAY field of either the Access Complete or Control messages. If an unsupported message type is requested, return an error.

3. FAL's written to versions of DAP prior to 5.6 will not return a Status message for successful completion with Get, Put, Delete, and Find operations. FAL's written to DAP version 5.6 and later versions will return successful status for these operations. See Sections 5.2.3, 5.2.4, 5.2.17 and 5.2.18 for the operation of these functions.

3.7 Continue Transfer Message

The Continue Transfer message tells the accessed process what action to take when an error is detected in an I/O transfer. Normally, the accessed process informs the accessing process of an I/O error using a Status message. The accessing process returns a Continue Transfer message. This message is also used when the accessed process suspends to await an operator decision as with Go/No-Go operation. The format of the Continue Transfer message is:

CONTRAN	CONFUNC

where:

CONTRAN : The OPERATOR field with TYPE = 5.

CONFUNC(1) : B This field specifies the recovery action to be taken:

Value	Meaning
1	Try again.
2	Skip and continue.
3	Abort transfer. (Discard all records in the pipeline until an Access Complete message is found indicating the pipeline is clear.)
4	Resume processing (used to restart accessed process processing data for this data stream if the accessed process is suspended).

3.8 Acknowledge Message

The Acknowledge message acknowledges access commands, control connects, and the taking of a checkpoint. Its format is as follows:

ACKNOWLEDGE

where:

ACKNOWLEDGE: The OPERATOR field with TYPE = 6.

3.9 Access Complete Message

The Access Complete message either terminates access or acknowledges a request to terminate access. The Access Complete message format is as follows:

ACCOMP	CMPFUNC	FOP	CHECK

where:

ACCOMP : The OPERATOR field with TYPE = 7.

CMPFUNC(1) : B The access completion functions are:

Value	Meaning
1	Close. Terminate access. Close a file that is currently open. When multiple data streams are in use, close all of them ($CLOSE).
2	Response. Sent by the accessed process in response to all Access Complete messages from the accessing process unless an error occurs.
3	Purge. Purge a file. That is, close and delete ($CLOSE + $ERASE) the file.
4	End-of-stream. Terminate the data stream associated with this STREAMID number, but do not close the file ($DISCONNECT).
5	Skip. Close the file currently associated with this link (forcefully, if necessary--ignore any errors that may occur in closing the file) and go on to process the next file. This function is for use with a series of wildcard transfers. NOTE FAL's written to DAP version 4.1 will not return an Access Complete (Response) to an Access Complete (End-of-Stream) (EOS). FAL's written to later versions of DAP will return an Access Complete (Response). The Access Complete (End-of-Stream) does not close the file. The accessed process should be in a state wherein it can accept another Control (Connect) to open another stream.

FOP(EX-6) : BM The file access options a user requires. Refer to Section 3.4 for option values. If any portion of the FOP field in the Access Complete is present, the whole FOP from the Attributes is superseded with unspecified bits being set Ø.

CHECK(2) : B The 16 bit file checksum if requested in the ACCOPT field of the Access message. See Section 5.5.2. Send the checksum only in the Access Complete (Close) message. Return a Status message if the checksum is incorrect. When this field is present, the accessed process compares the checksums. If this field is absent, make no checksum comparison and close the file even if it is known to contain errors.

3.10 Data Message

The Data message transfers the file data over a DAP link. The Data message format is as follows:

DATA	RECNUM	FILEDATA

where:

DATA : The OPERATOR field with TYPE = 8.

RECNUM(I-8) : B This field sends the record number when accessing relative files (or sequential files in a relative manner, in other words, by record number). For random store, this field contains the record number (for relative files) or hash code (if the user is generating his own hash codes with hashed files). When RECNUM is not used, the byte count is zero. When in block mode, this field contains the VBN instead of the record number. For file retrieval with recovery, this field contains the input file checkpoint locater.

FILEDATA The file data being transferred. This field is totally transparent and uses all 8-bits of each byte.

3.11 Status Message

The status message returns information on the status of DAP messages or data transfers. This message is sent synchronously in response to another DAP message or an error during data transfer. The format is:

STATUS	STSCODE	RFA	RECNUM	STV

where:

STATUS The OPERATOR field with TYPE = 9.

STSCODE A 2-byte status field (16 bits) subdivided as:

15 12 11 Ø

MACCODE	MICCODE

where:

MACCODE(4B):B The macro or functional group reason for the Status message. Table 2 specifies values for this field.

MICCODE(12B):B The specific type of status (by MACCODE type). Tables 3, 4, and 5 specify values for this field.

RFA(I-8) : B Returns the Record File Address of the record to which this Status message applies. If the Access message field ACCOPT indicates a return of status after each record is stored, then this field contains the record file address of the record in the accessed system's file.

RECNUM(I-8) : B Returns the record number for relative files when a Status message is returned after each record is transferred (as specified in the ACCOPT field of the Access message). The RECNUM field is null for non-relative files.

STV(I-8) : B Secondary status. Used to return secondary status information where required. (For example, RMS uses it for device error codes.)

Table 2
MACCODE Field Values

Value (Octal)	Name	Meaning
Ø	Pending	Operation in progress.
1	Successful	Returns information that indicates success.
2	Unsupported	This implementation of DAP does not support the specified request. For example, this is used when an unsupported bit/field or a field/value is encountered which a particular implementation does not support.
3		Reserved.
4	File Open	Errors that occur before a file is successfully opened.
5	Transfer Error	Errors that occur after opening a file and before closing that file.
6	Transfer Warning	For operations on open files, indicates the operation completed but not with complete success.
7	Access Termination	Errors associated with terminating access to a file.
1Ø	Format	Error in parsing a message. Format is not correct.
11	Invalid	Field of message is invalid (for example, bits that are meant to be mutually exclusive are set, an undefined bit is set, a field value is out of range or an illegal string is in a field).
12	Sync	DAP message received out of synchronization.
13-15		Reserved.
16-17		User defined STATUS MACCODES

Table 3
MICCODE Field Values for Use with MACCODE
Values of 2, 10, and 11 Octal

Type of Error	Code (Octal)	Reason
		NOTE MICCODE Format: Bits 0-5 specify the DAP message field number. Bits 6-11 specify the DAP message type number.
Miscellaneous	00 00	Unspecified DAP message error.
	00 10	DAP message type field (TYPE) error.
Configuration Message errors by field	01 00	Unknown field.
	01 10	DAP message flags field (FLAGS).
	01 11	Data stream identification field (STREAMID).
	01 12	Length field (LENGTH).
	01 13	Length extension field (LEN256).
	01 14	BITCNT field (BITCNT).
	01 20	Buffer size field (BUFSIZ).
	01 21	Operating system type field (OSTYPE).
	01 22	File system type field (FILESYS).
	01 23	DAP version number field (VERNUM).
	01 24	ECO version number field (ECONUM).
	01 25	USER protocol version number field (USRNUM).
	01 26	DEC software release number field (SOFTVER).
	01 27	User software release number field (USRSOFT).
	01 30	System capabilities field (SYSCAP).
Attributes Message errors by field	02 00	Unknown field.
	02 10	DAP message flags field (FLAGS).
	02 11	Data stream identification field (STREAMID).
	02 12	Length field (LENGTH).
	02 13	Length extension field (LEN 256).
	02 14	Bit count field (BITCNT).
	02 15	System specific field (SYSPEC).
	02 20	Attributes menu field (ATTMENU).
	02 21	Data type field (DATATYPE).
	02 22	File organization field (ORG).
	02 23	Record format field (RFM).
	02 24	Record attributes field (RAT).
	02 25	Block size field (BLS).
	02 26	Maximum record size field (MRS).
	02 27	Allocation quantity field (ALQ).
	02 30	Bucket size field (BKS).
	02 31	Fixed control area size field (FSZ).
	02 32	Maximum record number field (MRN).

(continued on next page)

Table 3 (Cont.)
MICCODE Field Values for Use with MACCODE
Values of 2, 10, and 11 Octal

Type of Error	Code (Octal)	Reason
Attributes Message errors by field (Cont.)	02 33	Run-time system field (RUNSYS).
	02 34	Default extension quantity field (DEQ).
	02 35	File options field (FOP).
	02 36	Byte size field (BSZ).
	02 37	Device characteristics field (DEV).
	02 40	Spooling device characteristics field (SDC); (reserved).
	02 41	Longest record length field (LRL).
	02 42	Highest virtual block allocated field (HBK).
	02 43	End of file block field (EBK).
	02 44	First free byte field (FFB).
	02 45	Starting LBN for contiguous file (SBN).
Access Message errors by field	03 00	Unknown field.
	03 10	DAP message flags field (FLAGS).
	03 11	Data stream identification field (STREAMID).
	03 12	Length field (LENGTH).
	03 13	Length extension field (LEN256).
	03 14	Bit count field (BITCNT).
	03 15	System specific field (SYSPEC).
	03 20	Access function field (ACCFUNC).
	03 21	Access options field (ACCOPT).
	03 22	File specification field (FILESPEC).
	03 23	File access field (FAC).
	03 24	File sharing field (SHR).
	03 25	Display attributes request field (DISPLAY).
	03 26	File password field (PASSWORD).
Control Message errors by field	04 00	Unknown field.
	04 10	DAP message flags field (FLAGS).
	04 11	Data stream identification field (STREAMID).
	04 12	Length field (LENGTH).
	04 13	Length extension field (LEN256).
	04 14	Bit count field (BITCNT).
	04 15	System specific field (SYSPEC).
	04 20	Control function field (CTLFUNC).
	04 21	Control menu field (CTLMENU).
	04 22	Record access field (RAC).
	04 23	Key field (KEY).
	04 24	Key of reference field (KRF).
	04 25	Record options field (ROP).
	04 26	Hash code field (HSH); Reserved for future use.
	04 27	Display attributes request field (DISPLAY).

(continued on next page)

Table 3 (Cont.)
MICCODE Field Values for Use with MACCODE
Values of 2, 10, and 11 Octal

Type of Error	Code (Octal)	Reason
Continue Message errors by field	05 00	Unknown field.
	05 10	DAP message flags field (FLAGS).
	05 11	Data stream identification field (STREAMID).
	05 12	Length field (LENGTH).
	05 13	Length extension field (LEN256).
	05 14	Bit count field (BITCNT).
	05 15	System specific field (SYSPEC).
	05 20	Continue transfer function field (CONFUNC).
Acknowledge Message errors by field	06 00	Unknown field.
	06 10	DAP message flags field (FLAGS).
	06 11	Data stream identification field (STREAMID).
	06 12	Length field (LENGTH).
	06 13	Length extension field (LEN256).
	06 14	Bit count field (BITCNT).
	06 15	System specific field (SYSPEC).
Access Complete Message errors by field	07 00	Unknown field.
	07 10	DAP message flags field (FLAGS).
	07 11	Data stream identification field (STREAMID).
	07 12	Length field (LENGTH).
	07 13	Length extension field (LEN256).
	07 14	Bit count field (BITCNT).
	07 15	System specific field (SYSPEC).
	07 20	Access complete function field (CMPFUNC).
	07 21	File options field (FOP).
	07 22	Checksum field (CHECK).
Data Message errors by field	10 00	Unknown field.
	10 10	DAP message flags field (FLAGS).
	10 11	Data stream identification field (STREAMID).
	10 12	Length field (LENGTH).
	10 13	Length extension field (LEN256).
	10 14	Bit count field (BITCNT).
	10 15	System specific field (SYSPEC).
	10 20	Record number field (RECNUM).
	10 21	File data field (FILEDATA).

(continued on next page)

Table 3 (Cont.)
MICCODE Field Values for Use with MACCODE
Values of 2, 10, and 11 Octal

Type of Error	Code (Octal)	Reason
Status Message errors by field	11 00	Unknown field.
	11 10	DAP message flags field (FLAGS).
	11 11	Data stream identification field (STREAMID).
	11 12	Length field (LENGTH).
	11 13	Length extension field (LEN256).
	11 14	Bit count field (BITCNT).
	11 15	System specific field (SYSPEC).
	11 20	Macro status code field (MACCODE).
	11 21	Micro status code field (MICCODE).
	11 22	Record file address field (RFA).
	11 23	Record number field (RECNUM).
	11 24	Secondary status field (STV).
Key Definition Message errors by field:	12 00	Unknown field.
	12 10	DAP message flags field (FLAGS).
	12 11	Data stream identification field (STREAMID).
	12 12	Length field (LENGTH).
	12 13	Length extension field (LEN256).
	12 14	Bit count field (BITCNT).
	12 15	System specific field (SYSPEC).
	12 20	Key definition menu field (KEYMENU).
	12 21	Key option flags field (FLG).
	12 22	Data bucket fill quantity field (DFL).
	12 23	Index bucket fill quantity field (IFL).
	12 24	Key segment repeat count field (SEGCNT).
	12 25	Key segment position field (POS).
	12 26	Key segment size field (SIZ).
	12 27	Key of reference field (REF).
	12 30	Key name field (KNM).
	12 31	Null key character field (NUL).
	12 32	Index area number field (IAN).
	12 33	Lowest level area number field (LAN).
	12 34	Data level area number field (DAN).
	12 35	Key data type field (DTP).
	12 36	Root VBN for this key field (RVB).
	12 37	Hash algorithm value field (HAL).
	12 40	First data bucket VBN field (DVB).
	12 41	Data bucket size field (DBS).
	12 42	Index bucket size field (IBS).
	12 43	Level of root bucket field (LVL).
	12 44	Total key size field (TKS).
	12 45	Minimum record size field (MRL).

(continued on next page)

Table 3 (Cont.)
MICCODE Field Values for Use with MACCODE
Values of 2, 10, and 11 Octal

Type of Error	Code (Octal)	Reason
Allocation message errors by field:	13 00	Unknown field.
	13 10	DAP message flags field (FLAGS).
	13 11	Data stream identification field (STREAMID).
	13 12	Length field (LENGTH).
	13 13	Length extension field (LEN256).
	13 14	Bit count field (BITCNT).
	13 15	System specific field (SYSPEC).
	13 20	Allocation menu field (ALLMENU).
	13 21	Relative volume number field (VOL).
	13 22	Alignment options field (ALN).
	13 23	Allocation options field (AOP).
	13 24	Starting location field (LOC).
	13 25	Related file identification field (RFI).
	13 26	Allocation quantity field (ALQ).
	13 27	Area identification field (AID).
	13 30	Bucket size field (BKZ).
	13 31	Default extension quantity field (DEQ).
Summary Message errors by field	14 00	Unknown field.
	14 10	DAP message flags field (FLAGS).
	14 11	Data stream identification field (STREAMID).
	14 12	Length field (LENGTH).
	14 13	Length extension field (LEN256).
	14 14	Bit count field (BITCNT).
	14 15	System specific field (SYSPEC).
	14 20	Summary menu field (SUMENU).
	14 21	Number of keys field (NOK).
	14 22	Number of areas field (NOA).
	14 23	Number of record descriptors field (NOR).
	14 24	Prologue version number (PVN).
Date and Time Message errors by field	15 00	Unknown field.
	15 10	DAP message flags field (FLAGS).
	15 11	Data stream identification field (STREAMID).
	15 12	Length field (LENGTH).
	15 13	Length extension field (LEN256).
	15 14	Bit count field (BITCNT).
	15 15	System specific field (SYSPEC).
	15 20	Date and time menu field (DATMENU).
	15 21	Creation date and time field (CDT).
	15 22	Last update date and time field (RDT).
	15 23	Deletion date and time field (EDT).
	15 24	Revision number field (RVN).

(continued on next page)

Table 3 (Cont.)
MICCODE Field Values for Use with MACCODE
Values of 2, 10, and 11 Octal

Type of Error	Code (Octal)	Reason
Protection Message errors by field	16 00	Unknown field.
	16 10	DAP message flags field (FLAGS).
	16 11	Data stream identification field (STREAMID).
	16 12	Length field (LENGTH).
	16 13	Length extension field (LEN256).
	16 14	Bit count field (BITCNT).
	16 15	System specific field (SYSPEC).
	16 20	Protection menu field (PROTMENU).
	16 21	File owner field (OWNER).
	16 22	System protection field (PROTSYS).
	16 23	Owner protection field (PROTOWN).
	16 24	Group protection field (PROTGRP).
	16 25	World protection field (PROTWLD).
Name Message errors by field	17 00	Unknown field.
	17 10	DAP message flags field (FLAGS).
	17 11	Data stream identification field (STREAMID).
	17 12	Length field (LENGTH).
	17 13	Length extension field (LEN256).
	17 14	Bit count field (BITCNT).
	17 15	System specific field (SYSPEC).
	17 20	Name type field (NAMETYPE).
	17 21	Name field (NAMESPEC).
Access Control List Message errors by field: (Reserved for future use)	20 00	Unknown field.
	20 10	DAP message flags field (FLAGS).
	20 11	Data stream identification field (STREAMID).
	20 12	Length field (LENGTH).
	20 13	Length extension field (LEN256).
	20 14	Bit count field (BITCNT).
	20 15	System specific field (SYSPEC).
	20 20	Access control list repeat count field (ACLCNT).
	20 21	Access control list entry field (ACL).

Table 4
MICCODE Field Values for Use with MACCODE
Values of Ø, 1, 4, 5, 6, and 7 Octal

Value (Octal)	Error/Reason
	NOTE MICCODE Format: Bits Ø-11 contain the error code number. Symbolic status codes, where supplied, refer to the corresponding RMS status codes. These codes are included here for ease of reference only. They have no meaning for DAP.
Ø	Unspecified error.
1	ER$ABO; Operation aborted.
2	ER$ACC; Fll-ACP could not access file.
3	ER$ACT; FILE activity precludes operation.
4	ER$AID; Bad area ID.
5	ER$ALN; Alignment options error.
6	ER$ALQ; Allocation quantity too large or Ø value.
7	ER$ANI; Not ANSI D format.
1Ø	ER$AOP; Allocation options error.
11	ER$AST; Invalid (i.e., synch) operation at AST level.
12	ER$ATR; Attribute read error.
13	ER$ATW; Attribute write error.
14	ER$BKS; Bucket size too large.
15	ER$BKZ; Bucket size too large.
16	ER$BLN; BLN length error.
17	ER$BOF; Beginning of file detected.
2Ø	ER$BPA; Private pool address.
21	ER$BPS; Private pool size.
22	ER$BUG; Internal RMS error condition detected.
23	ER$CCR; Cannot connect RAB.
24	ER$CHG; $UPDATE changed a key without having attribute of XB$CHG set.
25	ER$CHK; Bucket format check-byte failure.
26	ER$CLS; RSTS/E close function failed.
27	ER$COD; Invalid or unsupported COD field.
3Ø	ER$CRE; Fll-ACP could not create file (STV=sys err code).
31	ER$CUR; No current record (operation not preceded by GET/FIND).
32	ER$DAC; Fll-ACP deaccess error during CLOSE.
33	ER$DAN; Data AREA number invalid.
34	ER$DEL; RFA-Accessed record was deleted.
35	ER$DEV; Bad device, or inappropriate device type.
36	ER$DIR; Error in directory name.
37	ER$DME; Dynamic memory exhausted.
4Ø	ER$DNF; Directory not found.
41	ER$DNR; Device not ready.
42	ER$DPE; Device has positioning error.
43	ER$DTP; DTP field invalid.
44	ER$DUP; Duplicate key detected, XB$DUP not set.
45	ER$ENT; RSX-FllACP enter function failed.
46	ER$ENV; Operation not selected in ORG$ macro.
47	ER$EOF; End-of-file.
5Ø	ER$ESS; Expanded string area too short.
51	ER$EXP; File expiration date not yet reached.

(continued on next page)

Table 4 (Cont.)
MICCODE Field Values for Use with MACCODE
Values of 0, 1, 4, 5, 6, and 7 Octal

Value (Octal)	Error/Reason
52	ER$EXT; File extend failure.
53	ER$FAB; Not a valid FAB (BID NOT = FB$BID).
54	ER$FAC; Illegal FAC for REC-OP,0, or FB$PUT not set for CREATE.
55	ER$FEX; File already exists.
56	ER$FID; Invalid file I.D.
57	ER$FLG; Invalid flag-bits combination.
60	ER$FLK; File is locked by other user.
61	ER$FND; RSX-F11ACP FIND function failed.
62	ER$FNF; File not found.
63	ER$FNM; Error in file name.
64	ER$FOP; Invalid file options.
65	ER$FUL; DEVICE/FILE full.
66	ER$IAN; Index AREA number invalid.
67	ER$IFI; Invalid IFI value or unopened file.
70	ER$IMX; Maximum NUM(254) areas/key XABS exceeded.
71	ER$INI; $INIT macro never issued.
72	ER$IOP; Operation illegal or invalid for file organization.
73	ER$IRC; Illegal record encountered (with sequential files only).
74	ER$ISI; Invalid ISI value, on unconnected RAB.
75	ER$KBF; Bad KEY buffer address (KBF=0).
76	ER$KEY; Invalid KEY field (KEY=0/neg).
77	ER$KRF; Invalid key-of-reference ($GET/$FIND).
100	ER$KSZ; KEY size too large.
101	ER$LAN; Lowest-level-index AREA number invalid.
102	ER$LBL; Not ANSI labeled tape.
103	ER$LBY; Logical channel busy.
104	ER$LCH; Logical channel number too large.
105	ER$LEX; Logical extend error, prior extend still valid.
106	ER$LOC; LOC field invalid.
107	ER$MAP; Buffer mapping error.
110	ER$MKD; F11-ACP could not mark file for deletion.
111	ER$MRN; MRN value=neg or relative key>MRN.
112	ER$MRS; MRS value=0 for fixed length records. Also 0 for relative files.
113	ER$NAM; NAM block address invalid (NAM=0, or not accessible).
114	ER$NEF; Not positioned to EOF (sequential files only).
115	ER$NID; Cannot allocate internal index descriptor.
116	ER$NPK; Indexed file; no primary key defined.
117	ER$OPN; RSTS/E open function failed.
120	ER$ORD; XAB'S not in correct order.
121	ER$ORG; Invalid file organization value.
122	ER$PLG; Error in file's prologue (reconstruct file).
123	ER$POS; POS field invalid (POS>MRS,STV=XAB indicator).
124	ER$PRM; Bad file date field retrieved.
125	ER$PRV; Privilege violation (OS denies access).
126	ER$RAB; Not a valid RAB (BID NOT=RB$BID).
127	ER$RAC; Illegal RAC value.
130	ER$RAT; Illegal record attributes.

(continued on next page)

Table 4 (Cont.)
MICCODE Field Values for Use with MACCODE
Values of Ø, 1, 4, 5, 6, and 7 Octal

Value (Octal)	Error/Reason
131	ER$RBF; Invalid record buffer address (ODD, or not word-aligned if BLK-IO).
132	ER$RER; File read error.
133	ER$REX; Record already exists.
134	ER$RFA; Bad RFA value (RFA=Ø).
135	ER$RFM; Invalid record format.
136	ER$RLK; Target bucket locked by another stream.
137	ER$RMV; RSX-F11 ACP remove function failed.
140	ER$RNF; Record not found.
141	ER$RNL; Record not locked.
142	ER$ROP; Invalid record options.
143	ER$RPL; Error while reading prologue.
144	ER$RRV; Invalid RRV record encountered.
145	ER$RSA; RAB stream currently active.
146	ER$RSZ; Bad record size (RSZ>MRS, or NOT=MRS if fixed length records).
147	ER$RTB; Record too big for user's buffer.
150	ER$SEQ; Primary key out of sequence (RAC=RB$SEQ for $PUT).
151	ER$SHR; SHR field invalid for file (cannot share sequential files).
152	ER$SIZ; SIZ field invalid.
153	ER$STK; Stack too big for save area.
154	ER$SYS; System directive error.
155	ER$TRE; Index tree error.
156	ER$TYP; Error in file type extension on FNS too big.
157	ER$UBF; Invalid user buffer addr (Ø, ODD, or if BLK-IO not word aligned).
160	ER$USZ; Invalid user buffer size (USZ=Ø).
161	ER$VER; Error in version number.
162	ER$VOL; Invalid volume number.
163	ER$WER; File write error (STV=sys err code).
164	ER$WLK; Device is write locked.
165	ER$WPL; Error while writing prologue.
166	ER$XAB; Not a valid XAB (@XAB=ODD,STV=XAB indicator).
167	BUGDDI; Default directory invalid.
170	CAA ; Cannot access argument list.
171	CCF ; Cannot close file.
172	CDA ; Cannot deliver AST.
173	CHN ; Channel assignment failure (STV=sys err code).
174	CNTRLO; Terminal output ignored due to (CNTRL) O.
175	CNTRLY; Terminal input aborted due to (CNTRL) Y.
176	DNA ; Default filename string address error.
177	DVI ; Invalid device I.D. field.
200	ESA ; Expanded string address error.
201	FNA ; Filename string address error.
202	FSZ ; FSZ field invalid.
203	IAL ; Invalid argument list.
204	KFF ; Known file found.
205	LNE ; Logical name error.
206	NOD ; Node name error.
207	NORMAL; Operation successful.
210	OK_DUP; Record inserted had duplicate key.

(continued on next page)

Table 4 (Cont.)
MICCODE Field Values for Use with MACCODE
Values of Ø, 1, 4, 5, 6 and 7 Octal

Value (Octal)	Error/Reason
211	OK_IDX; Index update error occurred-record inserted.
212	OK_RLK; Record locked but read anyway.
213	OK_RRV; Record inserted in primary o.k.; may not be accessible by secondary keys or RFA.
214	CREATE; File was created, but not opened.
215	PBF ; Bad prompt buffer address.
216	PNDING; Async. operation pending completion.
217	QUO ; Quoted string error.
220	RHB ; Record header buffer invalid.
221	RLF ; Invalid related file.
222	RSS ; Invalid resultant string size.
223	RST ; Invalid resultant string address.
224	SQO ; Operation not sequential.
225	SUC ; Operation successful.
226	SPRSED; Created file superseded existing version.
227	SYN ; Filename syntax error.
230	TMO ; Time-out period expired.
231	ER$BLK; FB$BLK record attribute not supported.
232	ER$BSZ; Bad byte size.
233	ER$CDR; Cannot disconnect RAB.
234	ER$CGJ; Cannot get JFN for file.
235	ER$COF; Cannot open file.
236	ER$JFN; Bad JFN value.
237	ER$PEF; Cannot position to end-of-file.
240	ER$TRU; Cannot truncate file.
241	ER$UDF; File is currently in an undefined state; access is denied.
242	ER$XCL; File must be opened for exclusive access.
243	; Directory full.
244	; Handler not in system.
245	; Fatal hardware error.
246	; Attempt to write beyond EOF.
247	; Hardware option not present.
250	; Device not attached.
251	; Device already attached.
252	; Device not attachable.
253	; Sharable resource in use.
254	; Illegal overlay request.
255	; Block check or CRC error.
256	; Caller's nodes exhausted.
257	; Index file full.
260	; File header full.
261	; Accessed for write.
262	; File header checksum failure.
263	; Attribute control list error.
264	; File already accessed on LUN.
265	; Bad tape format.
266	; Illegal operation on file descriptor block.
267	; Rename; 2 different devices.
270	; Rename; new filename already in use.
271	; Cannot rename old file system.
272	; File already open.
273	; Parity error on device.

(continued on next page)

Table 4 (Cont.)
MICCODE Field Values for Use with MACCODE
Values of 0, 1, 4, 5, 6, and 7 Octal

Value (Octal)	Error/Reason
274	; End of volume detected.
275	; Data over-run.
276	; Bad block on device.
277	; End of tape detected.
300	; No buffer space for file.
301	; File exceeds allocated space -- no blks.
302	; Specified task not installed.
303	; Unlock error.
304	; No file accessed on LUN.
305	; Send/receive failure.
306	SPL ; Spool or submit command file failure.
307	NMF ; No more files.
310	CRC ; DAP file transfer checksum error.
311	; Quota exceeded
312	BUGDAP; Internal network error condition detected.
313	CNTRLC; Terminal input aborted due to (CNTRL) C.
314	DFL ; Data bucket fill size > bucket size in XAB.
315	ESL ; Invalid expanded string length.
316	IBF ; Illegal bucket format.
317	IBK ; Bucket size of LAN NOT = IAN in XAB.
320	IDX ; Index not initialized.
321	IFA ; Illegal file attributes (corrupt file header).
322	IFL ; Index bucket fill size > bucket size in XAB.
323	KNM ; Key name buffer not readable or writeable in XAB.
324	KSI ; Index bucket will not hold two keys for key of reference.
325	MBC ; Multi-buffer count invalid (negative value).
326	NET ; Network operation failed at remote node.
327	OK_ALK; Record is already locked.
330	OK_DEL; Deleted record successfully accessed.
331	OK_LIM; Retrieved record exceeds specified key value.
332	OK_NOP; Key XAB not filled in.
333	OK_RNF; Nonexistent record successfully accessed.
334	PLV ; Unsupported prologue version.
335	REF ; Illegal key-of-reference in XAB.
336	RSL ; Invalid resultant string length.
337	RVU ; Error updating rrv's, some paths to data may be lost.
340	SEG ; Data types other than string limited to one segment in XAB.
341	; Reserved
342	SUP ; Operation not supported over network.
343	WBE ; Error on write behind.
344	WLD ; Invalid wildcard operation.
345	WSF ; Working set full (can not lock buffers in working set).
346	; Directory listing--error in reading volume-set name, directory name, of file name.
347	; Directory listing--error in reading file attributes.
350	; Directory listing--protection violation in trying to read the volume-set, directory or file name.

(continued on next page)

Table 4 (Cont.)
MICCODE Field Values for Use with MACCODE
Values of 0, 1, 4, 5, 6, and 7 Octal

Value (Octal)	Error/Reason
351	; Directory listing--protection violation in trying to read file attributes.
352	; Directory listing--file attributes do not exist.
353	; Directory listing--unable to recover directory list after Continue Transfer (Skip).
354	SNE ; Sharing not enabled.
355	SPE ; Sharing page count exceeded.
356	UPI ; UPI bit not set when sharing with BRO set.
357	ACS ; Error in access control string (poor man's route through error).
360	TNS ; Terminator not seen.
361	BES ; Bad escape sequence.
362	PES ; Partial escape sequence.
363	WCC ; Invalid wildcard context value.
364	IDR ; Invalid directory rename operation.
365	STR ; User structure (FAB/RAB) became invalid during operation.
366	FTM ; Network file transfer mode precludes operation.
6000 to 7777	; User defined errors

Table 5
MICCODE Field Values
(with MACCODE Value of 12 Octal)

Value (Octal)	Error/Reason
	NOTE MICCODE Format: Bits 0-11 contain the message type number.
0	Unknown message type
1	Configuration message
2	Attributes message
3	Access message
4	Control message
5	Continue Transfer message
6	Acknowledge message
7	Access Complete message
10	Data message
11	Status message
12	Key Definition Attributes Extension message
13	Allocation Attributes Extension message
14	Summary Attributes Extension message
15	Date and Time Attributes Extension message
16	Protection Attributes Extension message
17	Name message
20	Access Control List Extended Attributes message

3.12 Key Definition Attributes Extension Message

Each key is defined by a group of 19 fields. The form of the key definition message is:

KEYDEF	KEYMENU	FLG	DFL	IFL	NSG	POS	SIZ	[POS	SIZ...]	REF	KNM

NUL	IAN	LAN	DAN	DTP	RVB	HAL	DVB	DBS	IBS	LVL	TKS	MRL

KEYDEF : The operator field with TYPE = 10.

KEYMENU(EX-6) : BM The following bit map specifies which of the following fields are present in this message. These fields and only these fields may appear in the message and they must be in the order specified.

Bit	Meaning (When Set)
0	FLG
1	DFL
2	IFL
3	NSG, POS and SIZ
4	REF
5	KNM
6	NUL
7	IAN
8	LAN
9	DAN
10	DTP
11	RVB
12	HAL (reserved)
13	DVB
14	DBS
15	IBS
16	LVL
17	TKS
18	MRL

FLG(EX-3) : BM Key option flag

bit 0 = XB$DUP duplicates allowed
bit 1 = XB$CHG allow keys to change
bit 2 = XB$NUL null key character defined

DFL(2) : B Data bucket fill.

IFL(2) : B Index bucket fill.

NSG(1) : B Number of segments (max. 8) needed to define the key in the record. For each segment, there will be a POS, SIZ pair. For example, if NSG contains 3, the following sequence of fields would appear:

NSG	POS	SIZ	POS	SIZ	POS	SIZ

POS(2) : B Position of key in record by byte number. The first byte in the record is byte 0.

SIZ(1) : B Size of key in record in bytes.

REF(1) : B Key of reference indicator.

KNM(I-40):A — Key name correspoding to key of reference (REF) above.

NUL(1) : B — Value of null key character.

IAN(1) : B — Index area number.

LAN(1) : B — Lowest level index area number.

DAN(1) : B — Data level area number.

DTP(1) : B — Data type.

RVB(I-8) : B — Root VBN for key.

HAL(I-5) : B — Hash algorithm value (reserved).

DVB(I-8) : B — First data bucket start virtual block number.

DBS(1) : B — Data bucket size field.

IBS(1) : B — Index bucket size field.

LVL(1) : B — Level of root bucket.

TKS(1) : B — Total key size.

MRL(2) : B — Minimum record length; the minimum record length in bytes which will totally contain the key field for the key described by this message.

3.13 Allocation Attributes Extension Message

Use the Allocation message when creating or explicitly extending a file to specify the character of the allocation. The form of the allocation message is:

ALLOC	ALLMENU	VOL	ALN	AOP	LOC	RFI	ALQ	AID	BKZ	DEQ

ALLOC : — The operator field with TYPE = 11.

ALLMENU(EX-6): BM — The following bit map specifies which of the following fields are present in this message. These fields and only these fields may appear in the message. They must be in the order specified.

Bit	Meaning (When Set)
0	VOL
1	ALN
2	AOP
3	LOC
4	RFI (reserved)
5	ALQ
6	AID
7	BKZ
8	DEQ

VOL(2) : B — Relative volume number of a volume set on which this area or file will be allocated.

ALN(EX-4) : BM — Alignment options

Bit	Meaning	Explanation
Ø	XB$ANY	No specified allocation placement.
1	XB$CYL	Align on cylinder boundary.
2	XB$LBN	Align to specified logical block.
3	XB$VBN	Allocate as near as possible to specified virtual block.
4	XB$RFI	Allocate as near as possible to specified related file.

AOP(EX-4) : BM — Allocation options

Bit	Meaning	Explanation
Ø	XB$HRD	If the requested alignment can not be done, return an error.
1	XB$CTG	Contiguous allocation required.
2	XB$CBT	Contiguous best try. The file will be created, but it will be contiguous only when it is possible.
3	XB$ONC	Allocate space on any cylinder boundary.

LOC(I-8) : B — Allocation starting point. Value is as follows:

If ALN = XB$CYL, LOC is the cylinder number where allocation is to start.
If ALN = XB$LBN, LOC is the logical block where allocation is to start.
If ALN = XB$VBN, LOC is the virtual block near which allocation should start; Ø (default) means as near as possible to current EOF.
If ALN = XB$RFI, LOC is a VBN but a VBN in the related file.

RFI(I-16) : B — Related file I.D. (reserved).

ALQ(I-5) : B — The amount of space in virtual blocks to be allocated. Returns actual size of allocation. For DISPLAY, returns size of area.

AID(1) : B — Area I.D. The area number used to identify an area for reference by a key definition.

BKZ(1) : B — Bucket size for this area.

DEQ(2) : B — The default extension quantity in virtual blocks. Specifies the amount of space to be added to the area whenever it is extended automatically. It overrides the DEQ in the main Attributes message.

3.14 Summary Attributes Extension Message

The Summary Attributes Extension message comprises the fields described below. The accessed process optionally returns this message to the accessing process on a file open, file create, display of a file's attributes or directory list. The format of the Summary Attributes Extension message is:

SUMMARY	SUMENU	NOK	NOA	NOR	PVN

where:

SUMMARY : The operator field with TYPE = 12.

SUMENU(EX-6) : BM The following bit map specifies which of the following fields are present in this message. These fields and only these fields may appear in the message and they must appear in the order specified.

Bit	Meaning (When Set)
Ø	NOK
1	NOA
2	NOR
3	PVN

NOK(1) : B Number of keys defined in file.

NOA(1) : B Number of areas defined in file.

NOR(1) : B Number of record descriptors in file.

PVN(2) : B Prologue version number.

3.15 Date and Time Attributes Extension Message

The Date and Time Attributes Extension message is composed of the fields described below. This message is optionally returned to the accessing process by the accessed process on a file open, file create, display of a file's attributes or directory list. The form of the message is:

DATIME	DATMENU	CDT	RDT	EDT	RVN

where:

DATIME : The operator field with TYPE = 13.

DATMENU(EX-6) : BM The following bit map specifies which of the following fields are present in this message. These fields and only these fields may appear in the message and they must appear in the order specified.

Bit	Meaning (When Set)
Ø	CDT
1	RDT
2	EDT
3	RVN

CDT(18) : A — Date and time file created in Network Standard Time (NST).

RDT(18) : A — Date and time file last updated in NST.

EDT(18) : A — Date and time file may be deleted in NST.

The preceding three fields should be in the following format:

dd-mon-yybhh:mm:ss

where:

dd is the day
mon is a three letter abbreviation for the month as follows:

JAN
FEB
MAR
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC

yy is the year
b is blank (space)
hh is the hour
mm is the minutes
ss is the seconds

RVN(2) : B — Revision number--the number of times the file has been modified.

Network standard time (NST) is the time standard chosen for each network to effect synchronization of remote file access operations and their results. For example, if a user in London creates a file in New York with a deletion time of 2 a.m., does he mean 2 a.m. in London or New York? NST resolves this problem. NST may be local time (for example, for networks wholly contained in one time zone), GMT or some other agreed on time.

3.16 Protection Attributes Extension Message

Use the Protection Attributes Extension message when creating a file to specify the protection for that file. If this information is not present when creating a new file, the file will be created with the default protection of the accessed node. This message may also be used to return a file's protection code to the accessing process on a file open, display of the file's attributes or directory list. The format of the Protection Attributes Extension message is:

PROTECT	PROTMENU	OWNER	PROTSYS	PROTOWN	PROTGRP	PROTWLD

where:

PROTECT : The OPERATOR field with TYPE = 14.

PROTMENU(EX-6) : BM The following bit map specifies which of the following fields are present in this message. If the field is not present, the default, if any, should be used. These fields and only these fields may appear in the message and they must appear in the order specified.

Bit	Meaning (When Set)
Ø	OWNER
1	PROTSYS
2	PROTOWN
3	PROTGRD
4	PROTWLD

OWNER(I-4Ø) : A The name or user code (for example, UIC such as 24Ø,22Ø) of the file owner. When creating a file and when the file system allows the owner to be user-specified, this field, if present, specifies the owner of the file. If this field is not present, the file owner information is taken from the user identification information which comes with the connect. Alternatively, if the User Identification message is being used (see Appendix A) owner information may be taken from it.

PROTSYS(EX-3) : BM File protection for system access rights.

Bit	Meaning (When Set)
Ø	Deny read access.
1	Deny write access.
2	Deny execute access.
3	Deny delete access.
4	Deny append access.
5	Deny directory list access.
6	Deny update access.
7	Deny change access protection attribute.
8	Deny extend access.

PROTOWN(EX-3) : BM File protection for file owner access rights. Refer to the bit map used for PROTSYS above.

PROTGRP(EX-3) : BM File protection for group access rights. Refer to the bit map used for PROTSYS above.

PROTWLD(EX-3) : BM File protection for general (world) access. Refer to the bit map used for PROTSYS above.

3.17 **Name Message**

Use the Name message when renaming a file to contain the new name the file will have after the operation is complete. Use the Name message with the directory listing to return the name of each file for which attributes are returned. It may also be used when opening or creating a file to contain the default or related file specification. The form of the name message is:

NAME	NAMETYPE	NAMESPEC

where:

NAME: The operator field with TYPE = 15.

NAMETYPE(EX-3) : BM Type of name contained in NAMESPEC field.

Bit	Meaning
Ø	File specification
1	File name
2	Directory name
3	Volume or structure name
4	Default file specification (reserved)
5	Related file specification (reserved)

NOTE

Here the file specification means the full file specification including the volume, directory, and file name.

NAMESPEC(I-2ØØ) : A The file specification in the format of the remote node. This ASCII field is not interpreted by DAP software.

3.18 **Access Control List Attributes Extension Message (Reserved for Future Use)**

The Access Control List (ACL) message specifies a list of users and the access rights they have to this file. Each ACL entry is in the format of the system where the file resides. The list is potentially very long and therefore this message has a facility for specifying if this is the last of a sequence of ACL messages. The form of the ACL message is:

ACLTYPE	ACLCNT	ACL	[ACL...]

ACLTYPE : The operator field with TYPE = 16.

ACLCNT(1) : B The absolute value of this field is the number of repetitions of the ACL field in this message. If this field contains a negative value, this is the last in a sequence of ACL messages.

ACL(I-8Ø) : A Access control list entry. There is one entry for each user having access to the file. This field is treated as a literal string (not parsed by accessing system) and is in the format of the accessed system.

4.Ø FILE ORGANIZATION

4.1 Types of Files

The following types of files are addressed by this specification:

- **Sequential.** Each record's position depends on the position of the previous record.
- **Relative.** Each record in the file has a unique identifying number, its record number. Records may be accessed randomly by specifying their record number in a Control message.
- **Hashed/Indexed.** These files have records organized according to some classification method, usually an access key. Within a particular key for indexed files, the records are assumed to be logically sequential.

4.2 Record Formats and Attributes

There are two ways in which ASCII records are stored in DIGITAL file systems:

1. **Byte Count.** A byte count associated with the record in the file indicates how long the record is and is used to determine record boundaries.
2. **Stream.** The ASCII record is stored exactly as is. The record is assumed to be terminated with one of the following delimiters:

   a. (FF) -- Form feed
   b. (DLE) -- Data link escape
   c. (DC1)
   d. (DC2)
   e. (DC3)
   f. (DC4)
   g. (VT) -- Vertical tab
   h. (LF) -- Line feed
   i. (ESC) -- Escape
   j. (^Z) -- Control Z

   Files using ASCII stream often do not have record attributes stored with the file.

DAP supports four ASCII record formats:

1. Fixed length records
2. Variable length records
3. Variable with fixed control
4. ASCII stream

In addition, DAP supports the following eight attributes:

1. FORTRAN carriage control
2. COBOL carriage control
3. Print file carriage control
4. Implied LF/CR envelope for printing
5. Embedded carriage control
6. Line sequential ASCII
7. MACY11
8. None of the above

4.2.1 **Handling Stream ASCII** -- Stream ASCII files consist of a string of ASCII characters with no explicit record structure imposed upon the data within the file. Records in a stream ASCII file are defined by a set of delimiters (see Section 4.2) where each record is terminated by one of the delimiters. DAP processes transferring stream ASCII data records perform no transformations upon the records. All characters in the records, up to and including the delimiter, are sent as a single record in a DAP Data message. No characters, including NULLs, are stripped or replaced. Stream ASCII files may have other attributes as well as stream ASCII, for example, FORTRAN carriage control.

4.2.2 **Conversions** -- DAP does not specify data or format conversions that may be necessary when sending data from one type of system to another. It is the responsibility of the user process (accessing process) to make any necessary conversions when transferring data between unlike systems. The accessed process (server) is non-intelligent and does only what it is told. The accessed process executes a protocol function only if it supports that type of operation or attribute. If an accessed process does not support an operation or attribute, it returns an appropriate Status message. Thus, if a conversion must be made, for example, sending a Stream ASCII file to a remote system supporting variable length byte count records but not Stream ASCII, the accessing process (user) must perform the conversion before sending each record of the file.

NOTE

While this is not a part of the DAP specification, the following algorithm has been used effectively in several DAP-speaking user processes to determine what (if any) conversions must be made when trying to store a file on a remote file system:

1. Open file using file's attributes on local system.

2. If no error on Attributes message, send file and terminate, else go to step 3.

3. Send new Attributes and Access Complete messages using the next most likely attributes to be supported by the remote system.

4. If no error on Attributes message, send file and terminate, else go to step 3.

   This should almost never require more than two repetitions of step 3 as most ASCII files are either stream or variable with implied CR/LF.

4.3 Data Formats

4.3.1 **Fixed-Length Records** - All records are of the fixed length specified in the MRS field. They are delimited by physical message blocks (that is, the last byte in a Data message is the end of the record).

4.3.2 **Variable-Length Records** - These records are like the fixed-length records except the record length is variable with the maximum length being specified in the MRS field.

4.3.3 **Variable with Fixed-Control Format Records** - These records are normal variable-length records with an associated fixed-length field used for control purposes. In DAP Data messages, this fixed-length control field immediately precedes and is contiguous with the variable part of the record. The length of the fixed field is found in the FSZ field in the Attributes message. MRS contains the maximum length of the variable portion only. FSZ + MRS = total maximum record length. Regardless of the type of the data in the variable portion of the record, the data in the fixed portion is always sent as a binary field contained in an integral number of 8-bit bytes.

4.3.4 **ASCII Stream** - Here a file contains just a stream of ASCII characters with no real concept of records. However, delimiters are used to terminate "records" for purposes of reading and writing the file.

4.4 Supported Data Types

The DATATYPE field of the Attributes message defines the code representation used to transfer data. These are:

ASCII
Image
EBCDIC (Reserved)

In addition, there is a COMPRESSION option mode, which can be used with any of the above. This compressed mode reduces the amount of data sent by encoding blanks and duplicates.

4.4.1 **ASCII** - This is the 7-bit ASCII code-set as defined in the 1968 ANSI standard. To transmit this within 8-bit frames, the high order bit is ignored except when using 7-bit compression. On card readers, this refers to the ASCII encoding of the punches into the 128 character codes.

4.4.2 **EBCDIC** - this is an IBM 8-bit EBCDIC code set. (Reserved)

4.4.3 **Image** - This is a mode for transmitting data in DAP where no code set is specified and data records (or blocks) are simply regarded as ordered strings of bits. The number of bits in a bit string (image mode record) need not be a multiple of 8 although the bit strings are in fact transmitted in 8-bit frames (a requirement of lower level protocols).

When transmitting Image data, if the number of data bits in the message is not a multiple of 8 (the number of actual data bits in a DAP Data message must be a multiple of the byte size as specified by BSZ in the Attributes message), the field BITCNT in the header of the Data message must be used. The BITCNT field specifies the number of bits in the final 8-bit frame of the Data message which are not actual data bits. The non-data bits are padding bits required to fill out the 8-bit frame.

When transferring bit strings where the bit count is a multiple of 8, it is not necessary to use BITCNT.

The order in the bit strings is low order bit of low order byte first, second bit of low byte, and so on, followed by the same sequence for successively higher order bytes. For example, consider retrieving the 3 byte record with 6-bit bytes shown below. The order of the string is as shown below with the first bit on the right and the last bit in the string on the left (the top row of single digit numbers inside the boxes is byte numbers and the bottom row is bit numbers. Thus for any bit position in the string, the top digit gives the byte and the bottom digit the bit within that byte).

byte 2	byte 1	byte Ø
2 2 2 2 2 2 5 4 3 2 1 Ø	1 1 1 1 1 1 5 4 3 2 1 Ø	Ø Ø Ø Ø Ø Ø 5 4 3 2 1 Ø

In Image record mode, this string transmits in 3 8-bit frames in a DAP Data message as follows:

frame 2	frame 1	frame Ø
x x x x x x 2 2 5 4	2 2 2 2 1 1 1 1 3 2 1 Ø 5 4 3 2	1 1 Ø Ø Ø Ø Ø Ø 1 Ø 5 4 3 2 1 Ø

with frame Ø being transmitted first, low order bit first. The x's are padding. BITCNT is 6 for this DAP Data message.

If this image record is being sent to a system which stores data in 7-bit bytes, this record is stored as follows:

byte 2	byte 1	byte Ø
x x x 2 2 2 2 5 4 3 2	2 2 1 1 1 1 1 1 Ø 5 4 3 2 1	1 Ø Ø Ø Ø Ø Ø Ø 5 4 3 2 1 Ø

4.4.4 **Compression** - Any of the encodings, even the Image mode, may be compressed on transmission to eliminate the sending of duplicates and multiple blanks. The compression technique is slightly different with 7-bit and 8-bit character encodings.

7-bit compression:

1. Send non-compressed characters with high order bit = Ø and low 7-bits = character: (Ø) (7-bit char).

2. Send blanks as high bits = 1Ø followed by number of blanks in 6-bit binary: (1Ø) (number blanks).

3. Send repetitions as high bits = 11Ø followed by number of repetitions in 5-bit binary: (11Ø) (number reps) followed by the repeated character.

4. An escape scheme allows sending of 12-bit card images when they do not encode into the ASCII code set by sending high bits = 1111 followed by columns 12-1 and another character of columns 2-9. For example, send (1111) (12,11,Ø,1) in the first character and (2,3,4,5,6,7,8,9) in the second character.

5. The high bit sequence 1110 has been reserved for future expansion.

6. In the case of repetitions the repeated character may be a 12-bit image which is sent as 2-characters.

7. Summary 7-bit compression:

0xxxxxxx	7-bit non-compressed characters
10xxxxxx	number of blanks (the character octal 40)
110xxxxx	number of repeated characters
1111xxxx	12-bit card image
xxxxxxxx	12-bit card image (continued)
1110xxxx	reserved

8-bit compression:

With 8-bit codes the blanks, repetitions and 12-bit card image are the same as the 7-bit case. Precede 8-bit non-compressed strings by a count of the string length with high bit = 0 followed by 7-bit count: (0) (count). This is followed by "count" 8-bit characters. Following this may be another string, blanks or repetitions.

Summary:

0xxxxxxx	number of non-compressed characters
10xxxxxx	number of nulls (all zero bytes)
110xxxxx	number of repetitions
1110xxxx	reserved
1111xxxx	12-bit card image
xxxxxxxx	12-bit card image (continued)

The receiver of compressed data must be able to expand it according to the rules just given. Do not use data compression unless the Configuration messages from both systems indicate they both support file compression (bit 15). The compressed format bit of the DATATYPE field in the Attributes message indicates the use of file compression.

5.Ø OPERATION

DAP transfers data to and from I/O devices and mass storage files independent of the I/O structure of the system being accessed. This transfer is accomplished by communication with a DAP process that accepts DAP requests on the network side and translates them into equivalent requests to the local I/O system. From the network it appears as if DECnet systems support DAP messages directly within their file systems.

DAP provides the mechanism for setting up the conversation path for remote file access, transferring data over the link, and terminating the logical link.

5.1 Setting up the Link

Processes that implement the DAP protocol operate at the application level within a DECnet system. They use the lower level interprocess communication facilities of the network for the creation and flow control of the data path (logical link) between the processes exchanging messages within the DAP environment. Once the link is established, the processes may exchange DAP messages over the link.

Each remote file access in progress uses a separate logical link. This means a single user cannot access more than one file at a time over a single logical link. It also means that several users cannot access the same file over a single logical link. However, a single logical link can be used to access more than one file provided access to the files is sequential and does not overlap.

A lower level interprocess communication protocol passes access control information (user identification, password, and account identification) from the accessing (user) DAP process to the accessed (server) DAP process.

Following link establishment, the DAP-speaking processes exchange Configuration messages. The purposes of this exchange are:

1. To establish the maximum buffer size for exchanging lower level protocol messages

2. To identify system type to each other

3. To enable one DAP process to know which version of the protocol the other DAP process speaks

4. To inform the opposite process of the generic capabilities of the system sending the message

The system type is used when it is necessary to know the type of both the operating system and the file system on the other end of the link. This is helpful, for example, in deciding if block mode file transfer can be used when transferring files between like systems or if multiple data streams can be initiated as is possible between RMS-based systems.

The capabilities field of the Configuration message indicates to each of the DAP processes the generic capabilities of the other DAP process with which it is communicating. This field determines the type of file support offered by a remote system without resorting to trial and error techniques.

The node where the file or device resides is the accessed node while the node where the user process is located is the accessing node. The accessing process initiates the connection. For each DAP message sent over the link, a transmit request and corresponding receive request must be issued to Session Control (DNA Session Control Functional Specification). This document explains the DAP messages only and assumes that the necessary receives and transmits are issued by the processes involved.

After link creation and the exchange of Configuration messages, the accessing process sends an Attributes message, optionally followed by one or more Extended Attributes messages, specifying the mode and format of the data and the structure of the file. This is then followed by an Access message specifying the desired operation. The Access message may be preceded by a Name message if the default filename option is required. The Attributes, Extended Attributes, Name, and Access messages may be blocked and sent together in one transmission if buffer space is available (the LENGTH field must be used) and if blocking is supported as indicated by the Configuration message. Alternately, if a DAP message is too long to be sent as a single entity due to limitations of the underlying Transport mechanism, the DAP message may be segmented and sent in two or more pieces (see FLAGS field in message header) if segmentation is supported.

Systems not retaining the file attributes use the Attributes message to set the attributes for the transfer. When creating a new file, the Attributes message sent by the accessing process specifies the attributes the new file should have. If the accessed system does not support these attributes, it returns a Status message to that effect. When storing records with systems retaining attributes, the accessing system uses the Attributes message returned by the accessed system to indicate the attributes the records being sent should have. For record retrieval with systems retaining attributes, records are transferred with the attributes of the Attributes messages returned by the accessed process.

After the initial set up messages are sent, the accessing system receives a response from the accessed process. If the access specified opens a file, an Attributes message and Extended Attributes messages followed by an Acknowledge message is sent from the accessed system containing the actual attributes of the accessed file. If the operation specified in the Access message deletes, renames or executes a file, no Attributes messages or Acknowledge messages are returned and the response is an Access Complete message.

To minimize the tying up of network resources (such as logical links and buffers), the Configuration, Attributes, Extended Attributes, Name, and Access messages should be sent in a timely manner. A timer may be set for each message and if it does not arrive in a reasonable time the link may be disconnected by the accessed process. After the Acknowledge message has been received, the file is open and the accessing process sets the pace for access of the file.

If there are errors in the setup procedure, a Status message will be returned.

NOTE

A receive must always be outstanding in order to accept both expected and unexpected DAP Status messages. Status messages are always sent as ordinary (not Interrupt) messages.

If an error is detected in a Status message, either during or after setup, a protocol error has occurred and there is nothing that can be done to recover so the link should be disconnected.

Errors in exchanging Configuration messages should be very rare since the information in Configuration messages will generally be "canned" and of an informative nature. If the accessed system detects an error in the Configuration message, it returns a Status message and the accessing system can either retry or disconnect. If the accessing system detects an error, it disconnects.

NOTE

If, however, a Configuration message appears to be in error because the SYSCAP field is too long and the DAP version number is greater than that to which the current software is written, assume that the SYSCAP field has been extended in the later version of DAP. Ignore this error. This is the only time when an error is ignored. The assumption is that the more sophisticated DAP process uses only a subset of the protocol and thus both sets of software can work together.

5.1.1 **Errors in the Setup Sequence** - The accessed process returns a Status message for errors detected in each message (Configuration, Attributes, Name, and Access). On receiving an error in response to one of these messages, there are three possibilities open to the accessing DAP process:

1. Disconnect the link.

2. Send the corrected message responsible for the error. There is no point in sending the original message unless there is sufficient doubt that the message was delivered properly or that the error indicated was of a temporary nature. For example, an attempt was made to open a file already open by another process.

3. Start a different access. A new access usually starts with an Attributes message, but it could start with an Access message (where the type of access does not require attributes such as ERASE) or even a Configuration message.

If the user process tries to recover by sending a corrected message or starting a new access, the accessed DAP process can accept any of the setup messages in response to a Status message. Table 6 contains a list of responses to setup message errors. In this table, Extended Attributes messages are considered as being Attributes messages and Name messages part of the Access message. When recovering from an error in an Attributes Extension message, the accessing process should back up to the Attributes message.

Table 6
Responses to Setup Message Errors

Error	Responses		
	Configuration Message	Attributes Message	Access Message
Configuration Message	1	Ø	Ø
Attributes Message	1	1	2
Access Message	1	1	1

where:

Ø Invalid response.
1 Valid response.
2 Valid response only for accesses requiring no attributes message.

Errors detected by the accessing process in Attributes, Extended Attributes, Name and Acknowledge messages cause the accessing process to disconnect the logical link thus terminating the access.

5.1.2 **Setup Sequence** - If a timer is used between setup messages, this same timer should be set by the accessed process after an error during setup. If the timer expires, a disconnect should be initiated.

The following conventions are used in DAP message sequence diagrams in this and subsequent sections:

- Brackets [ ] denote optional messages.
- Messages on the left side of the arrows are from the accessing process.
- Messages on the right side of the arrow are sent by the accessed process.

NOTE

The message sequence diagrams in this and subsequent sections are not definitive. However, although they do not cover every situation that can arise with DAP remote file access, they should be complete enough to give an idea of what should be done in those situations not explicitly addressed.

After a logical link is established, the setup sequence is as follows:

1. **Configuration information exchange:**

```
CONFIGURATION<------>CONFIGURATION
```

NOTE

Both accessing and accessed processes can send Configuration messages immediately on link establishment. The accessing process must send its Configuration message immediately.

2. **Setup for access:**

```
ATTRIBUTES------>

[EXTENDED
 ATTRIBUTES]---->

[NAME---------->
 (DEFAULT)]


ACCESS---------->

             <------ATTRIBUTES

                    [EXTENDED
             <------ATTRIBUTES]

             <------[NAME (EXPANDED FILESPEC)]

             <------ACKNOWLEDGE
```

NOTES

1. An Attributes message is not required when the Access message specifies ERASE, RENAME, DIRECTORY LIST, or EXECUTE command file. For these types of access, the accessed process returns neither the Attributes message nor the Acknowledge message unless specifically requested by the DISPLAY field. If the DISPLAY field is omitted from the Access message and the function is OPEN, CREATE, or SUBMIT, the accessed process returns a Main Attributes message. When a DISPLAY field is present in an Access message, the accessed process returns only the messages requested by the DISPLAY field. When RENAME is specified, follow the Access message by a Name message containing the new name for the file (see Section 5.2.8).

2. Request the optional Name message returned by the accessed process after opening the file by setting a bit in the DISPLAY field of the Access message. The Name message contains the file specification of the file opened or created after all the logical names have been resolved and defaults applied.

3. **Error in Configuration messages:**

(a) Disconnect after error detected:

```
CONFIGURATION------>(error detected)

             <------STATUS

DISCONNECT

               or
```

(b) Correcting erroneous message:

```
CONFIGURATION------>(error detected)

             <------STATUS

CONFIGURATION------>

               or
```

(c) If the error is in the returned message:

```
CONFIGURATION------>

(in error)   <------CONFIGURATION

DISCONNECT
```

4. **Error in Attributes message:**

(a) Disconnect after erroneous message:

```
ATTRIBUTES------>(error detected)

          <------STATUS

DISCONNECT
```

or

(b) Correcting erroneous message:

```
ATTRIBUTES------>(error detected)

          <------STATUS

ATTRIBUTES------>
```

or

(c) Starting a new remote file access on the same link. The new access shown must be one not requiring the Attributes message as no valid attributes are currently in effect (the former Attributes message contained an error). See Note 1 in Section 5.1.2.

```
ATTRIBUTES------>(error detected)

          <------STATUS

ACCESS    ------>
```

5. **Error in Optional Extended Attributes message:**

(a) Disconnect after erroneous message:

```
ATTRIBUTES------>

EXTENDED
ATTRIBUTES 1----->
     .       .
     .       .
     .       .

EXTENDED
ATTRIBUTES n----->(error detected)

          <------STATUS

DISCONNECT
```

or

(b) Correcting erroneous message (note that restart backs up to the Attributes message):

```
ATTRIBUTES------>

EXTENDED
ATTRIBUTES 1----->
     .       .
     .       .
     .       .

EXTENDED
ATTRIBUTES n----->(error detected)

           <------STATUS

ATTRIBUTES----->

EXTENDED
ATTRIBUTES 1----->
     .
     .
     .

EXTENDED
ATTRIBUTES n----->
     .
     .
     .
     etc.
```

(c) Starting new access on same link:

```
ATTRIBUTES-------->

EXTENDED
ATTRIBUTES 1------>
     .
     .
     .

EXTENDED
ATTRIBUTES n------>(error detected)

          <------STATUS

ACCESS    ------>
```

6. **Error in Name message for default file specification:**

(a) Disconnect after erroneous message:

```
NAME     ------> (error detected)

         <------STATUS

DISCONNECT
```

(b) Correcting erroneous message:

```
NAME       ------>(error detected)
           <------STATUS
NAME  ------>
```

(c) Starting new access on same link:

```
NAME            ------>(error detected)
                <------STATUS
ATTRIBUTES      ------>
[NAME           ------>]
ACCESS          ------>
```

7. **Error in Access message:**

(a) Disconnect after erroneous message:

```
ATTRIBUTES------>
ACCESS     ------>(error detected)
           <------STATUS
DISCONNECT------>
```

or

(b) Correcting erroneous message:

```
ATTRIBUTES------>
ACCESS     ------>(error detected)
           <------STATUS
ACCESS     ------>
```

or

(c) Starting new access on same link with new Attributes and Access messages:

```
ATTRIBUTES------>
ACCESS     ------>(error detected)
           <------STATUS
ATTRIBUTES------>
ACCESS     ------>
```

5.2 Transferring Data Over the Link

The message exchange sequence for transferring data over the link depends on the direction of data flow with respect to the accessing and accessed systems. Data may be sent to the accessing node as in a retrieve operation or from it as in a store operation.

Before data transfer can start, however, a data stream must be initiated by sending a Control message ($CONNECT) after the file is open. With file systems that support multiple data streams, additional data streams can be initiated with more Control messages. Multiple data streams are differentiated by using the STREAMID number. Data messages for a particular data stream must have the same STREAMID number as the Control message that initiated the data stream. If the STREAMID number is omitted, a default of Ø is used.

The sequence for initiating a data stream is as follows:

```
CONTROL(CONNECT)---->

                     <----ACKNOWLEDGE
```

If an error occurs, a Status message is returned instead of an ACK. A new data stream can be initiated any time the file is open by using the above message sequence.

NOTE

> Multiple data streams cannot be used if file transfer mode is specified in the RAC field of the Control message. The file transfer mode implies a single data stream with only data (no control messages) flowing over the link. By eliminating Control messages, efficiency is gained.

There are three classes of Status message which can be sent by the accessed process during data transfer:

1. **Successful.** The accessed process returns a success Status message for data transferred when not in file transfer mode. The successful Status messages synchronize DAP processes and return information to the user.

   In file transfer mode, data is pipelined, and successful status messages are omitted for efficiency.

2. **Warning.** This class of Status message is sent to the accessing process when an operation completes without complete success. For example, this message is sent if a record was inserted in an indexed file that had a duplicate key. Warning Status messages are always sent to the accessing process provided the Configuration message states the accessing process supports them. After sending a warning Status message, the accessed process suspends processing on that data stream until it receives a Continue Transfer message (or the next Control message for Record Retrieval) instructing it to resume processing or an Access Complete message terminating the access. On receiving a warning Status message, the accessing process either sends a Continue Transfer (Resume) to continue data transfer, a Control message, or an Access Complete if it wants to terminate the access.

3. **Error.** This class of Status message is sent when an operation was unable to complete at all, for example, if there were a read error. When an event in this class occurs, a Status message is always sent to the accessing process. After sending the Status message, the accessed process suspends processing on that data stream until it receives a Continue Transfer message telling it what to do or an Access Complete message terminating the access. On receiving an error Status message, the accessing process either sends an Access Complete to terminate the access or a Continue Transfer if it wants to attempt recovery.

These three classes of Status message are differentiated by the MACCODE field of the Status message (see Section 3.11 and Table 2). Status codes returned by the file system often assist in determining which class of Status message, if any, should be sent on the completion of each operation. However, ultimate responsibility for the classification and mapping of file system status codes into DAP Status messages resides with those who implement DAP server (accessed) processes. The following subsections contain examples of the handling of both warning and error Status messages.

5.2.1 **Sequential File Retrieval** - For sequential file retrieval, the accessed system sends data records. Once the initial startup sequence is completed and the data stream initiated, a single Control message (Get) is sent to start data records flowing. Thereafter, the file records are transmitted without waiting for any further DAP messages to control sending messages. The lower level protocols perform all flow control.

To specify sequential file retrieval, the accessing process specifies sequential file access (or virtual block number file transfer) in the Control (Get) message. The accessed process then sends file records (or blocks) without waiting for any further DAP Control messages. In contrast to sequential file retrieval, if sequential record access is specified, the accessing process must send a Control message for each record retrieved.

Data messages continue to arrive until one of the following occurs:

- The end-of-file is reached on the accessed system.
- An error occurs in accessing the file.
- The accessing system decides it has completed its access.

In the first case, the last record sent in a data message is followed by a Status message with end-of-file detected set. In the second case, a Status message is sent when an error occurs in accessing the original file.

If the accessing system receives a Status message with end-of-file, it sends an Access Complete message and waits for an Access Complete (Response). It then either disconnects or initiates another access by sending a setup sequence. If the accessing process receives a Status message with either an error or warning, it may either send an Access Complete Command and wait for an Access Complete (Response) or try to recover with a Continue Transfer.

If the accessing system decides to terminate access prior to end-of-file, it sends an Access Complete (Close) and waits for an Access Complete (Response) in return. In such cases, an accessing system issuing an Access Complete (Close) may still receive one or more records for the file, an end-of-file indication or even a Status message due to the pipelining delay in the system. It should pass over these records until an Access Complete (Response) is received. It may then disconnect or access another file.

A number of possible sequential file retrieval sequences are diagrammed below. In each case, the accessing process sends the messages to the left of the arrows. The access process sends the messages to the right of the arrows. Optional messages are bracketed.

1. **Retrieval until End-of-File (EOF):**

```
CONTROL (GET)------>
              <------RECORD 1
                  .
                  .
                  .
              <------RECORD n
```

NOTE

Transfer continues until End-of-File or error

```
              <------STATUS (End-of-File)

ACCOMP (CLOSE)------>
              <------ACCOMP (RESPONSE)
```

The accessing process may now issue another access or disconnect the link.

2. **Retrieval until error or warning status:**

```
                    <------RECORD n
                    <------STATUS
```

NOTE

The STATUS message occurs while trying to read record n+1.

When an error is received, the accessing process can do one of the following:

(a) Request link termination.

```
ACCOMP (CLOSE)-------->
                <------ACCOMP (RESPONSE)
```

(b) Request the information be sent again

```
CONTINUE (TRY AGAIN)------>
                      <------RECORD n+1
```

(c) Skip that record and continue

```
CONTINUE (SKIP)------>
                <------RECORD n+2
```

(d) When a warning is received, the accessing process can request link termination as in (a) above or resume processing by sending a Continue Transfer (Resume) message.

```
CONTINUE (RESUME)----->
                <------RECORD n+1
```

3. **Retrieval with access termination:**

```
                  <----RECORD m
ACCOMP (CLOSE)---->
                  <----ACCOMP (RESPONSE)
```

The accessed process may set a timer following sending Access Complete (Response). If neither a Disconnect or another message is received within the time interval, it may disconnect the link.

4. **Retrieval with checksum error on close:**

```
                  <----RECORD n
                  <----STATUS (EOF)

ACCOMP(CLOSE)---->
                  <----STATUS(checksum error)

ACCOMP(CLOSE)---->
                  <----ACCOMP(RESPONSE)
```

Do not send an Access Complete (Purge) to the accessed process as it purges the input file. Sending the Access Complete message with the CHECK field causes a comparison of the checksums. If the CHECK field is omitted, checking is by-passed. See Section 5.5.2.

5.2.2 **Sequential File Storage/Append** - In the store case, data is sent to the accessed system. Following the initialization of the data stream, the accessing system sends a Control (Put) message to tell the accessed process what to do. The Control message is followed by file records using the Data message. The accessed system accepts these messages and continues until the accessing system sends an Access Complete (Close). This causes a corresponding Access Complete Response to be returned following successful file closure, or a Status message to be sent if an error occurs in closing the file or a checksum failure is detected. For other than a checksum failure, the access is concluded and another access may start or the link may be disconnected. If a checksum failure is detected, another Access Complete (without the CHECK field) is sent, either close or purge, to terminate the access the way the user desires.

To specify sequential file storage, the accessing process specifies sequential file access in the Control message together with Put. To specify sequential file append, the operations are the same except "position to EOF" is specified in the Control message in addition to Put and sequential file access. As with sequential file retrieval, sequential file storage implies the use of only one data stream and no Control messages after the initial Control (Put) message.

Example 1 below shows an optional Access Complete (End-of-Stream) message flushing the pipeline before closing the file. This resynchronizes the accessing and accessed processes. Thus, error handling is easier if an error occurs in writing the final records to the file when the user issues a close command. After the error is returned to the user, he has the option of purging the file as the Access Complete (Close) is not yet in the pipe.

If an error occurs during record transfer, the accessed system returns a Status message. This must always be replied to with a Continue message sent as an Interrupt message (because of possible pipelining). In addition, to terminate the access, send an Access Complete message.

A list of sequential file storage sequences follows. In each sequence the accessing process sends the messages to the left of the arrows. The accessed process sends the messages to the right of the arrows.

1. **Store with no errors:**

```
CONTROL (PUT)          ------>
RECORD 1               ------>

                         .
                         .
                         .
                         .
```

NOTE

Transfer continues until access is complete or error

```
RECORD n               ------>
[ACCOMP (EOS)------>]
                       [<------ACCOMP (RESPONSE)]
ACCOMP (CLOSE)         ------>
                       <------ACCOMP (RESPONSE)
```

2. **Error during transfer:**

(a) Purge the new file and terminate:

```
RECORD n               ------>
                       <------STATUS
ACCOMP (PURGE)         ------>
CONTINUE (ABORT)       ------>(INTERRUPT)
```

NOTE

On receiving the Continue Transfer Interrupt message, the accessed system discards records until Access Complete (Purge) is received. It then purges the incomplete file and returns an Access Complete.

```
                       <------ACCOMP (RESPONSE)
```

or

(b) Close the new file and terminate:

```
ACCOMP (CLOSE)  ------>
CONTINUE (ABORT) ------>(INTERRUPT)
```

NOTE

The accessed system discards records until the Access Complete (Close) is received and then closes the incomplete output file.

```
                <------ACCOMP (RESPONSE)
```

or

(c) Retry--the accessed system still has the record which caused the error in its buffer:

```
CONTINUE (TRY AGAIN)------>(INTERRUPT)
RECORD n+1          ------>
```

or

(d) Skip the record and continue:

```
CONTINUE (SKIP)------>(INTERRUPT)
RECORD n+1     ------>
```

NOTE

On an error, the accessed process does not issue any more receives after sending the Status message and before receiving the Continue message, which tells it what to do. If the accessing process responds to the error by sending an interrupt Continue Transfer (Retry) message and the retry is successful, the accessed process posts a receive and carries on with the data transfer. If the retry fails, another Status message is sent. A Continue message with skip always posts a receive and tries to carry on having skipped the record which caused the original error. For file transfer store or append, continue messages must be sent in interrupt mode as there may be data in the pipeline.

3. **Warning during transfer:**

(a) Resume after receiving warning status:

```
RECORD n        ------>
                <------STATUS (WARNING)
CONTINUE(RESUME) ------> (INTERRUPT)
RECORD n+1      ------>
```

or

(b) To terminate the access after a warning, follow sequence 2(a) above to purge the output file or 2(b) to keep the output file.

NOTE

After sending the warning Status message, the accessed process issues no more receives until it receives an interrupt Continue Transfer message instructing it what to do.

4. **Stopping a sequential file storage operation before it is complete and purging the incomplete file on the accessed system:**

```
RECORD n---------------->

ACCOMP (PURGE)  -------> Purge the incomplete file

                <----ACCOMP (RESPONSE)
```

To save an incomplete file on the accessed system, the operations are as in Step 1.

5. **Checksum error on close:**

```
Record N---->
ACCOMP (CLOSE)---->

                <----STATUS (Checksum error)

ACCOMP(CLOSE/PURGE)---->

                <----ACCOMP(RESPONSE)
```

The user can either keep or purge the erroneous output file by sending either a close or purge Access Complete message without the CHECK field.

5.2.3 **Record Retrieval** - Record retrieval requires that a Control message (with a record key for random retrieval) be sent by the accessing process for each record accessed. To specify record retrieval, the accessing process sets sequential record access, keyed access or Record File Address (RFA) access in the Control message. Block mode transfer, similar to record retrieval, is specified by setting Virtual Block Number (VBN) access.

For keyed, VBN, or RFA access, the sequence is as follows:

```
CONTROL (get record with Key n)---->
                            <---- RECORD n, STATUS

CONTROL (get record with Key m)---->
                            <---- RECORD m, STATUS
```

For sequential record access, the state operation is as follows:

```
CONTROL (get sequential)--------->
                              <---- RECORD k, STATUS

CONTROL (get sequential)-------->
                              <---- RECORD k+1, STATUS
```

Once the location of a particular record in a file is found using random access, the user frequently wants to get subsequent records sequentially. To do this, switch the access mode from keyed or RFA to sequential in the Control message, and issue a Get. (With RMS systems, the user is free to switch access modes according to the RMS rules.)

```
CONTROL (get record with Key r)---->
                              <----RECORD r, STATUS

CONTROL (get sequential)      -------->
                              <----RECORD r+1, STATUS
```

Once a particular record in a file is found, it is possible to transfer the remainder of the file in sequential file access mode.

```
CONTROL (get record with key t)---->
                              <----RECORD t, STATUS

CONTROL (sequential file access, get)----->
                                   <----RECORD t+1, STATUS
                                   <----RECORD t+2, STATUS
                                     .
                                     .
                                     .
                                   to end-of-file
```

Error handling for sequential record retrieval is similar to error handling for sequential file retrieval. The handling of warnings is easier, however. In order to continue processing, the accessing process sends a Control (Get). A Contine Transfer (Resume) is not necessary, but should be ignored if received.

Error handling for random record retrieval is similar to that for sequential file retrieval. However, the Continue (Skip) recovery option, which is valid for sequential retrieval, is not valid for random retrieval. When a control request specifies a nonexistent record while doing random record retrieval, the accessed process will return an appropriate error message (for example, record number out of range or record not found).

5.2.4 **Record Store** - This is similar to sequential file store in messages exchanged. The access message specifies whether to open an existing file or create and open a new file. The Control message must specify Put access. For record storage, the accessing process may specify sequential record access, or keyed access. Optionally, VBN access may also be used.

For relative files, the data messages must include the relative record number field specifying the number of the record (RECNUM). For hashed files where the user is supplying his own hash code (RB$HSH set in the ROP field of a Control message), RECNUM contains the hash code. In all other cases, the contents of RECNUM are ignored and will probably be set null to minimize data transmission overhead. For sequential files, records are written starting at the current position within the file.

The sequence of records to be stored may be preceded by a Control (Put) message if it is necessary to change record options or access mode from the current value. Optionally, each record to be stored may be preceded by a Control (Put) message. This is inefficient, however, since it doubles the number of DAP messages transmitted. When storing a record, if the Data message is preceded by a Control message that contains a record number in the KEY field and the Data message also contains a record number in the RECNUM field, then the record number in the RECNUM field will be used.

The sequence for record storage with return of status is as follows:

```
RECORD n -------->

          <-------- STATUS

RECORD n+1 ------>

          <-------- STATUS
```

Section 5.2.2 describes error-handling. Note that a warning Status message requires an interrupt Continue Transfer (Resume) (or Abort) to restart processing because of possible pipelining problems. Continue (Skip) causes the accessed process to ignore the record which caused the error and go on to process the next DAP message.

5.2.5 **Append to Existing File** - The append operation is identical to sequential store and applies only to sequential files. The accessed system places the records at the logical end of the file. The Control message sets the position to EOF. The sequence is as follows:

```
RECORD 1-------->

         <-------- STATUS

RECORD 2-------->

         <-------- STATUS
```

5.2.6 **Deleting a File** - The delete operation (Erase) does not cause any file data to be transferred, but does manipulate file structures. Deleting a file does not require an Attributes message in the setup sequence.

The message sequence for the delete operation is as follows:

```
[ATTRIBUTES---------->]

ACCESS (ERASE)  ----->

                <-----ACCOMP (RESPONSE)
                     or
                <-----STATUS
```

5.2.7 **Command/Batch Execution Files** - The Data Access Protocol includes commands for the transfer and submission of files to a batch processing facility or command interpreter. The "submit-as-command-file" request in the Access message requests the storage of the data that follows in a temporary file. This request also indicates submission of the file to a batch-type facility upon access completion (closing of the file). The batch facility deletes the file following execution. DAP does not respond to any feedback from the batch facility. DAP does not guarantee that the file actually executes in the batch monitor. DAP transfers the file using sequential file storage (Section 5.2.2).

The "execute-as-command-file" requests the submission of the specified file to the batch facility only. No data follows this command. The specified file is previously established on the accessed system. The file is not deleted following execution by the batch facility, so that the sequence "store," and "execute-command-file" will transfer a file, submit it and retain the file for later use. The sequence for "submit-as-command-file" is identical to "store," while the "execute-command-file" is identical to Erase.

NOTE

Since errors are not returned to the originating node automatically, a test for errors might be included in indirect command files. Upon error or completion, a suitable message can be returned to the originating node.

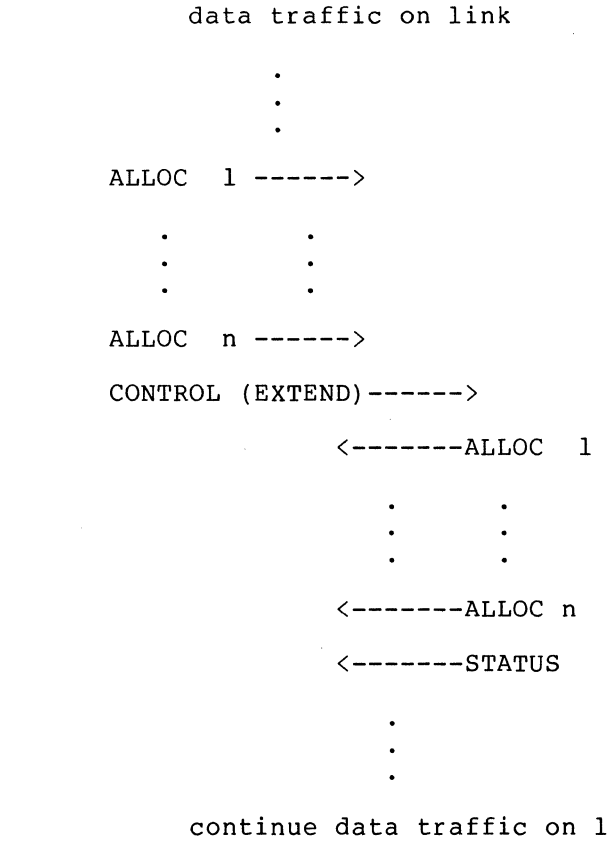
5.2.8 **Renaming a File** - The rename operation does not cause the transfer of any file data. However, it does require the transmission of two file specifications. The old name of the file is in the Access message; the new name for the file is in the Name message following the Access message. Rename does not require an Attributes message in the set up sequence.

The message sequence for the rename operation is as follows:

```
[ATTRIBUTES]--------->

ACCESS (RENAME)------>

NAME (FILESPEC)------>

          <----------ACCOMP (RESPONSE)
                          or
          <----------STATUS
```

5.2.9 **Extending Files** - File systems often offer an automatic file extension facility to extend an existing file when space runs out but there is more data to be written to the file. However, automatic file extension usually offers little control over the size and placement of file extensions.

Another method of extending a file is to use the explicit $EXTEND code of the Control message where a file system supports user directed extension. User directed file extension is initiated after the file is open. The DAP state operations are:



Precede the Control (Extend) message by one or more ALLOC (Allocation Attributes Extension) messages specifying the type of extension desired. Return a corresponding number of ALLOC messages, specifying the extensions performed. Return a Status message terminating the string of returned ALLOC messages. If an error occurs, a Status message will contain the error code.

5.2.10 **Display Attributes** - This command provides a means of obtaining attribute information about a file. It is specifically for use with RMS file systems. If this command is used, the accessing node must specify which groups of attributes it wants. It does this by setting the appropriate bits in the DISPLAY field of the Control message sent to the remote node. In the case of the allocation and key definition groups of attributes, the appropriate Key Definition and Allocation Attributes Extension messages precede the Control message to indicate for which Keys of reference or which areas attributes are required.

The state operations for display are:

```
[ATTRIBUTES
EXTENSION------------->
MESSAGES]

CONTROL (Display)------>

          <---------[ATTRIBUTES and ATTRIBUTES EXTENSION
                     MESSAGES AS REQUIRED]

          <---------STATUS
```

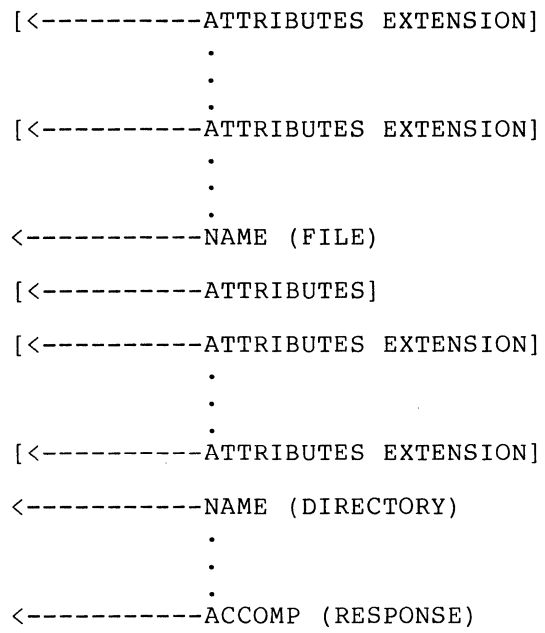
5.2.11 **Directory List** - A directory or multiple directory listing request causes the accessed process to return the file attributes of the files specified in the FILESPEC field of the Access message requesting the directory list. The accessed process returns attributes using the Attributes and Attributes Extension messages. The accessing process can specify which Attributes messages it wants by setting the appropriate bit(s) in the DISPLAY field of the Access message. If the DISPLAY field requests the Name message (bit 8), the resultant file name is returned with the Attributes messages in a Name message. This Name message is in addition to the Name messages shown in the state operations below.

The file specification in the Access message may contain such "wild cards" as are recognized by the accessed process.

As can be seen from the state operations for directory list below, a Name message for the file the Attributes messages describe precedes each group of Attributes messages. If no bits in the DISPLAY field of the requesting Access message are set, only the Name messages are sent (no Attributes messages). A Name message precedes all the files in a given directory. Optionally, if all the directories for which a listing is requested do not reside on the same volume-set (or structure), a Name message precedes the directories for each volume-set. The Name message indicates which volume-set the following directories are on.

The state operations for directory listing are:

```
          ACCESS(DIRECTORY)------------>

                          [<----------NAME (VOLUME-SET)]

                           <----------NAME (DIRECTORY)

                           <----------NAME (FILE)

                          [<----------ATTRIBUTES]

                          [<----------ATTRIBUTES EXTENSION]
                                      .
                                      .
                                      .
                          [<----------ATTRIBUTES EXTENSION]

                          <-----------NAME (FILE)

                          [<----------ATTRIBUTES]
```



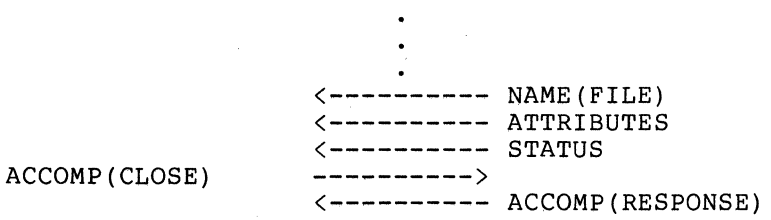
The accessing process may terminate a directory listing at anytime by sending an Access Complete(Close). This causes the accessed process to send no more Name or Attributes messages, to finish its operation (for example, close any files it may have open) and finally to return an Access Complete(Response). Note that due to pipelining, the accessing process may receive several Attributes and/or Name messages before receiving an Access Complete(Response) after it sends an Access Complete(Close).

Errors occurring during a directory listing result from trying to obtain information to generate either a Name message or an Attributes or Attributes Extension message when reading a directory or the attributes of a file. On a gross level, errors will be either read errors, access protection violations, or non-existent attributes (for some reason, the attributes for a file cannot be found or do not exist). Read errors are handled by using the Continue Transfer (Try again or Skip) or Access Complete (Close). Protection violations and non-existent attributes errors are handled using the Continue Transfer (Skip) or Access Complete (Close). Continue Transfer (Try again) usually just returns the same error.

Access Complete (Close) terminates the directory listing, closes any open files and leaves the link in a state where another DAP access may be started. Continue Transfer (Try again) repeats the operation that failed. Continue Transfer (Skip) causes the accessed process to skip over the operation causing the error, and attempt to recover the listing that lost some information. For example, this operation tries to read subsequent blocks in the directory, trying to get the next file name. Or the operation may attempt to read the next block containing attributes for the current file. In some cases, it may not be possible to recover using Continue Transfer (Skip).

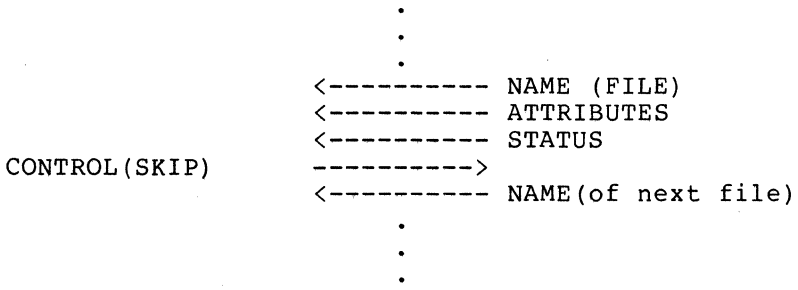
The following are examples showing error handling for directory listing:

1. Using Access Complete(Close) to terminate listing:

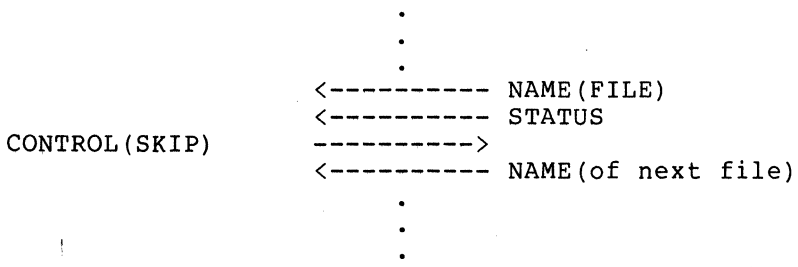


2. Using Control(Skip) to skip over error (some information will always be lost and in some cases it may not be possible to recover):

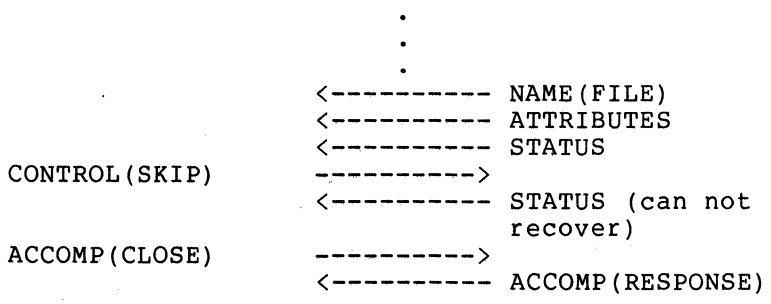
   (a) Error on reading file name:



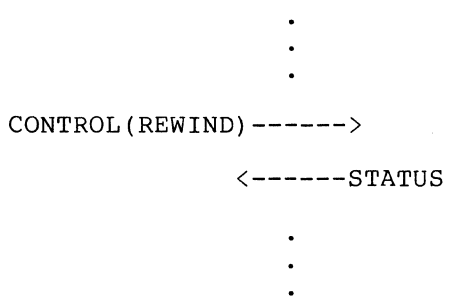
   (b) Protection violation on reading file attributes



   (c) Unable to recover after error:

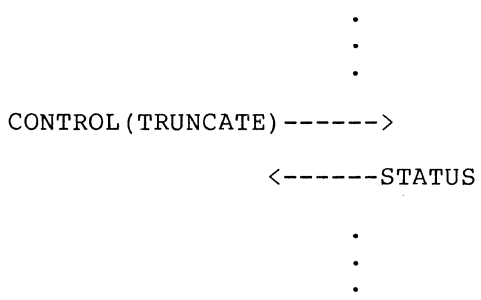


5.2.12 **Rewind Data Stream** - $REWIND sets the current context of the stream identified by the STREAMID field of the Control message to beginning of file (BOF). The state operations for rewind are:

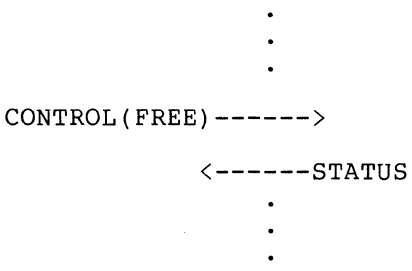


5.2.13 **Truncate File** - $TRUNCATE truncates sequential files and is not a valid operation on other file types.

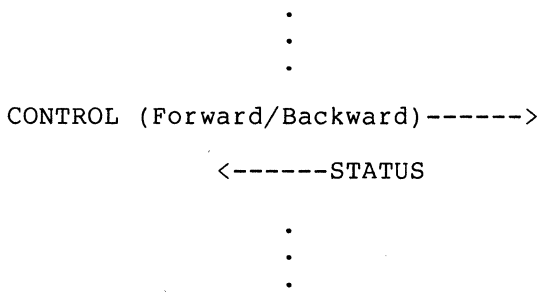
It causes the deletion of all records from the current record on, and the declaration of an EOF in place of the current record.



5.2.14 **Free Buckets** - $FREE unlocks all locked records in the stream identified by the STREAMID field of the Control message.

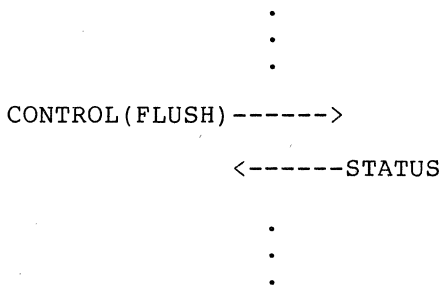


5.2.15 **Space Forward or Backward** - $SPACE forward or backward spaces the file the number of blocks specified in the KEY field.

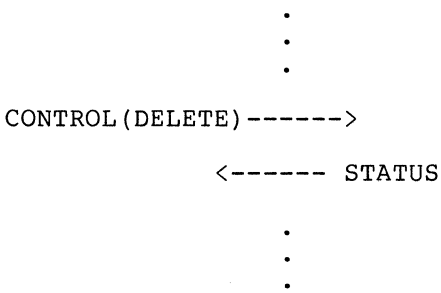


The RECNUM field of the Status message contains the number of blocks actually spaced. This may not be the same as the number of blocks specified in the Control message. For example, if the current position is VBN 7, a backspace of 10 will not actually backspace beyond the beginning of file.

5.2.16 **Flush I/O Buffers** - $FLUSH writes out all modified I/O buffers associated with the record access stream identified by the STREAMID of the Control message.



5.2.17 **Deleting a Record** - $DELETE deletes the current record for these files where record deletion is possible. For example, relative or indexed files usually support record deletion.



5.2.18 **Find** - $FIND operation is essentially identical to $GET except that it does not transfer any data. The specified record becomes the current record.

```
                    .
                    .
                    .

     CONTROL(FIND)------>

                 <------ STATUS

                    .
                    .
                    .
```

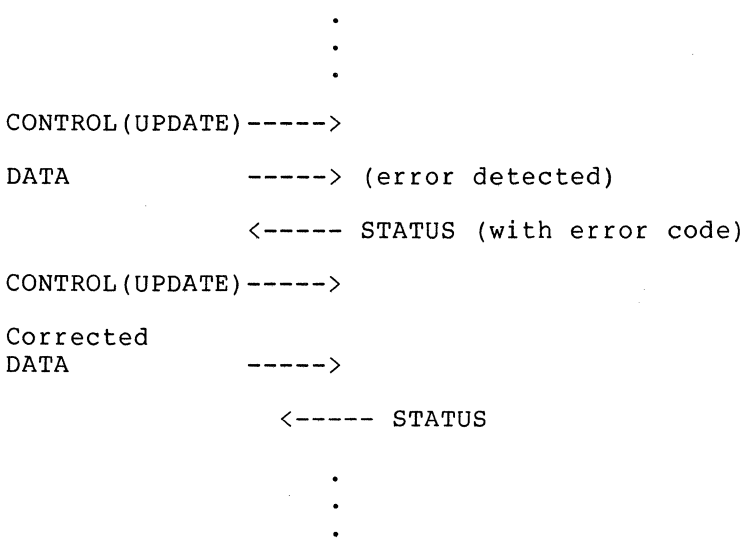
5.2.19 **Update** - $UPDATE causes the current record to be updated with the record in the following Data message on this stream. The Control(Update) and Data messages together form a transaction and must not have any intervening messages on that stream. For convenience and efficiency, they can be blocked together. The operation is:

```
                    .
                    .
                    .

  CONTROL(UPDATE)----->

  DATA           ----->

                 <----- STATUS

                    .
                    .
                    .
```

Because of the transaction nature of update, if an error occurs at any point in the operation, the whole transaction fails. If the accessed process detects an error in the Control message, it sends the appropriate Status message and it also discards the next Data message on that stream as it was a part of the transaction that failed. The accessing process recovers by starting the transaction over (having corrected the error) with new Control and Data messages as below:

```
                    .
                    .
                    .

  CONTROL(UPDATE)-----> (error detected)

  DATA           -----> (discard)

                 <----- STATUS (with error code)

  Corrected
  CONTROL(UPDATE)----->

  DATA           ----->

                   <----- STATUS (successful)

                    .
                    .
                    .
```

If an error is detected in the Data message, the accessed process returns the appropriate Status message and considers the whole transaction to have failed (current position may be lost depending on the nature of the error -- it may be necessary to re-establish the current position in the file). After correcting the error in the Data message, the accessing process must repeat the entire transaction:

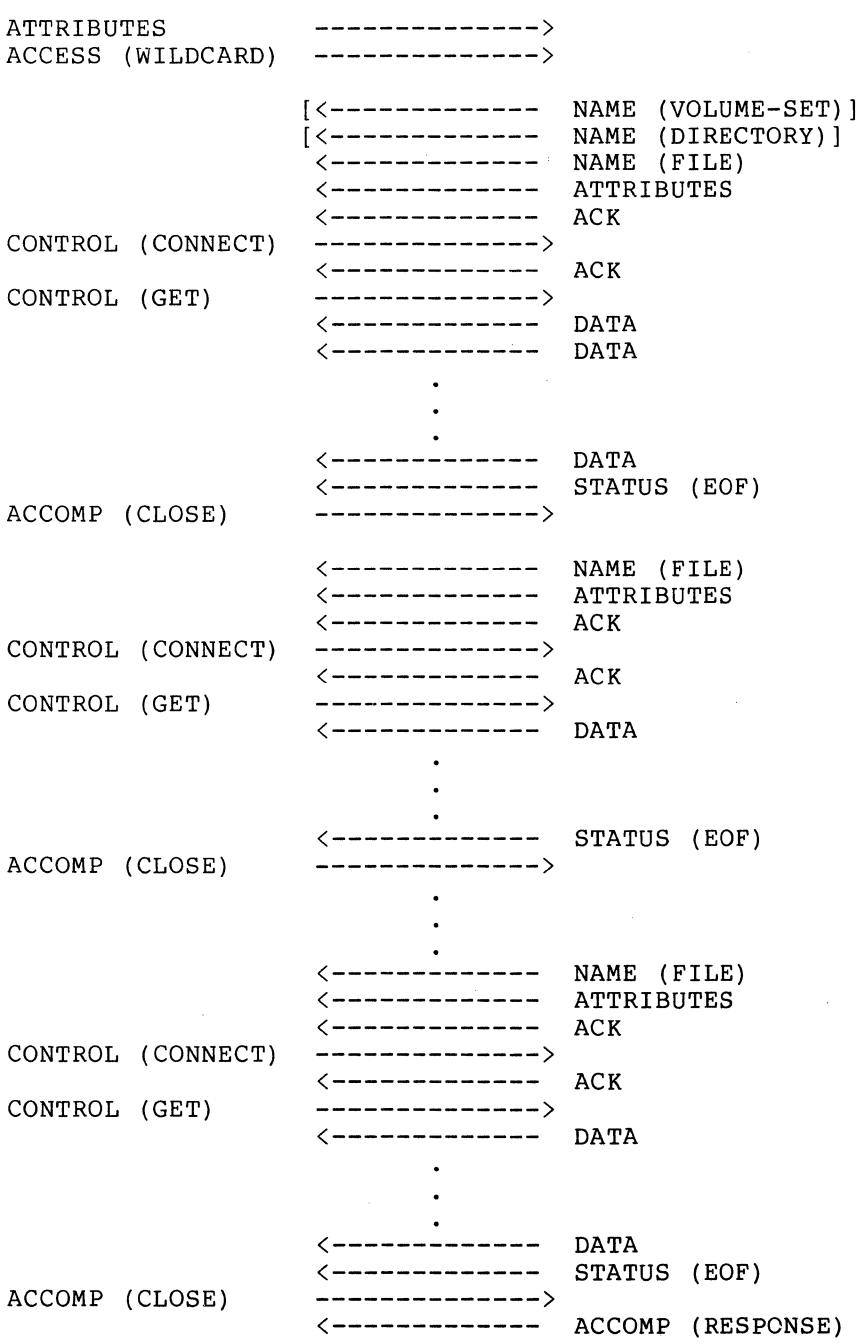


5.2.20 **Wildcard Operations** - The wildcard operation for sequential file retrieval, file deletion, file renaming and command file execution is supported through the DAP message sequences defined in the following subsections. The wildcard file specification contained in the Access message is in the format used on the accessed system.

NOTE

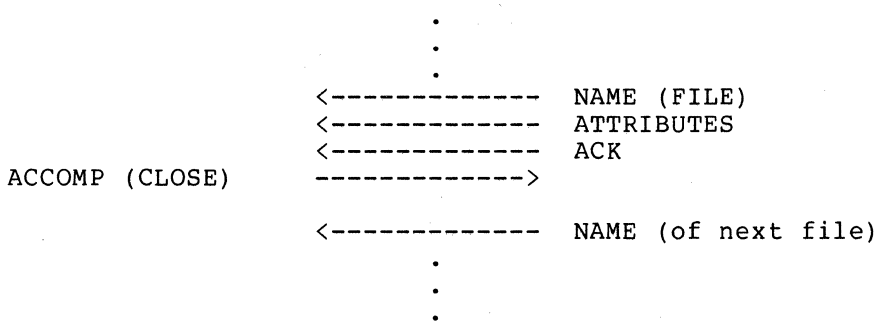
It is not necessary to specify wildcard support in the Configuration message in order to use wildcards with the directory list function. Directory list support implies wildcard support for directory list file specifications. However, wildcard support must be specified in order to use wildcards with any function other than directory list.

5.2.20.1 **Wildcard Sequential File Retrieval** - Wildcard sequential file retrieval operation is similar to sequential file retrieval as described in Section 5.2.1 except that Name messages and file attributes precede each transferred file. The state operations for wildcard sequential file retrieval are (optional Attributes Extension messages are not shown):

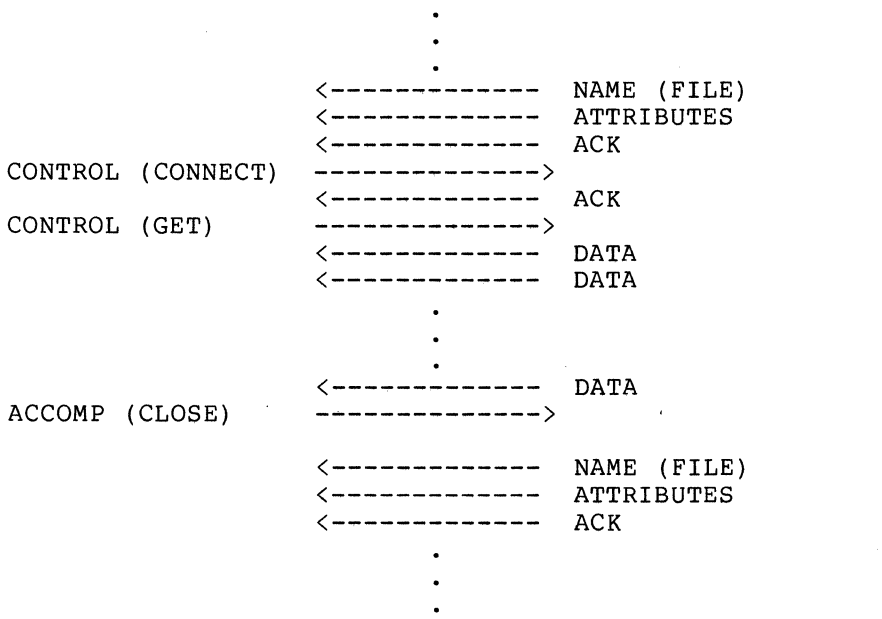


A new Name message for the volume-set or directory is sent only when either the volume-set or directory change.

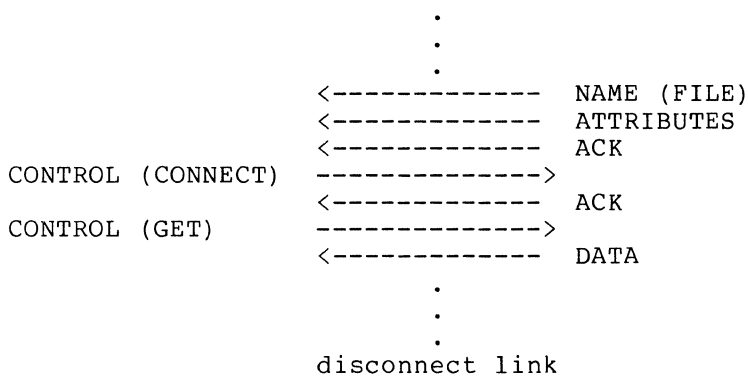
If the accessing process does not want one of the files specified by the wildcard file specification, it closes the file instead of establishing a data stream thus skipping the file.



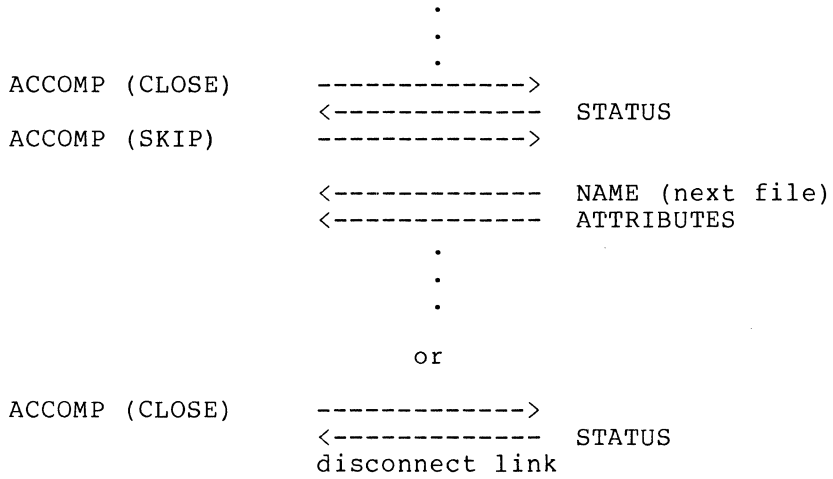
If, during a file transfer, the accessing process decides it does not require the remainder of the file but wants to go on to the next file, the accessing process terminates the access to the current file with an Access Complete(Close). The accessed process closes the current file and initiates the transfer of the next file in the wildcard series. Note that due to pipelining, the accessing process may receive several Data messages or a Status message before the Name message.



The accessing process may abort a wildcard file retrieval at anytime by disconnecting the link. When the accessed process detects link termination, it closes any currently open files.



Errors for wildcard sequential file retrieval are handled as with normal file retrieval except for errors in closing the file (excluding checksum errors). In this case Access Complete (Skip) forces file closure and clears the link so the accessed process can proceed to the next file to be transferred.

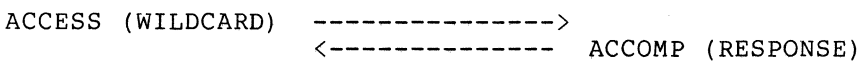


5.2.20.2 **Wildcard File Deletion** - Wildcard file deletion has two modes of operation:

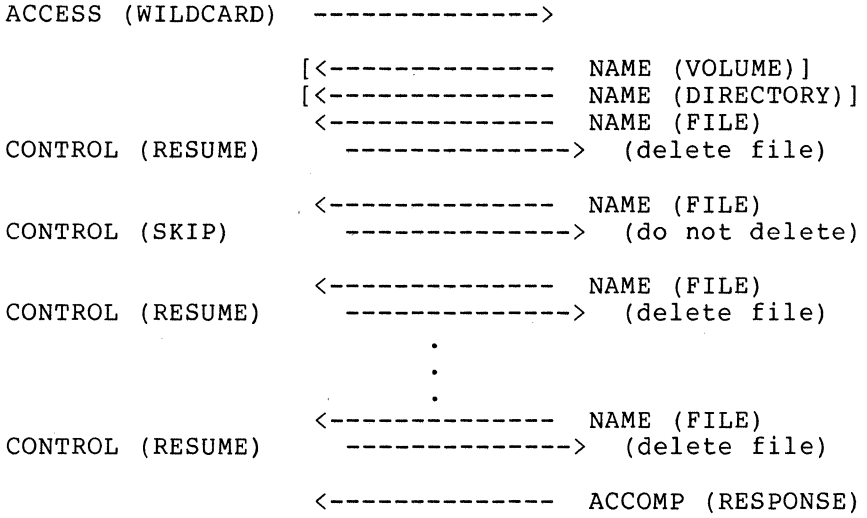
1. **Normal.** Delete all files specified by the wildcard specification before returning any response (except error) to the accessing process.

2. **Go/No-Go.** Return the name of each file meeting the wildcard file specification to the accessing process before deleting the file. The accessing process can then cause the file to be deleted by sending a Control (Resume) message or the file to be left in the file system by sending a Control (Skip) message. Setting bit 4 in the ACCOPT field of the Access message specifies the Go/No-Go mode of operation.

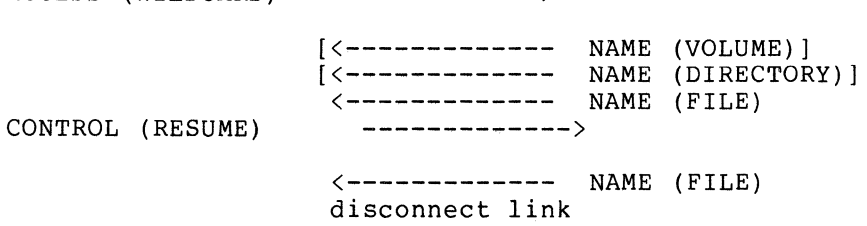
The operation of normal wildcard file deletion is:



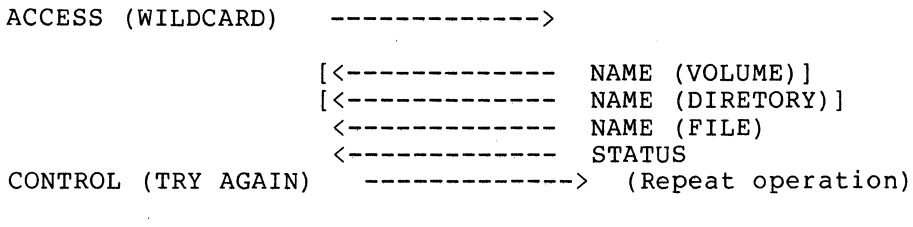
Operation of Go/No-Go wildcard file deletion is:



Abort Go/No-Go wildcard file deletion by disconnecting the link.



Use the Continue Transfer message to handle errors in wildcard file deletion. Either retry the deletion that caused the error or skip the file causing the error. Alternatively, abort the operations by disconnecting the link. The operation of error handling (with Go/No-Go) is:



```
                    <-------------  NAME (FILE)
                    <-------------  STATUS
CONTROL (SKIP)        ------------->  (Skip file)

                    <-------------  NAME (of next file)
CONTROL (RESUME)      ------------->  (delete next file)

                         or

                    <-------------  NAME (FILE)
                    <-------------  STATUS
                    disconnect link
```

The operation of error handling without the Go/No-Go option is identical to that with Go/No-Go except that once the error has been dealt with, the Control (Resume) is not required as in the above sequence. The operation without Go/No-Go is as follows:

```
ACCESS (WILDCARD)   ------------->

                   [<-------------  NAME (VOLUME)]
                   [<-------------  NAME (DIRECTORY)]
                    <-------------  NAME (FILE)
                    <-------------  STATUS
CONTROL (TRY AGAIN)   ------------->  (Repeat operation)

                         or

                    <-------------  NAME (FILE)
                    <-------------  STATUS
CONTROL (SKIP)        ------------->  (Skip file)

                         or

                    <-------------  NAME (FILE)
                    <-------------  STATUS
                    disconnect link
```

NOTE

Go/No-Go operation can also be used with single file delete. Setting bit 4 of ACCOPT requests this option.

5.2.20.3 **Wildcard File Rename** - Wildcard file rename operation is identical to wildcard file deletion except that a second file specification must be supplied as for normal rename (see section 5.2.8). This affects only the set-up as shown below (using Go/No-Go operation):

```
ACCESS (WILDCARD)--------------->
NAME (WILDCARD)  --------------->

                [<--------------  NAME (OLD VOLUME)]
                [<--------------  NAME (OLD DIRECTORY)]
                 <--------------  NAME (OLD FILE)
CONTROL (RESUME)   ------------->  (rename file)
                         .
                         .
                         .
                 <--------------  ACCOMP (RESPONSE)
```

In other respects, wildcard rename operation is identical to wildcard delete.

5.2.20.4 **Wildcard Command File Execution** - Wildcard command file execution operation is identical to wildcard file deletion operation.

5.3 Closing a File and Terminating Data Streams

The Access Complete End of Stream (EOS) command terminates a data stream. When the accessing process wishes to terminate a data stream, it may do so by sending an Access Complete (EOS) containing the STREAMID it wishes to terminate. This will not close the file even if it terminates the last active data stream.

An Access Complete (Close) closes the file and terminates the access. This includes closing out all remaining active data streams.

5.4 Terminating a Logical Link

Terminate the logical link by issuing a Disconnect Request. During the setup of the link, this may be done by the accessed process if optional timers indicate delay by the accessing process in supplying the required information. Once setup is complete, the accessing process controls the rate of access of the file. Disconnection at this point usually follows access completion. The accessing process may disconnect at any time; however, different systems may handle file closing and disposition differently if disconnection occurs during transfers.

The accessing process is not required to disconnect and reconnect following each access. However, if a new access is to be started, it must be initiated in a timely manner. If a timer is being used between setup messages, it should also be set by the accessed process following an Access Complete message.

5.5 File Security, Integrity and Protection

5.5.1 **Access Control** - DAP attempts to provide approximately the same degree of file security and protection over the network as is available locally. To do this, a DAP user must be a registered user of each system holding files he wishes to access. Embedded in the connect message sent by the accessing process is sufficient information for the user to be logged onto the system that has files he wishes to access. User access is first verified (not necessarily actually logged-on) and then file access is allowed to proceed under the normal rules for file access applicable to a local user.

If the accessing process wants to change the account under which the accessed process is running at the remote node, it must disconnect the logical link and reconnect specifying the new account in the connect.

5.5.2 **Data Integrity** - A checksum on all data transferred (in both directions) during a file access can be generated by both the transmitting and receiving node to ensure data integrity. If this optional checksum is required, a bit in the ACCOPT field of the Access message is set. The accessing and accessed DAP processes compute the checksum on the data in the DAP Data message whenever a Data message is sent or received. When the access is complete and the remote file is being closed, the checksum generated by the accessing process is sent to the accessed process in the CHECK field of the Access Complete (Close) message. The accessed process compares the checksum it received with the checksum it generated and closes the file if they agree. If they do not agree, a Status message is returned to the accessing process instead of an Access Complete (Response). If a checksum error occurs, the accessing process can either purge the output file or, if errors can be tolerated, keep and close the output file (Sections 5.2.1 and 5.2.2) by sending the appropriate Access Complete message without the CHECK field. Checksum errors should be logged by the accessing process (and optionally by the accessed process).

The checksum is a CRC computed only on the data in DAP Data messages. Data message headers are not included. The CRC polynomial is:

$$X^{16} + X^{15} + X^{13} + X^{7} + X^{4} + X^{2} + X + 1$$

Before starting data transfer, initialize the 16-bit CRC to -1. In other words, set all bits.

5.6 DAP-Based Applications Written by DIGITAL

DAP message types 128 through 191 are reserved for DEC-written applications based on DAP that require further application-specific messages in the protocol. FTS is an example of such an application. It requires the USERID message defined in Appendix A to supply accounting and access control information to the remote FTS server process.

5.6.1 **Access Control and Accounting for DAP-Based Applications** - The User Identification message is designed for use by applications using cooperating, DAP-speaking control processes to offer a file transfer based service to network users. One of the control processes is a queueing process that handles the user's requests for file transfer, and the other is a server process that handles the remote end of executing the users' transfers. These control processes characteristically establish a link and sequentially transfer several users' files over the same link. The server control process runs as a privileged process. It has access to all files within the system.

This scheme places the responsibility for access control (a user should be allowed to access only those files he has explicit permission to access) and accounting (charge each user for the resources he uses) on the cooperating control processes. The control processes are charged for the resources they use as they are logged on the systems they are running on. They need to pass the charges on to the users of the services. In order to perform access control and accounting functions, the server process needs the user's identification and optional account number. The User Identification message (see Appendix A) supplies this information.

Note that security is not an issue here. The server process "trusts" the queueuing process and assumes that the contents of the User Identification messages are correct and any necessary validation has been done by the queueing process. Therefore a password is not necessary or even desirable in this message. The server process can "trust" the queueing process as long as it is talking to a valid queueing process. This is ensured by the Session Control access control procedure which validates a password supplied by the queueing process before it allows a logical link to be established.

APPENDIX A

USER IDENTIFICATION MESSAGE

The User Identification message is an application message designed for use by DIGITAL's File Transfer System (FTS) and other DAP-based applications using the same queueing model. This queueing model consists of two controlling processes which speak DAP across the network. One of the processes accepts and queues requests from users to transfer files. The other is a server process which executes the user's requests at the remote node in response to DAP messages from the first process. A single logical link may be used to sequentially process any number of user requests. It is not necessary to disconnect the link after processing each user's file transfer request.

In order that appropriate access control and accounting measures can be taken by the server process, the queueing control process sends a User Identification message each time it initiates a file access for a new user. The User Identification contains the identity (for example, PPN) under which the access at the remote node takes place plus the optional user account number. The User Identification message is sent immediately before the Attributes message when initiating a new file access. The form of the message is:

USERID	IDMENU	IDENT	ACCOUNT	OPTIONS

where:

USERID : The operator field with TYPE = 128.

IDMENU(EX-6) : BM The following bit map specifies which of the following fields are present in this message. If the field is not present, the default, if any, should be used. These fields and only these fields may appear in the message and they must appear in the order specified.

Bit	Meaning (When Set)
Ø	IDENT
1	ACCOUNT
2	OPTIONS

IDENT(I-4Ø) : A The user identification (for example, UIC) under which the server process will execute the following remote file accesses. This establishes the user's identity for both resource usage accounting at the remote node and file access rights. If this field is defaulted, the previous identity in effect remains in effect. When a link is first established, the user identity from the Session Control connect message is the user identity in effect.

ACCOUNT(I-40) : A — For systems requiring additional information for accounting (for example, discrete project number) this field contains this accounting information.

OPTIONS(I-132) : A — Options field for additional information which may be required by the server control process. It may be used for supplemental accounting information, page headers, or, non-network routing information for mail.

APPENDIX B

REVISION HISTORY

B.1 Version 1.Ø to Version 4.1

A number of significant changes have been made to the Data Access Protocol since its first release. The major differences between DAP Version 4.1 and Version 1.Ø are:

- DAP Version 1.Ø could not adequately support indexed and ISAM file access.
- The format of the operator field has been expanded.
- The User Identification message has been eliminated.
- The Status and Error messages have been combined.
- The Access Complete message has been added.
- The Configuration message has been added.
- The two types of Data messages employed in Version 1.Ø have been merged into one Data message in Version 4.1.

While a definite incompatibility exists between Versions 1.Ø and 4.1, numerous steps have been taken to build a more flexible architecture. DAP Version 4.1 is flexible enough to allow new file access functions to be added to the protocol framework.

B.2 Version 4.1 to Version 5.6

A number of significant changes have been made to DAP since the Version 4.1 release. The major differences are as follows:

- The Key Definition, Allocation, Summary, Date and Time, Protection, and Name messages have been added. Previously some of these were present but were marked reserved.
- The format of the operator field has been expanded again to allow blocking of DAP messages greater than 256 bytes long and to allow an optional field in the Data message.
- A file transfer checksum on the complete file has been added for better file integrity.
- Indexed file access is now fully supported. Only record number random access was supported in 4.1.
- The Configuration message has been expanded to include more capabilities.

- Print file carriage control has been added.
- The Image data type definition has been clarified, especially for files containing non 8-bit bytes.
- The MACY11 data type has been added.
- Explanation of how to use the Extended Attributes messages has been added.
- Rename has been added.
- Display and Directory List have been added.
- Several new Control message options have been added.
- Wildcard support has been added.

The 5.6 version of DAP is upwardly compatible with the 4.1 version. The new facilities in 5.6 have been put into the protocol using extensibility built into DAP Version 4.1.

GLOSSARY

bucket

A grouping of a file's virtual blocks used for I/O transfer or structural format storage.

ISAM

Indexed Sequential Access Method. This access method is a combination of random and sequential access. Random access is used to locate a sequence of records and then access is switched to sequential to read the remaining records in the series.

JFN

Job File Number. The JFN is the job's global handle on a file.

key

A data item used to locate a record in a random access file system.

key field

For direct and indexed files, the position of the key within the record.

key of reference

The particular key field of the record for which the key applies.

MACY11

A format for fitting 16-bit words into 36-bit words for file transfers between PDP-11's and DECsystem 10's and 20's.

octets

Octets in this document are bytes of 8 bits, with bit 0 the rightmost (low-order, least-significant) bit and bit 7 the leftmost (high-order, most-significant) bit. Fields and bytes of other lengths are numbered similarly.

object type

Numeric value that may be used for process addressing by DECnet processes instead of a process name. See the Session Control specification for further details. DAP server processes are object type number 21 (octal).

RFA

Record File Address. The unique address of a record within a file. This method of addressing can be used explicitly with RMS.

RMS

Record Management Services. This file system will be used on all major DIGITAL systems except where space is limited (for example, RT-11). In addition to access modes provided by previous file systems, RMS provides random access for direct and indexed files and ISAM.

URD

Unit Record Device.

VBN

Virtual Block Number. This number is in the range 1 to n where n is the highest numbered block allocated to the file.

wild card

An asterisk (*) that replaces an element in a file specification. The asterisk specifies all known items in the range indicated by its position. For example, FILE.*;* specifies all known versions and types of all files named FILE.

DECnet DIGITAL Network
Architecture Data Access
Protocol Functional
Specification (DAP)
AA-K177A-TK

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Did you find errors in this manual? If so, specify the error and the page number.

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)_____

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code_____
or
Country

Please cut along this line.

—Do Not Tear - Fold Here and Tape —

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE DOCUMENTATION
146 MAIN STREET ML 5-5/E39
MAYNARD, MASSACHUSETTS 01754

— Do Not Tear - Fold Here and Tape —

Cut Along Dotted Line