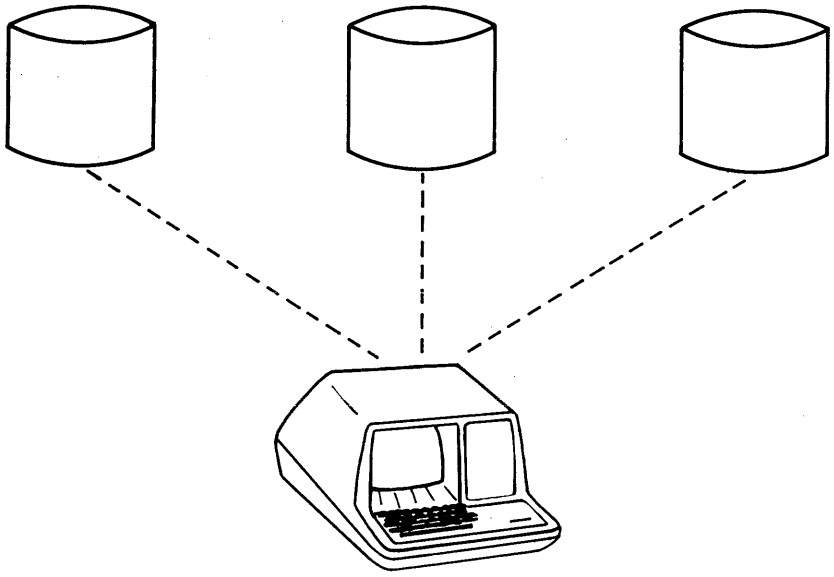
Order No. AA-D601A-TC

DAP



To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754.

digital

DECNET

DIGITAL NETWORK ARCHITECTURE

Data Access Protocol (DAP)

Functional Specification

Version 4.1

March 1978

digital equipment corporation · maynard. massachusetts

First Printing, March 1978

ABSTRACT

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 Network Services Protocol and its associated interface.

Table of Contents

List of Illustrations

List of Tables

1.0 SCOPE

This document describes the functions, characteristics, capabilities, and operation of the Data Access Protocol (DAP). It is primarily intended to assist developers and implementers in understanding how DAP functions within a system. The document is not intended to address specific implementations.

The Data Access Protocol Version 4.1 is specifically designed for remote file access via a file system such as Record Management Services. 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-independent manner, the device control features are not supported by DAP.

2.0 FUNCTIONAL DESCRIPTION

The Data Access Protocol is a user 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:

1. Retrieve a file from an input device (e.g., a disk file, card reader, or terminal);
2. Store a file on an output device (e.g., a disk file, line printer or terminal);
3. Provide ASCII file transportability between nodes;
4. Provide Error Recovery;
5. Allow multiple data streams to be sent over a logical link;
6. Command file execution and submission;
7. Provide for random access of records in a file;
8. Provide for file deletion;
9. Rename files; and
10. List directories.

2.2 Relationship to DECnet

DECnet is a family of products that create distributed networks from DIGITAL computers and their interconnecting data links. DECnet creates a general mechanism for sharing resources and providing interprocess communications within a distributed data processing environment. DECnet implementations adhere to a common network architecture that defines the structure and protocols used to communicate through the network.

The DIGITAL Network Architecture provides a modular design for DECnet. Its functional components are defined within three distinct layers: the Physical Link Control Layer, the Network Service Layer, and the Application Layer. Each layer performs a well-defined set of network functions (via network protocols) and presents a level of abstraction and capability to the layer above it:

Physical Link Control Layer - The Physical Link Control Layer controls the physical link operation to ensure both data integrity and sequentiality.

Network Service Layer - The Network Service Layer provides the mechanism that permits node-to-node communications and process-to-process communications between processes in the same or different nodes. It performs the network routing, message handling and information flow control functions.

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 Network Services Layer. DAP resides within the Application Layer.

2.3 Generic Model

As an aid toward understanding the Data Access Protocol, a generic model is presented. This model consists of a summary of the DAP messages and a typical DAP message exchange sequence (illustrating how DECnet Sequential File Retrieval is accomplished between two dialogue processes).

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

Table 2-1. DAP Messages

Message	Function
Configuration Message	Used to exchange system capability and configuration information between DAP speaking processes. This message is sent immediately following link establishment. It contains information about the operating system, the file system, protocol version, and buffering ability.
Attributes Message	Provides information on how data is being structured in the file being accessed. The message contains information on file organization, data type, format, record attributes, record length, size, device characteristics, and security.
Access Message	Specifies the file name and type of access requested.
Control Message	Used to send control information to a file system and to establish data streams.
Continue-Transfer Message	Allows recovery from errors. Used for retry, skip, and abort.
Acknowledge Message	Used to acknowledge access commands and control connect messages used to establish data streams.
Access Complete Message	Denotes Termination of Access.
Data Message	Transfers the file data over the link.
Status Message	Used to return status and information on error conditions.

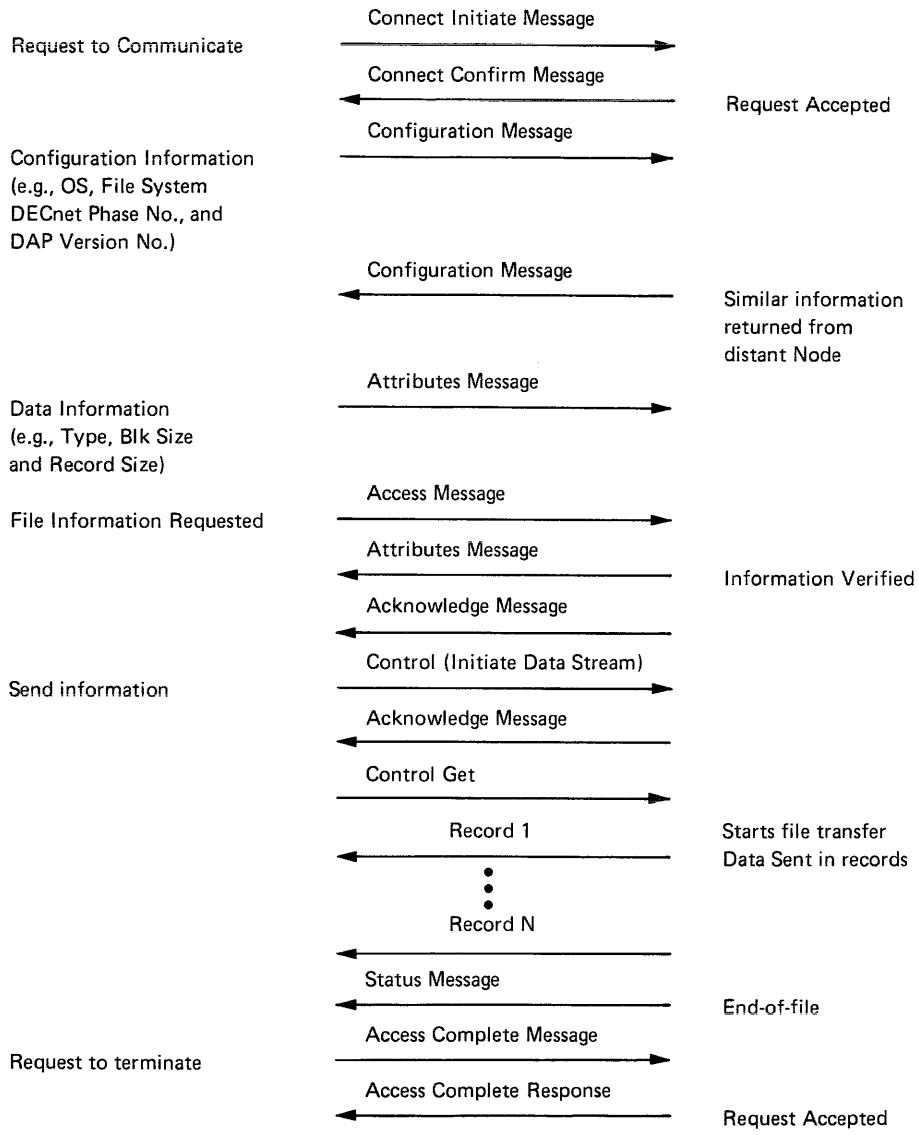


Figure 2-1. Typical DAP Message Exchange (Sequential File Retrieval)

3.0 MESSAGE FORMATS

3.1 Notation

The following notation is used to describe the DAP messages:

Field (length) : coding = description of field.

where:

Field = the name of the field being described.

length = 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 0 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.

4. The letters "I-n" means this is an image field, with n being a number that specifies the maximum length of 8-bit bytes in the image. 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=0). All 8-bits of each byte are used as information bits. The meaning and interpretation of each image field is defined with that specific field.

coding = 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 description.

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

3. All fields are presented to the Network Services 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:

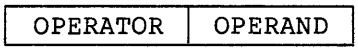
   a. A field that appears under a MENU may be omitted by setting the corresponding MENU bit zero.

   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, the default value is not used.

3.2 General Message Format

All DAP messages have the following form:



where:

OPERATOR = This field describes the characteristics and type of message. It is divided into four subfields, TYPE, FLAGS, STREAMID and LENGTH.

TYPE(1) : B = The type of DAP message. These numbers are given with each DAP message description. Types 196-255 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)
0	Stream identification field present.
1	Length field present.
2-6	Reserved.
7	Extension (0).

STREAMID(1) : B = The stream identification field. This field is included only if bit 0 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.

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. Messages between 1 and 255 bytes long may be blocked. Lengths greater than 255 bytes are sent unblocked or as the last part of a blocked message.

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

3.3 Configuration Message

The configuration message is used to pass system configuration information between both operating systems involved in a DAP Message exchange. This message is sent immediately following link establishment. 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 allocated for message exchange. The two cooperating DAP speaking processes will use the lesser of the two buffer sizes as the maximum size. If a system has an unlimited buffer size, it sends a 0 and the two systems will use the nonzero buffer size as the maximum.

OSTYPE(1) : B = Operating system type (the sending system). Values in the range 1-127 are reserved for DIGITAL use; 128-255 are reserved for user-specified operating systems.

Value	OS Type
0	Illegal
1	RT-11
2	RSTS
3	RSX-11S
4	RSX-11M
5	RSX-11D
6	IAS
7	VAX/VMS
8	Reserved for TOPS-20
9	Reserved for TOPS-10

FILESYS(1) : B = File system type (of the file system being used by the process sending this message). Values in the range 1-127 are reserved for DIGITAL use; 128-255 are reserved for user-specified file systems.

Value	System
0	Illegal
1	RMS-11
2	RMS-20
3	RMS-32
4	FCS-11
5	RT-11
6	No file system supported

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.

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 0.

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 0 by DIGITAL.

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

Bit	Meaning (When Set)
0	Supports file preallocation.
1	Supports sequential file organization.
2	Supports relative file organization.
3	Reserved - intended to support direct file organization.
4	Reserved - intended to support indexed file organization.
5	Supports sequential file access.
6	Supports random access by record number.
7	Extension (0 if subsequent fields are not needed).
8	Supports random access by Virtual Block Number.
9	Reserved - intended to support random access by Key.
10	Reserved - intended to support random access by user generated hash code.
11	Reserved - intended to support random access by Record File Address.
12	Reserved - intended to support ISAM.
13	Reserved - intended to support switching access mode.
14	Supports append to file access.
15	Extension (0 if subsequent fields are not needed).
16	Supports command file submission and/or execution.
17	Reserved - intended to support file compression.
18	Supports multiple data streams.
19	Supports status return (See ACCOPT field in ACCESS message).
20	Supports blocking of DAP Messages.
21-22	Reserved.
23	Extension (0).

3.4 Attributes Message

The Attributes message is used to describe how data is being represented in a file that is being transferred. The Attributes message is sent as a part of the initial setup. The Attributes Message format is:

ATTRIB	ATTMENU	DATATYPE	ORG	RFM	RAT	BLS	MRS	ALQ	BKS	FSZ	MRN	RUNSYS

DEQ	FOP	BSZ	DEV	SDC	NOK	NOA	NOR	CDT	RDT	EDT	OWNER	PROTSYS	PROTOWN

PROTGRP	PROTWLD

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:

ATTRIB : = The OPERATOR field with TYPE=2.

ATTMENU(EX-6) : BM = The bit map below specifies which of the indicated fields, groups of fields, or messages follow. Some groups of attribute fields can become extremely long. To resolve the buffering problem (i.e., all attributes messages must fit into a 256-byte buffer), these groups are split into separate messages and sent separately. The following bit map specifies which of the attributes fields will be present in the main attributes message (if the corresponding bit is set):

Bit	Meaning (when set)
0	DATATYPE
1	ORG
2	RFM
3	RAT
4	BLS
5	MRS
6	ALQ
7	Extension (0 if subsequent fields are not needed).

Bit	Meaning (when set)
8	BKS
9	FSZ
10	MRN
11	RUNSYS
12	DEQ
13	FOP
14	Reserved; intended for BSZ.
15	Extension (0 if subsequent fields are not needed).
16	Reserved; intended for DEV.
17	Reserved; intended for SDC.
18	Reserved; intended for summary attributes fields: NOK, NOA and NOR.
19	Reserved; intended for Date and Time attributes fields CDT, RDT and EDT.
20	Reserved; intended for use with the Protection attributes fields OWNER, PROTSYS, PROTOWN, PROTGRP and PROTWLD.
21	Reserved; intended for use with the Access Control List Message.
22	Reserved; intended for use with the Key Definition Message.
23	Extension (0 if the following field is not needed).
24	Reserved; intended for use with the Allocation Message.
25	Reserved; intended for use with the File Header Characteristics Message.
26-30	Reserved (0).
31	Extension (0).

DATATYPE(EX-2) : BM = The type of data being transferred. The default is IMAGE. This field is very important for file/record retrieval. Many file systems do not explicitly store with the file attributes, information as to whether the file contains ASCII, EBCDIC or Image data. Therefore, the contents of a file are interpreted according to the data type supplied by the user. In many cases, for DAP remote file access, where conversions may be necessary on ASCII files, this field (supplied by the user) is the only indication of data type.

Bit	Meaning (When Set)
0	ASCII (see Note 1).
1	IMAGE (see Note 2).
2	EBCDIC (Reserved).
3	Compressed Format (Reserved for future use).
4	Executable Code.
5	Privileged Code.
6	This bit is set if the attributes of the file (as stored with the file), match those specified in the accessing ATTRIBUTES message. This bit is used only in the ATTRIBUTES message being returned (i.e., by the accessed DAP process).
7	Extension (0).

Notes:

1. 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 set to zero.

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, or binary. It is up to the user process to determine how to use the data.

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$IDX;	Indexed (Reserved).
60	FB$DIR;	Direct (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 (disk only).
4	FB$STM; ASCII Stream Format (sequential files only).
5	FB$LSA; Line-sequenced ASCII format (sequential files only).

RAT(EX-3) : BM = Information about the attributes of individual records;

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	Reserved (0)-Intended for COBOL carriage control.
3	FB$BLK; Records that do not span blocks (see Note 2).
4	Records have embedded format control.
5-6	Reserved (0).
7	Extension (0).

Notes:

1. FORTRAN Carriage Control. For line printers and some terminals, the first character of each record is to be treated as a carriage control character according to the ANSI definition.

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 (e.g., disk and magnetic tape) this may be a factor in determining the actual format used on the device.

BLS(2) : B = Physical block size in bytes. 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 its buffer size.

If the accessed process receives a zero value, it should do one of the following:

a. If the accessed process has a limited buffer size, it should return its maximum buffer size to the accessing process.

b. If the accessed process has unlimited buffer space (i.e., dynamic buffering), it should return a zero value.

ALQ(I-5) : B = This field specifies the allocation quantity. For file creation, it specifies the initial size of the new file. The actual size of the new file is returned in this field.

NOTE

On opening existing files, this value is ignored. This field is used only to return the file size.

BKS(1) : B = Bucket size. Used only for access to relative (not RMS-20), direct and indexed files with RMS.

FSZ(1) : B = Size in bytes of fixed part of variable length record with fixed control format.

MRN(I-5) : B = Maximum record number for file (for relative files only). If set to 0, checking is suppressed.

RUNSYS(I-40) : 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, which is the amount of space, in blocks, added to the file each time the file is implicitly extended. Default will be the normal default for individual file systems.

FOP(EX-6) : BM = The File Access Options a user requires:

Bit	Meaning (when set)
0	FB$RWO; rewind on open.
1	FB$RWC; rewind on close.
2	reserved (0).
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 (0).
6	FB$ACK; check attributes specified for open against those in file and return error if they don't match.
7	Extension (0 if subsequent options are not needed).
8	FB$CTG; a contiguous file extension required.
9	FB$SUP; supersede existing file on create.
10	FB$NEF; do not position to EOF on opening magnetic tape file for PUT.
11	FB$TMP; create temporary file.
12	FB$MKD; create temporary file and mark for delete on close.
13	FB$FID; open by file ID.
14	FB$DMO; rewind and dismount magnetic tape on close.
15	extension (0 if the following bits are not used).
16	FB$WCK; Enable Write checking.
17	FB$RCK; Enable Read checking.
18	FB$CIF; create new file if one by the same name does not exist. If one does exist, open the file. Ignore version number.
19	FB$LKO; Override file lock on open (reserved).
20	FB$SQO; Sequential access only (reserved).
21	Reserved for maximum version number.
22	Reserved for spool file on close.
23	Extension (0 if subsequent fields not needed).
24	Reserved for submit command file.
25	Reserved for delete sub-option on submit command file.
26	Reserved for contiguous best try.
27-30	Reserved (0).
31	Extension (0).

BSZ(1) : B (Reserved for future use) = Number of bits per byte in the file data. The default value is 8. Zero is invalid value.

DEV(EX-6) : BM (Reserved for future use)

= For attributes sent to the accessing node. This field contains the generic characteristics of the device on which a file resides.

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 (e.g., magnetic tape).
6	; Reserved.
7	- extension (0 if subsequent bits are not set).
8	FB$FOD ; a file-oriented device (e.g. a disk or magnetic tape).
9	FB$SHR ; device can be shared.
10	FB$SPL ; device is being spooled.
11	FB$MNT ; device is currently mounted.
12	FB$DMT ; device is marked for dismount.
13	FB$ALL ; device is allocated.
14	FB$IDV ; device is capable of providing input.
15	; extension (0 if subsequent bits are not set).
16	FB$ODV ; device is capable of providing output.
17	FB$SWL ; device is software write-locked.
18	FB$AVL ; device is available for use.
19	FB$ELG ; device has error logging enabled.
20	FB$MBX ; device is a mailbox.
21	FB$RTM ; device is realtime in nature, not suitable for RMS use.
22	FB$RAD ; a random access device.
23	extension (0 if subsequent bits are not set).
24	- device has read checking enabled.
25	- device has write checking enabled.
26	- device is foreign, (i.e., not Files-11).
27	- network device.
28	- generic device.
29-30	- Reserved (0).
31	- Extension (0).

SDC(EX-6) : BM
(Reserved for future use)
= Spooling device characteristics. SDC uses the same bit definitions as in the DEV field.

NOK(1) : B
(Reserved for future use)
= Number of keys defined in file.

NOA(1) : B
(Reserved for future use)
= Number of areas defined in file.

NOR(1) : B
(Reserved for future use)
= Number of record descriptors in file.

CDT(18) : A
(Reserved for future use)
= Date and time file created in GMT.

RDT(18) : A
(Reserved for future use)
= Date and time file last updated in GMT.

EDT(18) : A
(Reserved for future use)
= Date and time file may be deleted in GMT.

The preceding three fields should be in the following format:

dd-mon-yybhh:mm:ss

where:

dd is the day
mon is a 3-letter abbreviation for the month
yy is the year
b is blank (space)
hh is the hour
mm is the minutes
ss is the seconds

OWNER(I-40) : A
(Reserved for future use)
= The name or user code (e.g., a UIC such as 240,220) of the file owner. This field is used only when returning the file's attributes. When creating a file, the file owner information is taken from the user identification information which comes with the connect.

PROTSYS(EX-3) : BM = File protection for system access rights.
(Reserved for
future use)

Bit	Meaning (When Set)
0	XB$RDV ; deny read access.
1	XB$WRV ; deny write access.
2	XB$EXE ; deny execute access.
3	XB$DLE ; deny delete access.
4	deny append access.
5	deny list access.
6	deny update access.
7	extension (0 if bits 8 and 9 are not used).
8	deny change access protection attribute.
9	deny extend access.

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

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

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

3.5 Access Message

The access message specifies the file name and type of access requested. This message is followed by a Control Message if data transfer over the link is requested. The optional FILESPEC in the message definition below is used only with file rename. The format for the access message is:

ACCESS	ACCFUNC	ACCOPT	FILESPEC	(FILESPEC)	FAC	SHR

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 a file (Reserved).
4 - $ERASE; Delete a file.
5 - Reserved.
6 - Directory List (Reserved).
7 - Submit as a command (batch) file.
8 - Execute command (batch) file.

In the case of ACCESS code 3 above, two file specifications are present. The first is the old file specification and the second is the new file specification.

If $CREATE is specified, but a file of that name already exists, the rules of the remote node for file creation will be followed. 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) is not concerned with functions beyond remote data access. DAP should not be extended to attempt to cover RJE, Spooling, etc., which, while they involve file transfer, are also concerned with command processing, parameter passing, job queueing, etc. The two command file submission commands are here for historical reasons. They have already been implemented in the first release of DAP-based software. However, their functionality will not be extended.

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

Bit	Meaning (When Set)
0	- I/O errors. A record may be skipped or repeated as specified by the Continue Message. I/O errors are not fatal.
1	- If set (1), a status message will be returned following each record sent to the accessed process in the record access mode.
2	- If set (1), return a status message 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 NSP message. When a user requires a record file address to be returned, this option is used.
3-6	- Reserved.
7	- Extension (0).

FILESPEC (VAR-128) : A = The file specification in the format required by the remote node. This ASCII field is treated as a quoted string by DAP software. It is delivered, as is, to the file system at the remote node.

FAC(EX-3) : BM = The file access operations a user requires:

Bit	Meaning (When Set)
0	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 access (see Note).
6	Reserved.
7	Extension (0).

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)
0	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 access.
6	FB$NIL; No access by other users.
7	Extension (0).

3.6 Control Message

The control message is used to send 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; get record. 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.
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 is used to initiate a data stream. If the STREAMID number is omitted, a default of zero is assumed.
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; indicates to the accessed system, that the information to follow should be written into the file.

Value	Meaning

5 - $DELETE; delete current record.

6 - $REWIND; rewind file. Reserved for future use.

7 - $TRUNCATE; truncate file. Writes End-of-file at current position. Used with sequential files only. Reserved for future use.

8 - Reserved for future use.

9 - $RELEASE; unlock record specified by Record File Address in KEY field. Reserved for future use.

10 - $FREE; unlock all locked records for this data stream. Reserved for future use.

11 - $SPACE; forward or backward space the file by the number of blocks specified in KEY below. Reserved for future use.

NOTE

KEY contains a two's complement number which is positive for forward spacing and negative for backward.

12 - $FLUSH; write out all modified I/O buffers and attributes for this data stream. Reserved for future use.

13 - $NXTVOL; perform end-of-volume and start-of-next-volume processing. Reserved for future use.

14 - $FIND; find record. Same as 1, but the data is not transferred. Reserved for future use.

15 - $EXTEND; extend this file by the amount specified in the following allocation attributes extension message. Reserved for future use.

16 - $DISPLAY; retrieve this file's attributes as defined by the field DISPLAY. Reserved for future use.

CTLMENU(EX-4):BM = The following bits when set, indicate optional fields are present:

Bit	Field
0	RAC
1	KEY
2	KRF (Reserved)
3	ROP
4	HSH (Reserved)
5	DISPLAY (Reserved)
6	Reserved (0)
7	Extension (0)

RAC(1) : B = Sets the access mode:

Value	Meaning
0	RB$SEQ; sequential record access.
1	RB$KEY; keyed access.
2	RB$RFA; access by Record File Address (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, each block must be requested by a Control message 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
Direct Files	Record Key
Record File Address Access Mode	Record File Address
Block Mode Access	Virtual Block Number (binary, range 1 to n)

KRF(1) : B (Reserved for future use) = Key of reference. If this field is not present, the key of reference is not changed. Default is primary if never set.

0 - primary key
1 - 255-secondary key indicator

ROP(EX-6) : BM = Optional record processing features. If this field is not present, the options in force for the last access are retained.

Bit	Meaning (When Set)
0	RB$EOF; position to EOF.
1-2	Reserved (0).
3	RB$HSH; use hash code in HSH (Reserved).
4	RB$LOA; follow fill quantities (Reserved).
5	RB$ULK; manual locking/unlocking (Reserved).
6	Reserved (0).
7	extension (0 if following not needed).
8	RB$RAH; read ahead (Reserved).
9	RB$WBH; write behind (Reserved).
10	RB$KGE; key is >=(Reserved).
11	RB$KGT; key is > (Reserved).

HSH(I-5) : B = Hash code if keyed access on direct file is employed and the user is doing hashing.
(Reserved for future use)

DISPLAY(EX-4) : BM= Attributes messages which are to be returned in response to a request to retrieve the file's attributes are:
(Reserved for future use)

Bit	Meaning (When Set)
0	main attributes message.
1	key definition attributes.
2	area definition attributes.
3	access control list.
4	file header characteristics message.

3.7 Continue Transfer Message

The continue transfer message is used when an error is detected in the I/O transfer and the user process wishes to continue DAP transfer. The normal use of this message would be to receive an error message, take appropriate action, and then send a continue transfer to allow the DAP link to resume operation. The format of the continue transfer message is:

CONTRAN	CONFUNC

where:

CONTRAN : = The OPERATOR field with TYPE=5.

CONFUNC(1) : B = This field is used to specify the recovery action to be taken:

Value	Meaning
1	Try again.
2	Skip this record and continue.
3	Abort transfer (i.e., discard all records in the pipeline until an access complete message is found indicating the pipeline is clear).

3.8 Acknowledge Message

The acknowledge message is used to acknowledge access commands and control connects. Its format is as follows:

ACKNOWLEDGE

where ACKNOWLEDGE: = The OPERATOR field with TYPE=6.

3.9 Access Complete Message

The Access Complete Message is used either to terminate access or to acknowledge a request to terminate access. The Access Complete Message format is as follows:

ACCOMP	CMPFUNC	FOP

where:

ACCOMP : = The OPERATOR field with TYPE=7.

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

Value	Meaning
1	command. Terminate access. Close a file that is currently open. When multiple data streams are in use, they are all closed-out ($CLOSE).
2	response. Sent by the accessed process in response to an Access Complete Command or a purge from the accessing process.
3	purge. A file is to be purged. That is, closed and deleted ($CLOSE + $ERASE).
4	end-of-stream. Terminate the data stream associated with this STREAMID number but do not close the file ($DISCONNECT).

NOTE

The Access Complete (EOS) does not require a response nor is the file closed. However, 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.

3.10 Data Message

The Data message is used to transfer the file data over a DAP link. When data messages are longer than the maximum NSP message size, the message should be sent using NSP segmentation. The Data message format is as follows:

DATA	RECNUM	FILEDATA

where:

DATA : = The OPERATOR field with TYPE=8

RECNUM(I-8) : B = This field is used to send the record number when accessing relative files (or in some cases sequential files in a relative manner). For random store, this field will contain the record number (for relative files) or hash code (if the user is generating his own hash codes with direct files). When RECNUM is not used, the byte count is zero. When in the block mode, this field will contain the VBN instead of the record number.

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

3.11 Status Message

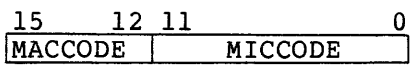
The status message is used to return information on the status of DAP messages or data transfers. It is sent synchronously in response to another message or an error during data transfer. The format is:

STATUS	STSCODE	RFA	RECNUM	STV

where:

STATUS = The OPERATOR field with TYPE=9

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



where:

MACCODE(4B):B = The macro or functional group reason for the error. Values for this field are specified in Table 3-1.

MICCODE(12B):B= The specific reason for the error (by MACCODE type). Values for this field are specified in Tables 3-2, 3-3 and 3-4.

RFA(I-8) : B = Used to return 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 destination file.

RECNUM(I-8) : B = Used to return 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). Null for non-relative files.

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

Table 3-1. MACCODE Field Values

Value (Octal)	Name	Meaning
0	Pending	Operation in progress.
1	Successful	Returns information that indicates success.
2	Unsupported	This implementation of DAP does not support 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	Errors that occur after opening a file and before closing that file.
6		Reserved.
7	Access Termination	Errors associated with terminating access to a file.
10	Format	Error in parsing a message. Format is not correct.
11	Invalid	Field of message is invalid (e.g., 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.
13-17		Reserved.

Table 3-2. 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 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 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).
	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); Reserved.
	02 37	Device characteristics field (DEV); Reserved.

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

Type of Error	Code (Octal)	Reason
	02 40	Spooling device characteristics field (SDC); Reserved.
	02 41	Number of keys in file field (NOK); Reserved.
	02 42	Number of areas in file field (NOA); Reserved.
	02 43	Number of record descriptors in file field (NOR); Reserved.
	02 44	File creation date and time field (CDT); Reserved.
	02 45	File revision date and time field (RDT); Reserved.
	02 46	File expiration date and time field (EDT); Reserved.
	02 47	File owner identification field (OWNER); Reserved.
	02 50	System access file protection field (PROTSYS); Reserved.
	02 51	Owner access file protection field (PROTOWN); Reserved.
	02 52	Group access file protection field (PROTGRP); Reserved.
	02 53	World access file protection field (PROTWLD); Reserved.
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 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).
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 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); Reserved.
	04 25	Record options field (ROP).

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

Type of Error	Code (Octal)	Reason
	04 26	Hash code field (HSH); Reserved for future use.
	04 27	Display attributes request field (DISPLAY); Reserved.
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 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).
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 20	Access complete function field (CMPFUNC).
	07 21	File options field (FOP).
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 20	Record number field (RECNUM).
	10 21	File data field (FILEDATA).
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 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).

Table 3-3. MICCODE Field Values for Use with MACCODE Values of 0, 1, 4, 5, and 7 Octal

Value (Octal)	Error/Reason
	NOTE MICCODE Format: Bits 0-11 contain the error code number. Symbolic status codes, where supplied, refer to the corresponding RMS status codes. They are included here for ease of reference only -- they have no meaning for DAP.
0	Unspecified error.
1	ER$ABO; operation aborted.
2	ER$ACC; F11-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.
7	ER$ANI; not ANSI "D" format.
10	ER$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$BOF; beginning of file detected.
20	ER$BPA; private pool address not multiple of "4".
21	ER$BPS; private pool size not multiple of "4".
22	ER$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$COD; invalid or unsupported "COD" field.
30	ER$CRE; F11-ACP could not create file (STV=sys err code).
31	ER$CUR; no current record (operation not preceded by GET/FIND).
32	ER$DAC; F11-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$DME; dynamic memory exhausted.
40	ER$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-F11ACP enter function failed.
46	ER$ENV; operation not selected in "ORG$" macro.
47	ER$EOF; end-of-file.
50	ER$ESS; expanded string area too short.
51	ER$EXP; file expiration date not yet reached.
52	ER$EXT; file extend failure.
53	ER$FAB; not a valid FAB ("BID" NOT = FB$BID).

Table 3-3. MICCODE Field Values for Use with MACCODE Values of 0, 1, 4, 5, and 7 Octal (Cont.)

Value (Octal)	Error/Reason
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$RAT; illegal record attributes.
131	ER$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$RFA; bad RFA value (RFA=0).
135	ER$RFM; invalid record format.

Table 3-3. MICCODE Field Values for Use with MACCODE Values of 0, 1, 4, 5, and 7 Octal

Value (Octal)	Error/Reason
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 (0, odd, or if BLK-IO not word aligned).
160	ER$USZ; invalid user buffer size (USZ=0).
161	ER$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.
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.

Table 3-3. MICCODE Field Values for Use with MACCODE
Values of 0, 1, 4, 5, and 7 Octal (Cont.)

Value (Octal)	Error/Reason
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.
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	; cannot submit command file.
307	NMF ; no more files.

Table 3-4. 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

4.0 FILE ORGANIZATION

4.1 Types of Files

The following types of files are addressed by this specification:

1. Sequential - Each record's position depends on the position of the previous record. Records may not be processed in any other order.

2. 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.

3. Direct/Indexed - These files have records organized according to some classification method, usually an access key. Within a particular key, the records are assumed to be 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 a vertical form effector (VFE), which is one of line feed (LF), vertical tab (VT), or form feed (FF).

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

DAP supports five ASCII record formats:

1. Fixed length records,

2. Variable length records,

3. Variable with fixed control,

4. Line sequential ASCII records, and

5. ASCII stream

In addition, DAP supports the following attributes:

1. FORTRAN carriage control,

2. COBOL carriage control,

3. Implied LF/CR envelope for printing,

4. Embedded carriage control, or

5. None of the above.

4.2.1 Conversions - Not all systems support all the above record formats and attributes. Transferring ASCII data between systems with different record formats and attributes sometimes require conversions.

If a conversion is necessary in transferring ASCII data between systems, the conversion must always be done by the accessing (user) process, not by the accessed (essentially passive server) process. In other words, all intelligence resides with the user. The remote server process does only as it is told and exhibits no intelligence.

Thus the accessed process will return an error when creating a file whose type, as specified in the ATTRIBUTES and ACCESS messages, is not supported by the remote system. Also, when accessing an existing file, data is transferred according to the attributes of the existing file at the remote node. That is, when retrieving data from an existing file, the data is sent with the attributes of the remote file. When updating or appending data to an existing file, the data sent by the user process must have the attributes of the existing file. No conversions will be performed by the accessed process.

Where conversions are necessary, they can be reduced to those shown in Table 4-1. For purposes of this table, variable length records, variable length records with fixed control, and line sequential ASCII records can all be considered as variable length records.

Table 4-1. Conversion Table

From \ To	Implied LF/CR		No Attributes (Attributes) Stream			
Implied LF/CR	1	N	2	C	3	C
No Attributes	4	X	5	N	6	C
Stream	7	C	8	N	9	N
FORTRAN/COBOL Carriage Control	10	X	11	C	12	C

where:

N = no conversion necessary.

C = conversion.

X = no conversion allowed (error).

Table 4-1 lists six situations in which conversions are allowed. They are:

2. A LF/CR envelope is placed around each record by the accessing process.

3. A CR/LF is appended to each record by the accessing process if the final character of the record is not already a VFE.

6. Same as 3.

7. The receiving node collects and concatenates successive records until a VFE is encountered. If the VFE is a LF preceded by a CR, the CR/LF is stripped and the record stored. Otherwise the record is stored as is.

11. FORTRAN or COBOL carriage control characters cause an FF or a CR followed by the appropriate numbers of LFs to be placed before or after the record depending on the rules for interpreting the control character. The control character is discarded.

12. Same as 11.

4.2.2 Conversion Rules and Restrictions - The following three rules should be observed when conversions are performed:

1. Any conversion of variable to fixed-length ASCII records should be done by the accessing process.

2. No attempt should be made to convert any file format into FORTRAN or COBOL carriage control form.

3. No attempt should be made to convert image files except between fixed and variable length records.

These techniques have the following effects and restrictions:

1. A file transferred from a system having the implied LF/CR envelope for records when transferred to a stream system will list without the initial LF.

2. Records transferred from a byte count system to a stream system containing embedded vertical form effectors (VFE's) will be subdivided on retrieval from the stream system by the VFE's. However, such files will list properly, regardless of the system listing them.

Any attributes other than ASCII or image data types or fixed/variable length records are useful only for local file system attribute storage. No translation or processing is mandatory. Thus, if a file is stored as EBCDIC, but an ASCII retrieval is done, it is not required, or even desirable, to translate the data. Some systems may choose to do so, but it is currently outside the scope of this document to detail that type of operation.

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 ones except the maximum length is that specified in the MRS field.

4.3.3 Variable with Fixed-Control Format Records - These records are normal variable-length ones 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 LSA Records (Line Sequential ASCII Records) - These records are variable-length ones that have as their first five characters a 5-digit ASCII line number. This line number is in binary at the RMS interface. It is sent across the network as three bytes of binary data preceding the ASCII data.

4.3.5 ASCII Stream - Here a file contains just a stream of ASCII characters with no real concept of records. However, Vertical Field Effectors are used to delimit "records" for purposes of reading and writing the file.

5.0 OPERATION

The Data Access Protocol (DAP) is a user level protocol that resides within the Application Layer of the DIGITAL Network Architecture (DNA). Its purpose is to transfer 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 user level within a DECnet system. They use the Network Service Protocol and the network facilities for the creation and flow control of the data path (logical link) between the processes exchanging messages within the DAP environment. The originating process issues a Connect Initiate command requesting the creation of a logical link to the DAP process at the destination node. This request may specify an actual process name or the generic DAP object type. All systems must support connection to the generic DAP object type as a minimum, if device-independent file access is to be performed. The destination DAP process completes the connection by returning a Connect Confirm command. Once the link is established, the processes may now exchange DAP messages over the link.

A separate logical link is used for each remote file access. This means a single user can not access more than one file at a time over a single logical link. It also means several users cannot access the same file over a single logical link.

During link establishment, the identity of the user is obtained from the Connect Initiate Message.

Following link establishment, the DAP-speaking processes exchange configuration messages for four purposes:

1. To establish the maximum buffer size for exchanging NSP messages.

2. To identify system type to each other;

3. To enable the other DAP process to know which version of the protocol this process speaks; and

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 in deciding if block mode file transfer can be used when transferring files between like systems or 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. It is used to determine the type of file support offered by a remote system without resorting to trial and error techniques. It can be used to help produce more positive, useful error messages.

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 is the one that initiates the connection. For each DAP message sent over the link, a transmit request and corresponding receive request must be issued to NSP. This discussion is concerned with 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 specifying the desired mode and format of the data. This is then followed by an access message specifying the desired operation. The Attributes and Access messages may be blocked and sent together in one transmission if buffer space is available (the LENGTH field must be used) and blocking is supported as indicated by the Configuration message. Data messages are the only messages whose length can exceed 256 bytes.

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. When storing records with systems retaining attributes, the accessed system uses the attributes message sent by the accessing process to indicate the attributes of the records being sent by the accessing process. For record retrieval with systems retaining attributes, records are transferred with the attributes of the attributes message returned by the accessed process. File systems that store attributes often have no information as to the type of data stored in the file. When this is the case, the user-supplied data type in the attributes message is used to determine whether any conversion is necessary when transferring records from one type of system to another.

After the initial set-up messages are sent, the accessing system will receive a response from the accessed processs. If the access specified opens a file, an Attributes Message followed by an Acknowledge Message will be sent from the accessed system containing the actual attributes of the accessed file. If the operation specified in the Access Message deletes or renames a file, no Attributes Message or Acknowledge Message is returned. The response is an Access Complete Message.

To minimize the tying up of network resources (e.g., logical links and buffers), the Configuration, Attributes, 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. 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 Status Messages. Status Messages are always sent as ordinary (not interrupt) DAP messages.

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 just disconnects. 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, it should be assumed that the SYSCAP field has been extended in the future version of DAP. This error should be ignored. This is the only time when an error is ignored. The assumption is that the more sophisticated DAP process will use only a subset of the protocol and thus both sets of software will be able to work together. This standard specifies the DAP for version 4.1.

5.1.1 Errors in the Setup Sequence - Errors in the Configuration Message, Attributes Message, and Access Message, all return a Status Message. 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 (e.g., an attempt to open a file already open by another process).

3. Start a different access. A new access will usually start 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 should be capable of accepting any of the three setup messages in response to a Status Message. Table 5-1 provides a list of responses to setup message errors.

Table 5-1. Responses to Setup Message Errors

Error	Responses		
	Configuration Message	Attributes Message	Access Message
Configuration Message	1	0	0
Attributes Message	1	1	2
Access Message	1	1	1

where:

0 = invalid response.
1 = valid response.
2 = valid response only for accesses requiring no attributes message.

5.1.2 Setup Sequence - If a timer is being 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.

```
     Accessing Process                Accessed Process

1.   Establish link:

         issue connect------>
                      <------complete connect

2.   Configuration information exchange:

         CONFIGURATION------>
         Message
                      <------CONFIGURATION
                             Message

3.   Setup for access:

         ATTRIBUTES------>
         Message

         ACCESS    ------>
         Message

                   <------ATTRIBUTES
                          Message

                   <------ACKNOWLEDGE
                          Message
```

NOTE

An Attributes Message is not required when the Access Message specifies the ERASE, RENAME, or EXECUTE command file. For these types of access, neither the Attributes Message nor the Acknowledge Message is returned by the accessed process.

Figure 5.1. Set-up Sequence

4. Error in configuration messages:

(a) CONFIGURATION------>(received in error)
Message

<------STATUS Message

disconnect ------>

or

(b) CONFIGURATION------>(received in error)
Message

<------STATUS Message

CONFIGURATION------>
Message

or

(c) if the error is in the returned message

CONFIGURATION------>
Message

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

disconnect ------>

5. Error in Attributes Message:

(a) ATTRIBUTES------>(received in error)
Message

<------STATUS Message

disconnect------>

or

(b) ATTRIBUTES------>(received in error)
Message

<------STATUS Message

ATTRIBUTES------>
Message

or

(c) ATTRIBUTES------>(received in error)
Message

<------STATUS Message

ACCESS ------>
Message

Figure 5-1 (Cont.). Set-up Sequence

6. Error in access message:

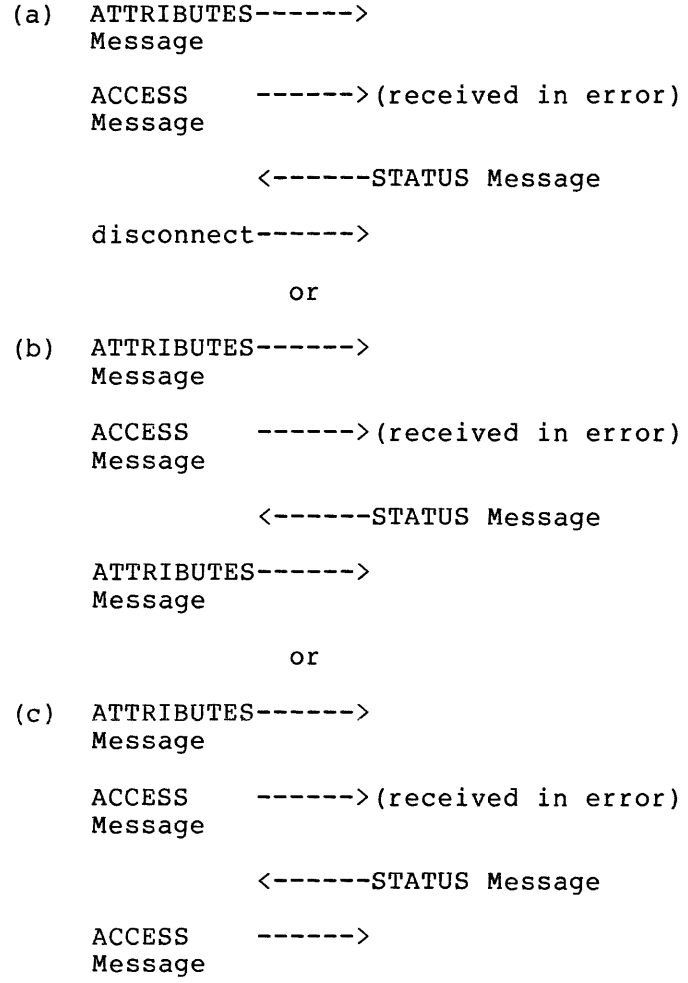


Figure 5-1 (Cont.). Set-up Sequence

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 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 0 is used.

The sequence for initiating a data stream is as follows:

```
Accessing Process   Accessed Process

CONTROL(connect)---->

                    <----ACKNOWLEDGE
```

If an error occurs, a Status Message will be 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.

5.2.1 Sequential File Retrieval - For sequential file retrieval, data records are sent from the accessed system. Once the initial startup sequence is completed and the data stream initiated, a single Control (GET) Message is sent to start data records flowing. Thereafter, the file records are transmitted without waiting for any further DAP messages to control sending messages. All flow control is performed implicitly by the Network Service Protocol.

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 will then send file records 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 will continue to arrive until: a) the end-of-file is reached on the accessed system; b) an error occurs in accessing the file; or c) 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 will be 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 Command and waits for an Access Complete Response. It then either disconnects or initiates another access by sending a setup sequence. If the accessing system receives an error, 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 command and waits for an access complete response in return. In such cases, an accessing system issuing an access complete command may still receive one or more records of the file or even an end-of-file indication or an error indication 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.

```
 Accessing Process                Accessed Process
```

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

```
    CONTROL (GET)------>
                 <------[STATUS,] RECORD 1

                         NOTE
                   Transfer continues until .
                   End-of-File or error      .
                                             .
                         [STATUS,] RECORD n

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

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

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

NOTE

The Status Messages in square brackets are optional depending on whether they are asked for in the ACCOPT field of the Access Message. If they are required, they should immediately precede the data message in a single NSP message. The optional status message precedes the data message (record) so that it is always possible to block the two DAP messages for transmission as a single unit even if the data message is longer than 255 bytes.

2. Retrieval until error:

(a) Accessing Process Accessed Process

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

When an error is received, the Accessing Process can request link termination.

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

or

(b) request the information be sent again

```
    CONTINUE (Try again)------>
                         <------RECORD n
```

or

(c) skip that record and continue

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

Figure 5-2. File Retrieval Sequence

3. Retrieval with access termination:

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

The accessed process may set a timer following sending ACCOMP and if neither a disconnect or another message is received within the time interval it may disconnect the link.

Figure 5-2 (Cont.). File Retrieval Sequence

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. These messages will be accepted by the accessed system and will continue until the accessing system sends an Access Complete Command. This procedure will cause a corresponding Access Complete Response to be returned following successful file closure, or a Status Message to occur if an error is incurred in closing the file. In either case, the access is concluded and another access may start or the link may be disconnected.

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 in the Control Message where "position to EOF" is also specified. As with sequential file retrieval, sequential file storage implies the use of only one data stream. If optional status messages are desired, the ACCOPT field of the access message must be used to request them.

If an error occurs during record transfer, the accessed system will return a Status Message. This must always be replied to with a Continue Message sent as an interrupt message (because of possible pipelining). In addition, if it is desired to terminate the access, an Access Complete Message should be sent.

```
 Accessing Process              Accessed Process
```

1. Store with no errors:

```
CONTROL (PUT)------>
RECORD 1    ------>
            [<------STATUS]
      NOTE          .
Transfer continues  .
until access        .
complete or error   .

RECORD n          ------>
                  [<------STATUS]
ACCOMP (COMMAND)  ------>
                  <------ACCOMP (RESPONSE)
```

Figure 5-3. Sequential File Storage

NOTE

The Status Messages (in square brackets) are optional depending on whether they are asked for in the ACCEPT field of the Access Message. They are not used to indicate an error condition. An error will be contained in a Status Message without brackets (see Step 2).

2. Error during transfer:

(a) Purge the new file and terminate

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

NOTE

The accessed system will discard records until ACCOMP (PURGE) is received.

Purge incomplete file <------ACCOMP (RESPONSE)

or

(b) close the new file and terminate

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

NOTE

The accessed system discards records until ACCOMP (COMMAND) is received.

close incomplete 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     ------>
```

Figure 5-3 (Cont.). Sequential File Storage

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 message and the retry is successful, the accessed process will post a receive and carry 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. Continue messages must be sent in interrupt mode as there may be data in the pipeline.

3. If the accessing system wants to stop a sequential file storage operation before it is complete and purge the incomplete file on the accessed system, the sequence is as follows:

```
Accessing Process        Accessed Process

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

ACCOMP (PURGE)  ------->

purge incomplete file<----ACCOMP (RESPONSE)
on accessed system
```

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

Figure 5-3 (Cont.). Sequential File Storage

5.2.3 Record Retrieval - Record retrieval is similar to sequential file retrieval except that a control message (with a record key for random retrieval) must be sent by the accessing process for each record accessed. Record retrieval is specified by the accessing process setting sequential record access, Keyed access or Record File Address access in the Control Message. Block mode transfer is similar to record retrieval and is specified by setting Virtual Block Number access.

For keyed or Record File Address access, the sequence is as follows:

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

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

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

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

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

Once the location of a particular record in a file is found using random access, the user frequently wants to get subsequent records sequentially. This can be done by switching the access mode from keyed or Record File Address to sequential in the Control Message and issuing a GET. (With RMS systems, the user is free to switch access modes according to the RMS rules.)

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

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

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)---->
                                <----[STATUS,]RECORD t

CONTROL (sequential file access, get)----->
                                     <----[STATUS,]RECORD t+1
                                     <----[STATUS,]RECORD t+2, t+3,
                                          ... to end-of-file
```

Error handling for sequential record retrieval is similar to error handling for sequential file retrieval. When an EOF is reached while accessing a file sequentially, the accessed process sends the Status Message "end-of-file-detected." This prevents automatic file closure and control is retained by the accessing process.

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 (e.g., record number out of range or record not found).

5.2.4 Record Store - This is similar to sequential file store in messages exchanged. For relative files, the data messages must include the relative record number field specifying the number of the record (RECNUM). For direct 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. For indexed files, RECNUM is null. For sequential files, records are written starting at the current position within the file.

The access message specifies whether to open an existing file or create and open a new file. PUT access must have been specified in the Control Message. For record storage, the accessing process may specify sequential record access, or keyed access. Optionally, VBN access may also be used.

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 procedure is inefficient 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.

When an accessed RMS system must return Record File Addresses to the accessing RMS system (bit 1 of ACCOPT in the Access Message set), the sequence for record storage with return of status is as follows:

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

         <-------- STATUS

RECORD n+1 ------>

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

Errors are handled as indicated in Section 5.2.2 except for the use of continue skip.

5.2.5 Append to Existing File - The append operation is identical to sequential store and applies only to sequential files. The records are placed at the logical end of the file by the accessed system:

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

RECORD 2-------->
```

If it is necessary to return Record File Addresses, the sequence is the same as that described for Record Store (see Section 5.2.4).

5.2.6 Deleting a File - The delete operation 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 that a store operation be done on the data that follows in a temporary file and that this file be submitted to a batch-type facility upon access completion (closing of the file). The file will be deleted following execution by the batch facility. DAP does nothing with regard to any feedback from the batch facility and does not guarantee that the file actually executes in the batch monitor. The file is transferred using sequential file storage.

The "execute-as-command-file" requests that the specified file be only submitted to the batch facility. No data follows this command (the specified file having been 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.

5.3 Closing a File and Terminating Data Streams

The ACCOMP End of Stream (EOS) command is used to terminate a data stream. When the accessing process wishes to terminate a data stream, it may do so by sending it the appropriate STREAMID number to terminate. This is particulary useful when multiple data streams are employed. This will not close the file even if it terminates the last active data stream.

An ACCOMP (COMMAND) is used to close the file and terminate the access, which includes closing out all remaining active data streams.

5.4 Terminating a Logical Link

The logical link is terminated 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 will usually follow 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. Disconnection will normally occur only at the end of a group of transfers.

5.5 File Security and Protection

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 whose 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 user wants to change the account under which he is running at the remote node, he must disconnect the logical link and reconnect specifying the new account in the connect.

APPENDIX A

GLOSSARY

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.
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 NSP 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 (e.g., 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.

APPENDIX B

RSX/IAS/RT DECNET IMPLEMENTATIONS

DECnet remote file access (and transfer) is implemented via three distinct pieces of software: a File Access Listener (FAL), a set of user callable subroutines (Network File Access Routines) called NFARS and a Network File Transfer utility (NFT). A brief description of each is provided below.

B1.0 FAL

FAL is the mechanism which maps DAP protocol messages to the local file system. FAL accomplishes this by accepting DAP file access requests from the user on the network side and mapping them into equivalent requests to the local file and/or operating system. Figure B-1 is the FAL State Diagram.

FAL is a user level process, resident on every node whose file system is to be accessed via the network. It is a passive element in that it services requests for remote access to the local file systems, it does not generate activity by itself, and is idle (suspended) when no such requests are in progress. Requesting processes are connected to FAL through the network provided communication mechanism. The Data Access Protocol (DAP) is used for exchanging commands and data between FAL and the accessing process. A single FAL process can handle multiple accesses and logical links simultaneously.

B2.0 NFARS

To simplify remote file access a set of FORTRAN callable subroutines, NFAR's are provided. The routines build, send, and interpret DAP messages for the user. The basic functions provided by the user interface are reflected in the NFAR's to effect remote file access. The NFAR's accomplish this functionality by communicating with the cooperating remote task FAL over the network using DAP messages.

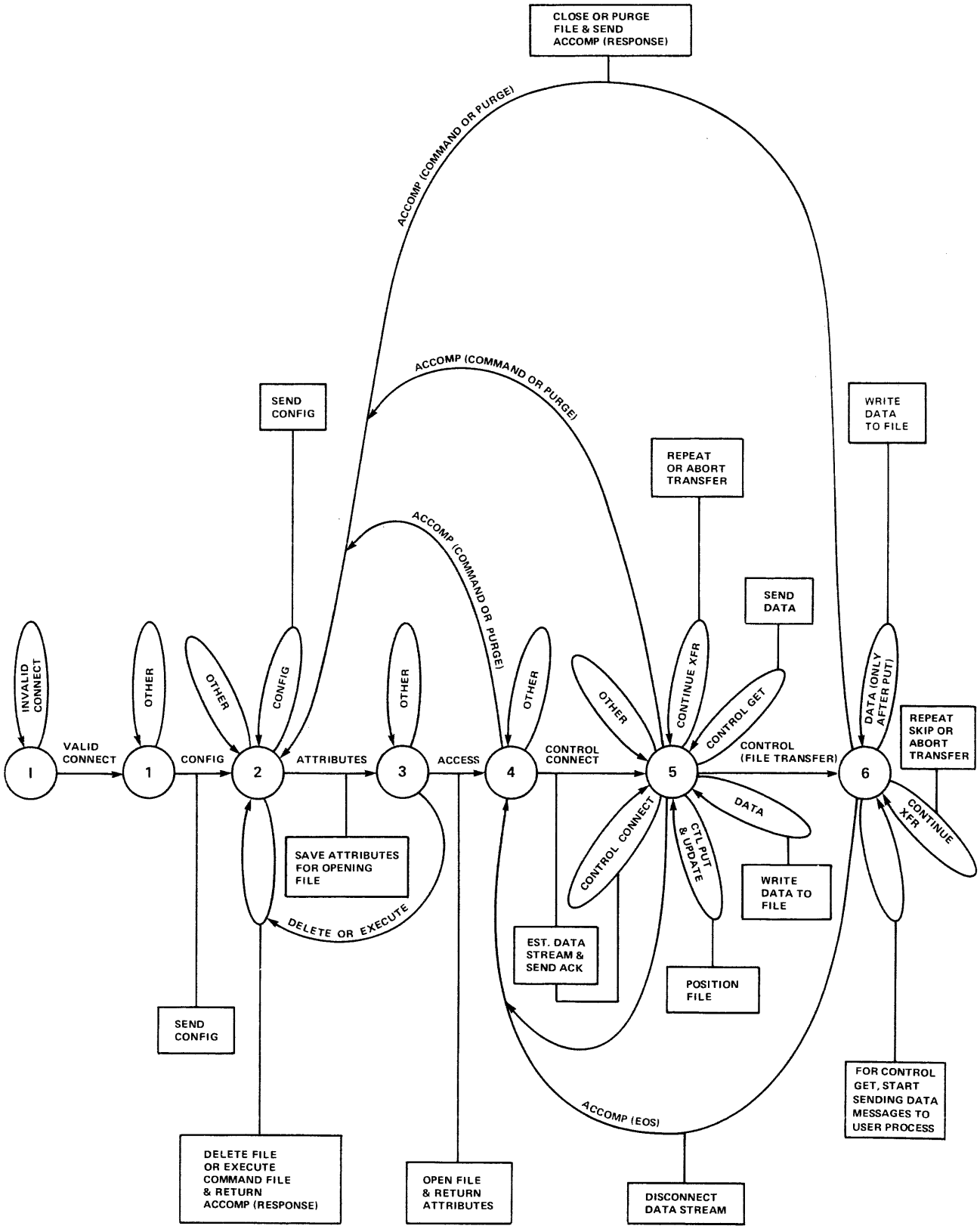


Figure B-1. FAL State Diagram

B3.0 NFT

NFT is an internode file manipulation utility which allows a user to:

a) transfer files to a remote node;
b) retrieve files from a remote node;
c) delete a file at a remote node;
d) execute command files at a remote node; and
e) submit command files to a remote node for execution there.

NFT calls the NFAR's directly, as user programs do, to perform the requested operations. It maps commands entered by the user, into NFAR calls which are interpreted by the FAL process on the remote node. For example in a network with nodes A, B, and C, a user on node A could transfer files between: A and B, A and C, or B and C using NFT.

APPENDIX C

REVISION HISTORY

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 are:

a. DAP Version 1.0 could not adequately support indexed and ISAM file access;

b. The format of the operator field has been expanded;

c. The USERID message has been eliminated;

d. The status and error message have been combined;

e. The ACCESS COMPLETE Message has been added;

f. The CONFIGURATION Message has been added; and

g. The two types of DATA Messages employed in Version 1.0 have been merged into one DATA Message in Version 4.1.

While a definite incompatibility exists between Versions 1.0 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.

digital

digital equipment corporation

Printed in U.S.A.