HP 1000

NS Message Formats Reference Manual

Services

[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
nf	1530 [illegible]	NS NT
[illegible]	15[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
PTOP	[illegible]	[illegible]
PXP	1541 [illegible]	[illegible]
[illegible]	[illegible]	[illegible]
aUT	1570 /TCP	NSVT (Streamwise)
rfa	2560 /TCP	NS Remote [illegible]
nssTM	2564 /TCP	[illegible]

HP 1000

NS
Message Formats

Reference Manual

HEWLETT PACKARD
11000 WOLFE RD., CUPERTINO, CA 95014

Part No. 5958-8523
E1292

Printed in U.S.A. DEC 1992
Fourth Edition

PRINTING HISTORY

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The dates on the title page change only when a new edition or a new update is published. No information is incorporated into a reprinting unless it appears as a prior update; the edition does not change when an update is incorporated.

The software code printed alongside the date indicates the version level of the software product at the time the manual or update was issued. Many product updates and fixes do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one to one correspondence between product updates and manual updates.

First Edition . . . . . MAY 1986
Update 1. . . . . OCT 1986
Update 2. . . . . AUG 1987
Second Edition . . . . . OCT 1989
Third Edition. . . . . AUG 1991
Fourth Edition . . . . . DEC 1992

LIST OF EFFECTIVE PAGES

The List of Effective Pages gives the date of the most recent version of each page in the manual.

Effective Pages	Date
All	DEC 1992

PREFACE

This manual, the *NS Message Formats Reference Manual*, describes data communication messages and headers passed between computer systems communicating over Distributed System (DS) and Network Services (NS) links.

This manual includes headers that will appear in trace files generated by the NS-ARPA/1000 and NS/3000 tracing facilities. These include Data Link and Transport layer headers, such as the IEEE 802.3 header and the TCP (Transmission Control Protocol) header, as well as headers accompanying DS/1000-IV Compatible Services and NS Common Services.

NOTE

Throughout this manual, NS/3000 refers to NS on the HP 3000 MPE V and XL, unless otherwise noted.

Organization of This Manual

This manual contains the following sections:

Section 1 **Link Layer and Transport Layer Services**--describes the IEEE 802.3 header, the Ethernet header, the Internet Protocol (IP) header, the Transmission Control Protocol (TCP) header, the Packet Exchange Protocol (PXP) header, the UDP header, the IFP header, the ARP header, and the Probe protocol header and messages.

Section 2 **Socket Registry Messages**--describes messages for Socket Registry.

Section 3 **Network File Transfer Messages**--describes messages for Network File Transfer (NFT).

Section 4 **File Transfer Protocol Messages**--describes messages for File Transfer Protocol (FTP).

Section 5 **Remote Process Management Messages**--describes messages for Remote Process Management (RPM).

Section 6 **Virtual Terminal Access Messages**--describes messages for Virtual Terminal Access on the HP 3000.

Section 7 **RTE-RTE DS/1000-IV Compatible Transport and Services Headers**--describes the Router/1000 headers that accompany messages generated by NS Common Services transmitted over a Router/1000 link, and messages generated by DS/1000 IV Compatible Services transmitted over any type of link, by an NS-ARPA/1000 system.

Section 8 **RTE-MPE DS/1000-IV Compatible Services Messages**--describes messages transmitted between HP 1000s and HP 3000s (RTE-MPE or MPE-RTE) using DS/3000 and DS/1000 IV communications software.

Appendix A **Well-Known Addresses and Ports**--lists well-known TCP SAPs, UPD Ports, Ethernet Type Fields, MAC Address Vendor Codes, and Ethernet Multicast and Broadcast Addresses.

Related Publications

Refer to the publications listed below for more information about NS, DS, and Hewlett-Packard products supporting X.25 protocols.

NS-ARPA/1000 manuals:

NS-ARPA/1000 Generation and Initialization Manual (91790-90030)
NS-ARPA/1000 Maintenance and Principles of Operation Manual (91790-90031)
NS-ARPA/1000 User/Programmer Reference Manual (91790-90020)
NS-ARPA/1000 BSD IPC Programmer's Guide (91790-90060)
NS-ARPA/1000 User/Programmer DS/1000 Compatible Services Reference Manual (91790-90050)
NS-ARPA/1000 Quick Reference Guide (91790-90040)
NS-ARPA/1000 Error Message and Recovery Manual (91790-90045)

NS/3000 manuals:

NS3000/V Network Manager Reference Manual, Volume I (32344-90002)
NS3000/V Network Manager Reference Manual, Volume II (32344-90012)
NS3000/V User/Programmer Reference Manual (32344-90001)
NetIPC 3000/V Programmer's Reference Manual (5958-8581)
NS3000/V Error Message and Recovery Manual (32344-90005)
NS3000/XL Configuration Planning and Design Guide (36922-61002)
NS3000/XL Operations and Maintenance Reference Manual (36922-61005)
NetIPC 3000/XL Programmer's Reference Manual (36920-61005)
NS3000/XL Error Messages Reference Manual (36923-61000)

DS and DS-related manuals:

DS/1000-IV User's Manual for RTE-A and RTE-6/VM (91750-90012)
DS/1000-IV Network Manager's Manual Generation and Initialization for RTE-A and RTE-6/VM (91750-90013)
DS/1000-IV Theory of Operation and Troubleshooting for RTE-A and RTE-6/VM (91750-90014)
DS/1000-IV Quick Reference Guide for RTE-A and RTE-6/VM (91750-90015)

DS/3000 HP 3000 to HP 3000 User/Programmer Reference Manual (32185-90001)
DS/3000 HP 3000 to HP 3000 Network Administrator Manual (32185-90002)
DS/3000 HP 3000 to HP 1000 Reference Manual (32185-90005)

X.25 and LAN manuals:

DSN/X.25/1000 Reference Manual (91751-90002)
DSN/X.25/1000 Advanced Guide (91751-90003)
X.25 Link for the HP 3000 Reference Manual (32187-90001)
NS X.25 3000/V Link Guide (24405-90002)
DS X.25 to NS X.25 Migration Guide (24405-90001)

HP 12076A LAN/1000 Link Node Manager's Manual (12076-90002)

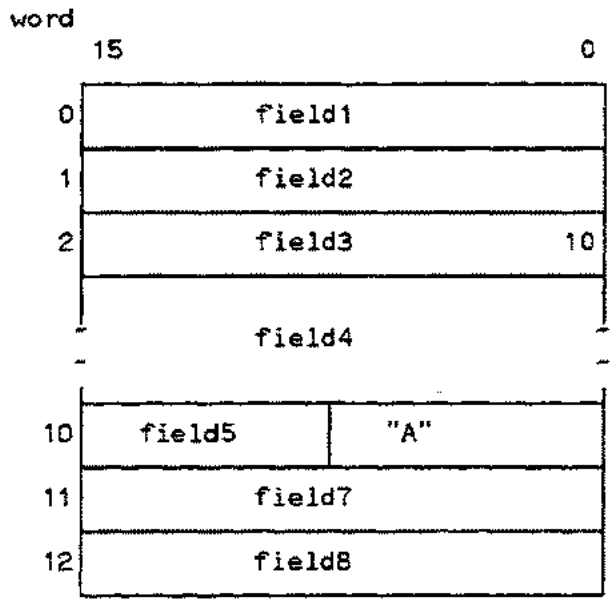
CONVENTIONS USED IN THIS MANUAL

In this manual, diagrams are used to describe the format of various messages and headers passed between network nodes.

The sequence of multiple headers within a message, and the sequence of messages (in cases in which multiple messages appear in a trace file) is described in the *NS-ARPA/1000 Maintenance and Principles of Operation Manual* and the manuals for NS/3000.

General Format

The diagrams in this manual usually take the form shown in the example below:



In the sample above, as in similar illustrations throughout this manual, each full horizontal field represents a single 16-bit word. Each word is labeled with a number for easy reference. The top-to-bottom sequence in which words of a given message are shown represents the sequence of these words in the message (or header), and the sequence in which words are transmitted. Within an individual word, bits are numbered from right to left (0 to 15), with the least significant bit (bit 0) shown as the rightmost in the diagram.

In a few instances, the diagram format used is slightly different than that shown above. In these cases, each full horizontal field represents a 32-bit word.

Transmission and Storage Sequence

Words of messages are transmitted from one node to another over a link in the top-to-bottom sequence indicated by the diagrams in this manual. Within each word, bits are *transmitted* according to the following convention: The least significant bit of the most significant (8 bit) byte is transmitted first, followed by the remaining bits of that byte, in order of significance. This byte is then followed by the least significant byte, with its bits transmitted in order of least significant to most significant. This means that for a single word, bits are transmitted in the following sequence (read from left to right):

8 9 10 11 12 13 14 15 0 1 2 3 4 5 6 7

The *storage* sequence of a message is system-dependent. For HP 1000s, messages are stored in the same bit sequence in which they were transmitted.

Field Labels

Field labels correspond to the meanings of the values that will be found in them in an actual message. The meanings of these names are described for each diagram. In some cases, field names correspond to parameter names for calls or commands that are described in detail in other Hewlett-Packard manuals. In these cases, the field names used in this manual correspond to these parameter names. Such cases, as well as the appropriate manual to refer to, are noted at the beginning of the sections in which this situation occurs.

Numerical Representation

Throughout this manual, numerical values included within full message or header diagrams are octal values. These values correspond to the octal values generated by the NS-ARPA/1000 tracing program NSTRC.

The illustration of a field may include a numerical value or a parameter represented by a single character. For example, word 2 in the example shows a field named `field3` that includes the value 10. This means that the entire word is the field denoted by the label `field3`, and that the octal value 10 (8 decimal) is represented by the word's least significant four bits.

Illustrations showing the individual bits that comprise a word or byte show the value of each bit field as a 0 or a 1.

Other Symbols

The symbols

indicate fields that are not shown in full. For example, `field4` in the sample above consists of words 3 through 9, which are not pictured.

Field values that are surrounded by quotation marks (") indicate that the field contains the ASCII character shown. For example, the right byte of word 10 in the example contains the ASCII value for the character A. The notation " " in a field indicates the field is filled with an ASCII blank.

The notation

`n (label)`

where `n` is some (octal) number and `label` is some descriptive name indicates that the value in the field will be `n` and that the label, or field name in parenthesis is what the value represents.

The notation

%nnn

represents a number (*nnn*) in octal.

CONTENTS

CONTENTS (continued)

CONTENTS (continued)

Section 6
VIRTUAL TERMINAL ACCESS MESSAGES

Section 7
RTE-RTE DS/1000-IV COMPATIBLE TRANSPORT AND SERVICES

CONTENTS (continued)

CONTENTS (continued)

Section 8
RTE-MPE DS/1000-IV COMPATIBLE SERVICES MESSAGES

CONTENTS (continued)

LINK, NETWORK, AND TRANSPORT LAYER PROTOCOL HEADERS

SECTION 1

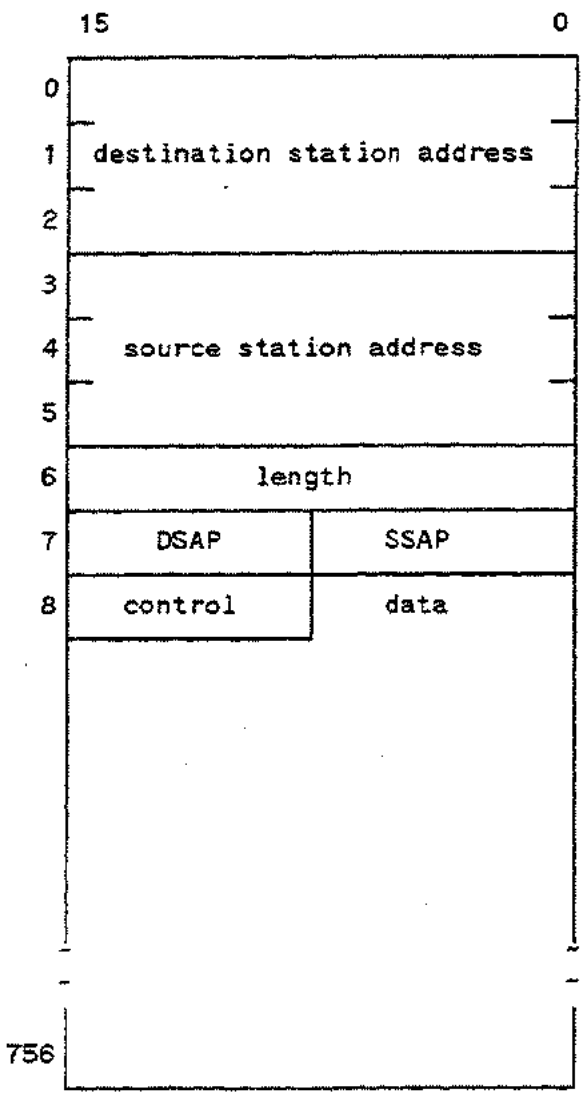
This chapter illustrates headers and messages of protocols used to provide data link layer (OSI layer 2), network layer (OSI layer 3), and transport layer (OSI layer 4) functions over IEEE 802.3 and Router/1000 links. This does not include transport for DS/1000-IV Compatible Services, which is described in Sections 3 and 4 of this manual.

These headers and messages include the following:

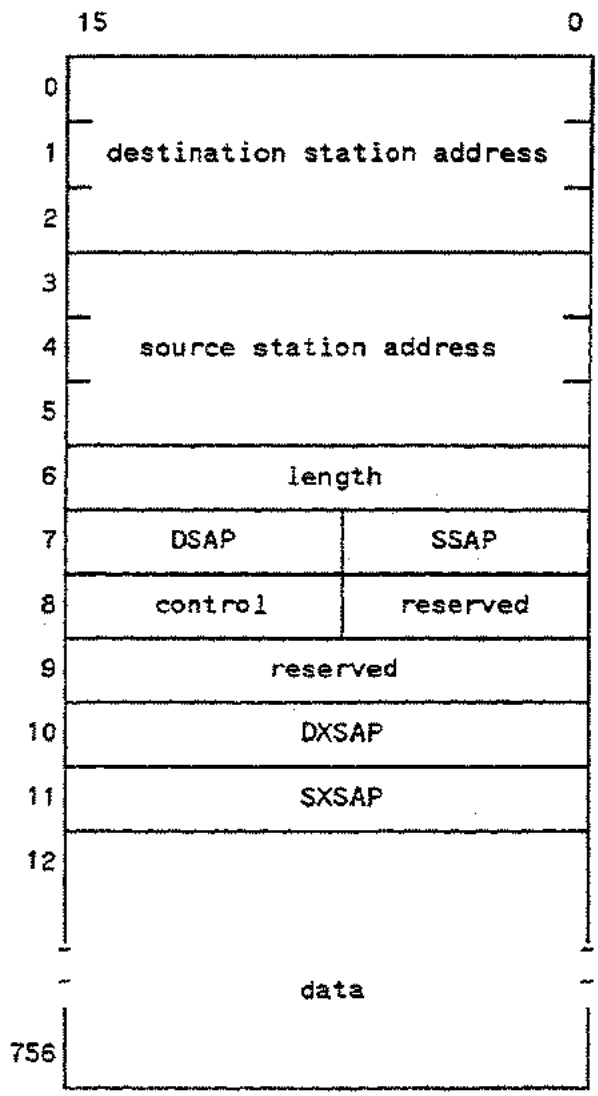
- the IEEE 802.3 Header
- the Ethernet Header
- the Internet Protocol Header
- the Transmission Control Protocol (TCP) Header
- the Packet Exchange Protocol (PXP) Header
- the User Datagram Protocol (UDP) Header
- the Interface Protocol (IFP) Header
- the ARP and Probe Protocol Header and messages
- Router and Non-Router Control Buffer Messages

IEEE 802.3 Header

Standard
IEEE 802.3 Header



Standard
IEEE 802.3 Header
with HP Expansion Header



The standard header is 17 bytes.
The standard header with an HP expansion SAP used is 24 bytes.
The total number of bytes for header and data is 1514 bytes.

destination station address	The address of the intended receiver node. If the bits in the address are all ones, the address is a broadcast address, meaning that the message is addressed to all nodes on the LAN. If bit 8 of the first word (word 0) is set to one (1), the address is a multicast address which may be used to address more than one node.

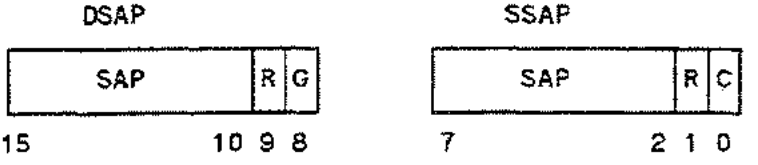
source station address — The address of the sending node.

length — The length of the packet in bytes; includes the DSAP field through the end of the data.

DSAP (Destination Service Access Point) and
SSAP (Source Service Access Point):

The DSAP and SSAP fields are used for dispatching individual packets within a node. The SAP fields are bits 10 through 15 in the DSAP and bits 2 through 7 in the SSAP. The SAP fields contain either an IEEE-defined SAP or an HP-defined SAP. The only IEEE-defined SAP currently defined is 6 for IP. If the SAP of one of the HP Address Expansion protocols is given, then dispatch information is provided in the HP expansion SAP fields, DXSAP and SXSAP (described below.) In this case, data for the protocol addressed by the expansion header starts in the 25th byte instead of the 18th byte.

The DSAP and SSAP fields should be the same in a given IEEE 802.3 header. The bits for the DSAP and SSAP fields are defined below:



R = 1 if the SAP is an IEEE-defined SAP.
= 0 if the SAP is an HP-defined SAP.

G = 1 if the SAP is a group SAP.
= 0 if the SAP is an individual SAP.

C = 1 if the incoming packet is a response.
= 0 if the incoming packet is a command.

SAP = 6 for IEEE SAP for IP
= HP SAP is one of the following values:
= 370 octal (F8 hexadecimal) for Network Management Services
= 374 octal (FC hexadecimal) for other services and non-IEEE protocols

data — Data starts in the 18th byte with the standard header. Data starts in the 25th byte with the standard header with DXSAP and SXSAP also given.

control The control field has one of the values shown below:

```
XID (exchange identifier)  = 277 octal
TEST                       = 343 octal (without poll bit set)
                           = 363 octal (with poll bit set)
UI (unnumbered information)= 003 octal
```

Because the IEEE 802.3 Link Interface software expects only UI packets, the receipt of XID or TEST packets is logged as an error.

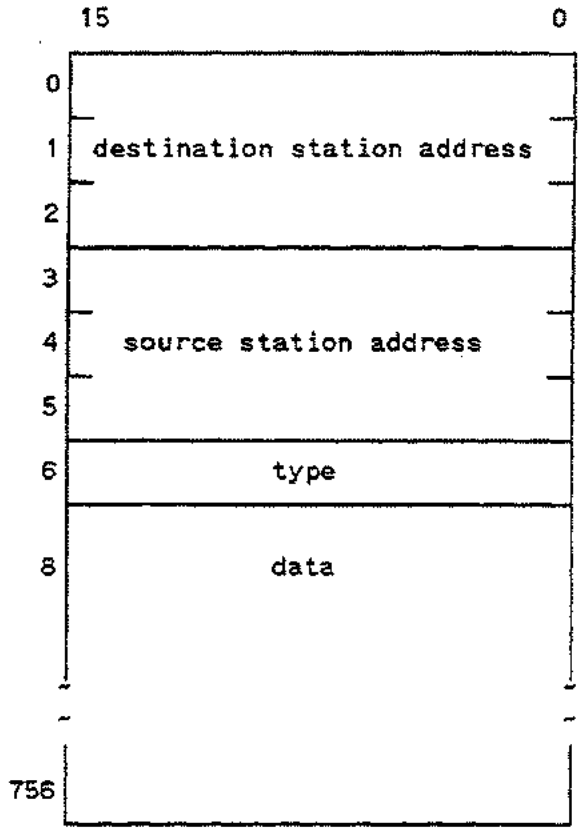
reserved The reserved fields must be all zeroes. These fields are present if DXSAP and SXSAP are used. Otherwise, data starts in the 18th byte.

DXSAP (Desination Expansion Service Access Point) and
SXSAP (Source Expansion Service Access Point):

The DXSAP and SXSAP identify a particular service or protocol. Values in the DXSAP and SXSAP fields are defined internally within HP. Some possible values are shown below:

SSAP and DSAP fields:	Possible DXSAP/SXSAP values:
Network Management services --370 octal (F8 hexadecimal)	3020, 3021, 3022, 3023, 3024, 7002 octal
Other services and non-IEEE protocols--374 octal (FC hexadecimal)	2403 octal (PROBE protocol)

ETHERNET Header



The standard header is 14 bytes.
The total number of bytes for header and data is 1514 bytes.

destination station	The address of the intended receiver node. If the bits in the address are all ones, the address is a broadcast address, meaning that the message is addressed to all nodes on the LAN. If bit 8 of the first word (word 0) is set to one (1), the address is a multicast address which may be used to address more than one node.
source station address	The address of the sending node.
type	The type field identifies the higher-level protocal associated with the Ethernet frame. It is uninterpreted at the data link level. It is specified at this level because a uniform convention for the placement and assignment of this field is crucial if multiple higher level protocols are to be able to share the same Ethernet network without conflict.

The value for TCP/IP in the type field is: 1000 (octal) or 0800 (hex).

Refer to Table A-3 "Ethernet Type Fields" in Appendix A for a list of valid values for the Ethernet type field.

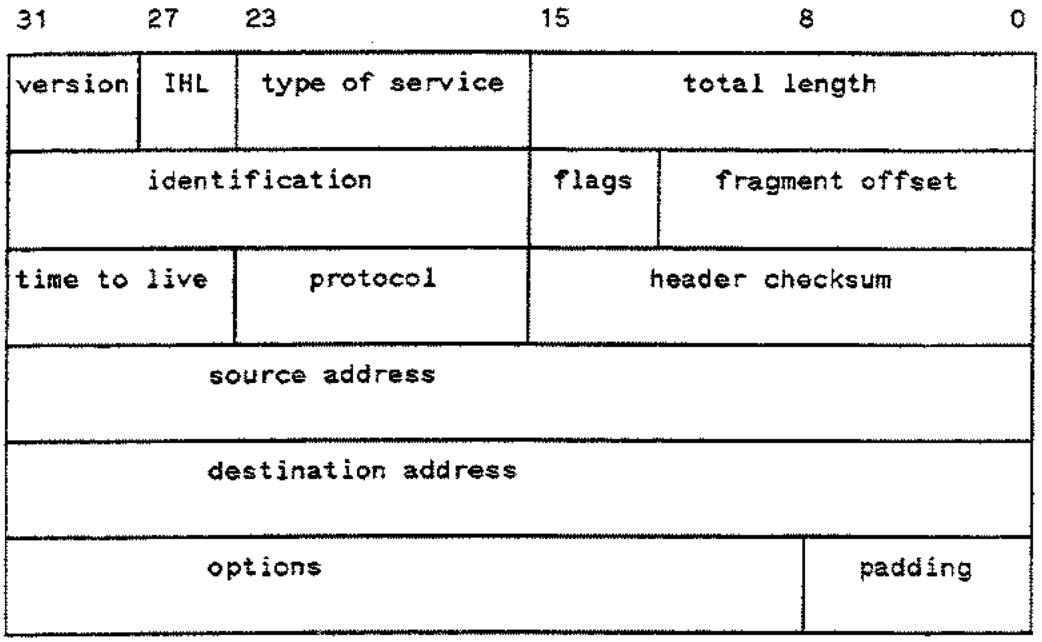
data

The data field contains 46-1500 bytes. Each octet (8 bits) contains any arbitrary sequence of values. Therefore, full transparency is provided. The data field is the information received from layer three. The information or packet from layer three is broken into frames of information of 46-1500 bytes by layer two. A minimum packet size is used to guarantee distinguisability from a collision fragment. If the data field is less than 46 bytes, it is padded with undefined characters. Upon receipt of this packet, the user does not know how much of the packet is valid data and how much contains undefined characters.

NOTE

Each packet is transmitted from the first byte to the last byte. However, each byte is transmitted least significant bit first to most significant bit last.

Internet Protocol (IP) Header



IHL — Internet Header Length. The IHL is the length of the internet header in 32-bit words.

type of service — The type of service byte is set to 0 by NS-ARPA/1000 and NS/3000.

total length — The length of the datagram, in bytes, including the IP header and data.

identification — A value assigned by the sending IP to aid in the correct reassembly of fragments.

flags — The three-bit flag field has the following format:

R =reserved. This bit is always 0.
D =Don't Fragment bit. This bit is set if no fragmentation should occur.
M =More Fragments bit. This bit is set if the current fragment is not the last fragment.

time to live — indicates maximum time a datagram is allowed to remain in the internet. TTL is decremented at each Internet Gateway. the datagram is destroyed if TTL reaches zero.

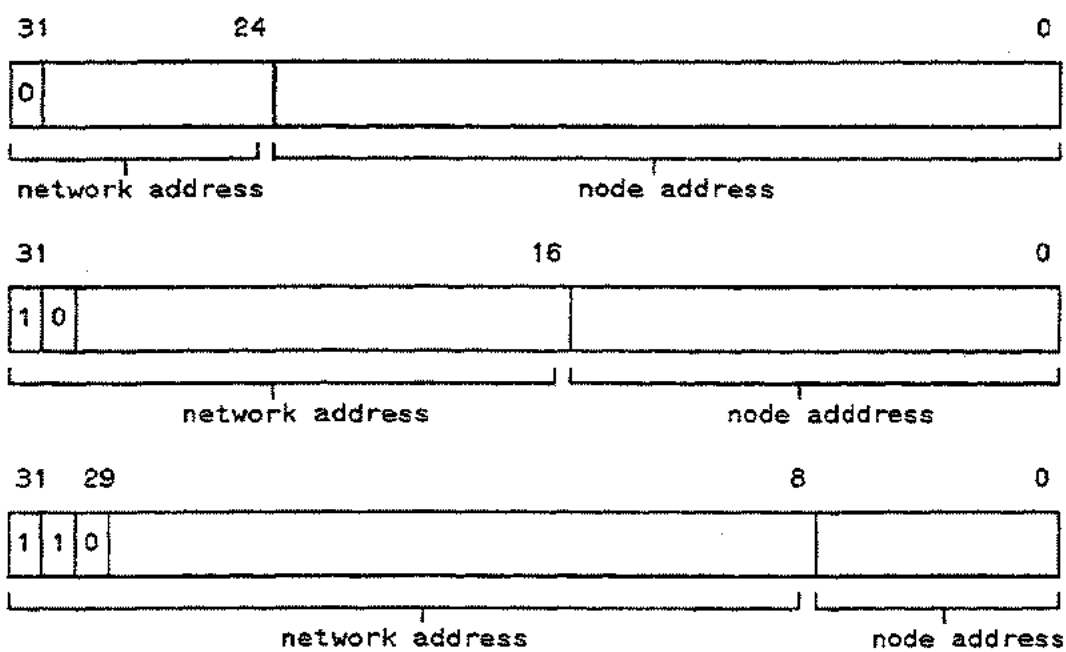
fragment offset — Indicates where in the datagram the current fragment belongs. The fragment offset is given in units of 8 byte pieces of data.

protocol — The protocol field indicates the next higher level protocol used in the data portion of the internet datagram. The following values indicate particular protocols:

Protocol Name	Value (octal)	
ICMP	1	
TCP	6	
HPPXP	361	
UDP	21	(not implemented for NS-ARPA/1000)

source and destination addresses:

These fields are formatted in one of three ways, depending on the values of the highest order bits as shown below:



Note: A network address of 0 indicates the local network.

IP addresses are commonly displayed as

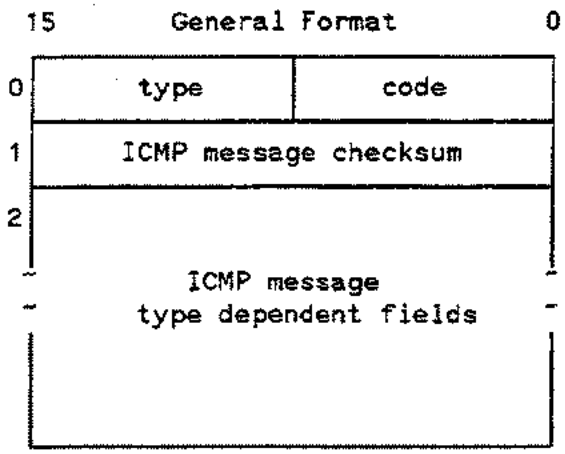
nnn.nnn.nnn.nnn

where nnn is a decimal number from 0 to 255 representing one byte of the four byte address.

options — Not generated by NS/3000 or NS-ARPA/1000 nodes, but processed by NS/3000 gateway and NS/3000 end destination IPs.

padding — Used by NS/3000 nodes to insure that IP Header ends on a 32-bit boundary. Padding is zero.

Internet Control Messages Protocol (ICMP) Messages (Appended to IP Header as an Upper Level Protocol (ULP))



ICMP Message Types

Type (decimal)	Code (decimal)	Meaning
0	0	Echo Reply
Destination Unreachable Messages (Type 3):		
3	0	Network Unreachable
3	1	Host Unreachable
3	2	Protocol Unreachable
3	3	Port Unreachable
3	4	Fragmentation Needed But Don't Fragment Bit Set
3	5	Source Route Failed
4	0	Source Quench
Redirect Messages (Type 5):		
5	0	Redirect for Network
5	1	Redirect for Host
5	2	Redirect for Type of Service and Network
5	3	Redirect for Type of Service and Host
8	0	Echo Request

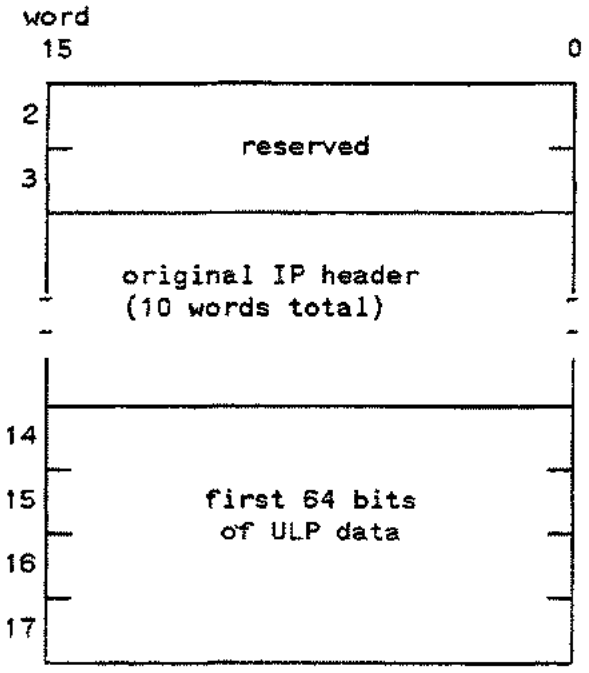
(ICMP Message Types continued on next page)

(ICMP Message Types continued from previous page)

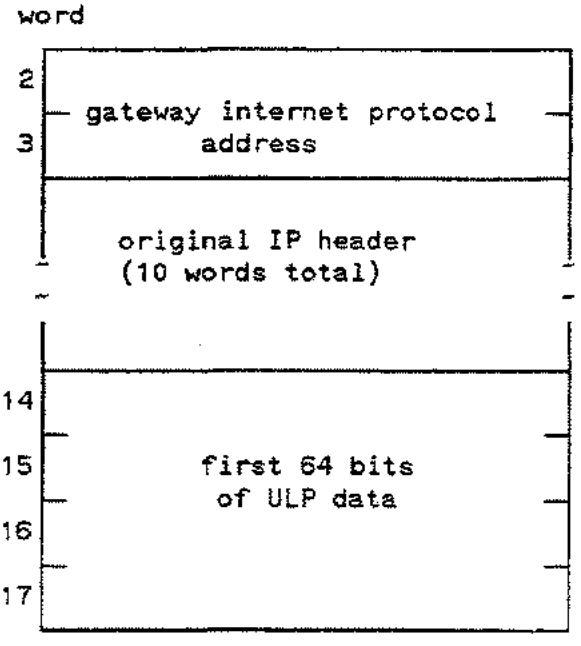
Time Exceeded Messages (Type 11):		
11	0	Transit Time Exceeded
11	1	Fragment Reassembly Time Exceeded
12	0	Parameter Problem in Options
13	0	Timestamp Request
14	0	Timestamp Reply
15	0	Information Request
16	0	Information Reply

ICMP Message Type Dependent Fields

Message types: Destination Unreachable (type 3), Source Quench (type 4), and Time Exceeded (type 11)

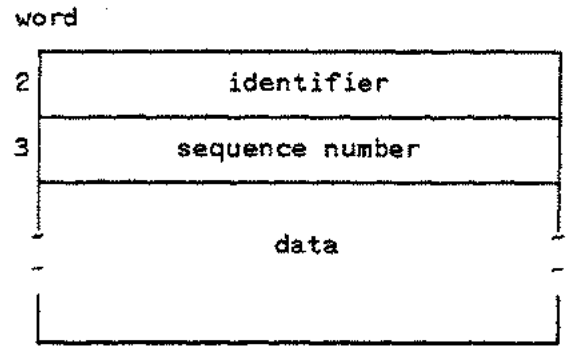


Message type: Redirect (type 5)

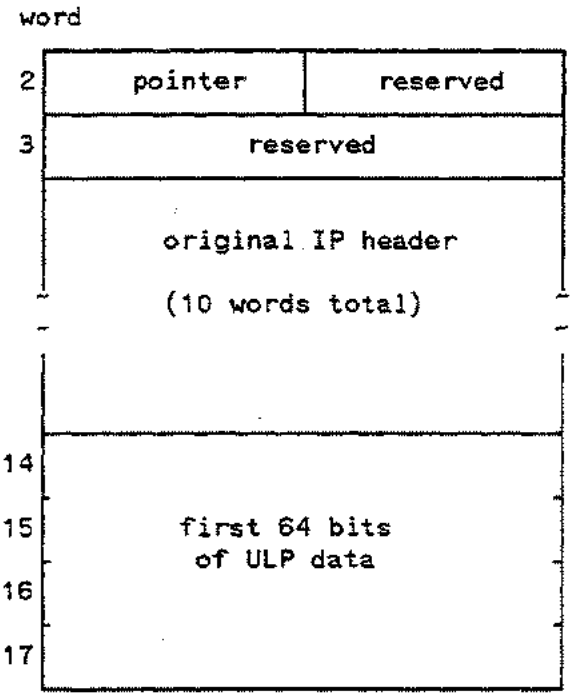


ICMP Message Type Dependent Fields, continued

Message types: Echo Request
(type 8) and Reply (type 0)

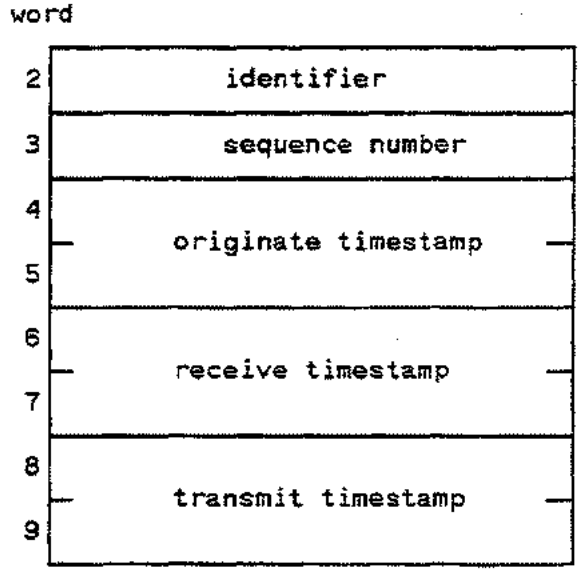


Message type: Parameter
Problem (type 12)

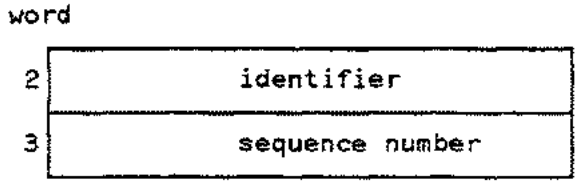


ICMP Message Type Dependent Fields, continued

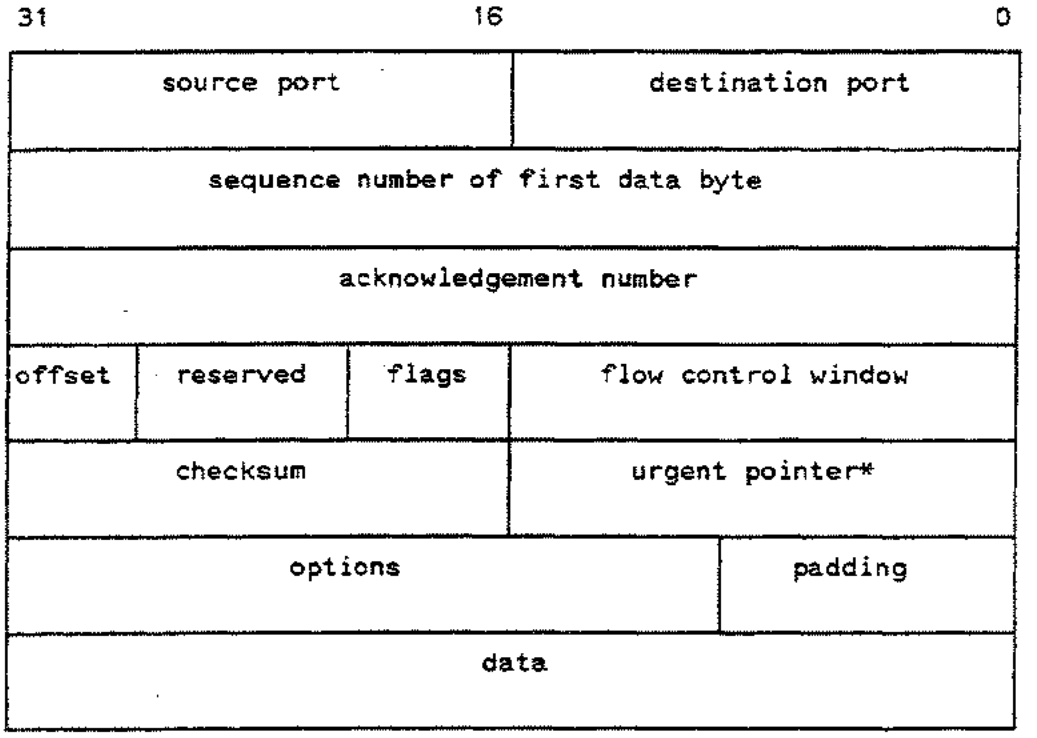
Message types: Timestamp Request (type 13) and Reply (type 14)



Message types: Information Request (type 15) and Reply (type 16)



Transmission Control Protocol (TCP) Header



*	not implemented for NS-ARPA/1000 or NS/3000.
source ports, destination port	port identifiers that must be concatenated with the IP address in order to distinguish ports (with same port IDs) on different nodes
sequence number	number of first byte of data (every byte is numbered)
acknowledgement number	next sequence number sender expects to receive
data offset (4-bit field)	word offset of data portion from beginning of message (= number of 32-bit words in TCP header)
flags (6 bits)	the flags shown below are set to indicate the information described:

U=URG: message contains urgent data (not implemented for NS-ARPA/1000)

A=ACK: acknowledgement piggybacked

P=PSH: transmit data immediately ("push" flag)

R=RST: reset the connection

S=SYN: synchronize sequence numbers (sequence number above is new Initial Sequence Number and first data is ISN + 1)

F=FIN: release connection

window — number of additional data bytes sender is willing to accept (beginning with the one indicated by the acknowledgement number field)

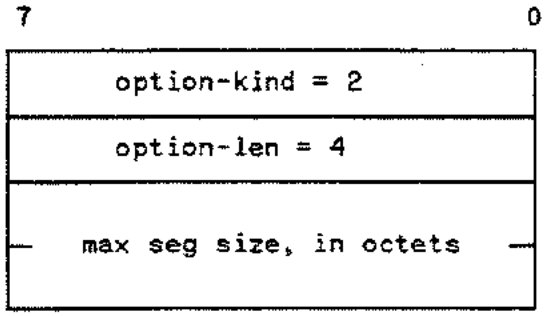
checksum — 16-bit ones complement of ones complement sum of all 16-bit words in header and data. The field value will be 0 if no checksum was computed.

urgent pointer — The urgent pointer field is set to 0 if the connection was opened with the checksum option (via NetIPC IPCCONNECT call) on; otherwise, this field contains the packet's beginning sequence number.

options — option kinds:
0 = end of option list
1 = no-operation
2 = maximum segment size

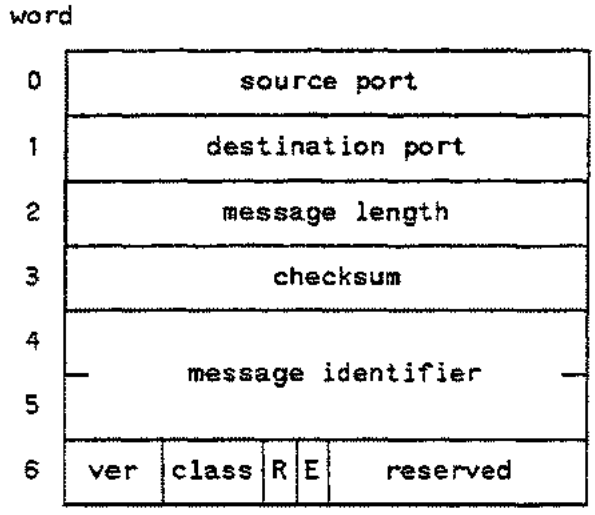
Options may be one of two formats:
1. single octet of option-kind, or
2. 1 octet of option-kind, and
   1 octet of option length, and
   option data.

For example, maximum segment size:



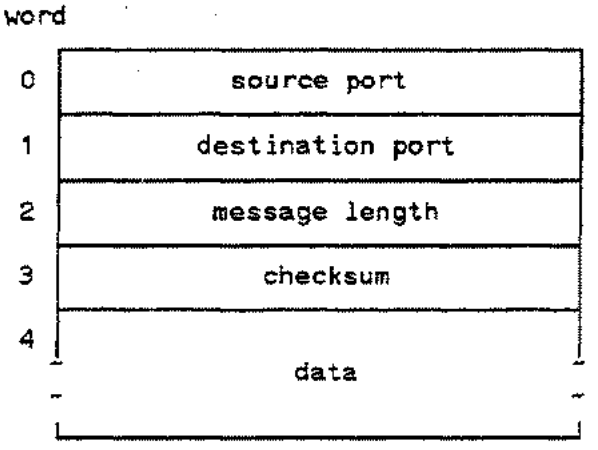
padding	variable length field as necessary to pad TCP header to fall on 32-bit boundary

Packet Exchange Protocol (PXP) Header



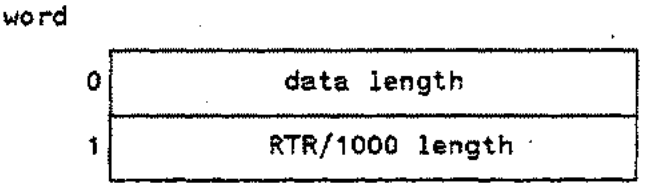
source port and destination ports	ports of the sending process %3005 is the well-known port for Socket Registry. Refer to Section 2 for the format of Socket Registry messages.
message length	length of message in bytes, including header and data
checksum	a checksum of 0 indicates that no checksum was calculated
ver	version number of HP's PXP protocol
class	class currently set to 1
R	reply bit. If the reply bit is set, the packet contains a reply from a server. If R is not set, the packet contains a master request.
E	error bit. If the error bit is set, an error was detected during request packet processing.

User Datagram Protocol (UDP) Header



source port and destination ports	ports of the sending process source port may be 0 when a reply is not required or when the reply address is based on other information
message length	length of message in bytes, including header and data
checksum	a checksum of 0 indicates that no checksum was calculated

Interface Protocol (IFP) Header (NS-ARPA/1000 only)



data length	user data length, in bytes.
RTR/1000 length	length of the Router/1000 header in bytes. This length varies depending upon the type of service used.

ARP vs. PROBE

ARP (Address Resolution Protocol) and PROBE reside in the Nework Layer and provide name server and requester functionality.

PROBE is more complicated than ARP and can be used to obtain additional information. PROBE takes a destination node's name from the upper layer protocol and sends a request out over the network which will be received by all or most nodes. The destination node replies with its local name, internet address, link address, and protocol stacks supported.

PROBE also sends unsolicited messages to announce its presence on the LAN and allow routing tables to be updated. These messages are sent every 5 minutes. A node may also generate a PROBE package indicating that is is about to be removed from the LAN to allow interested nodes to update their routing information.

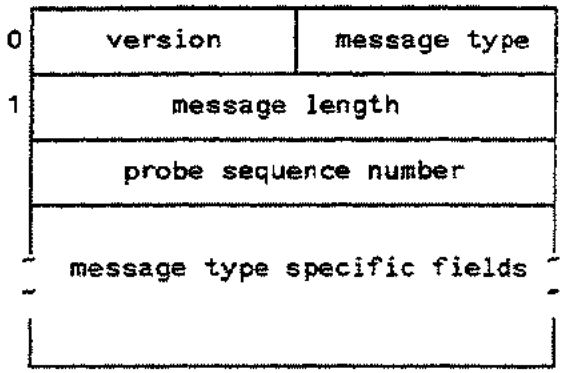
PROBE transmits its requests and unsolicited messages via a LAN multicast. Only systems configured to listen to the multicast address 09-00-09-00-00-01 (requests and unsolicited messages) and 09-00-09-00-00-02 (probe proxy requests) will pick up PROBE packets.

ARP is a simpler protocol in that it simply requests link level addresses and replies to requests for addresses. ARP may be used to try to resolve internet address into LAN station address. It does not generate any unsolicited messages.

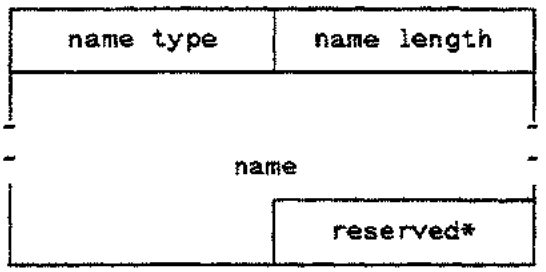
An ARP search begins when a node broadcasts a request onto a LAN. It terminates either when a uni-cast reply is received from another node or upon timeout after three tries without receiving a valid response.

PROBE Protocol Messages

Common Header Format



Standard Name Format



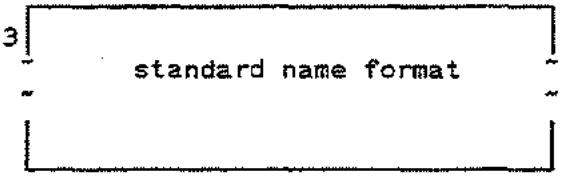
* The reserved byte, used for padding, will be present only if the environment name is of odd length.

PROBE Message Types

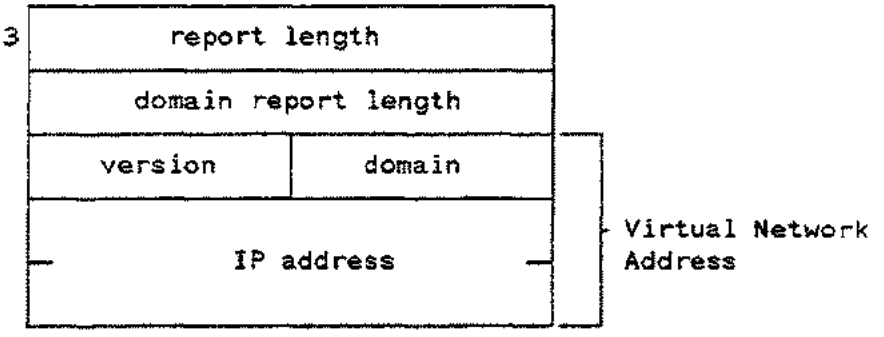
Type (decimal)	Name
16	Name Request
17	Virtual Address Request
18	Gateway Request
19	Name Reply
20	Virtual Address Reply
21	Unsolicited Reply
22	Gateway Reply
23	Proxy Name Request
24	Proxy Name Reply
25	Node Down

PROBE Message Type Specific Fields

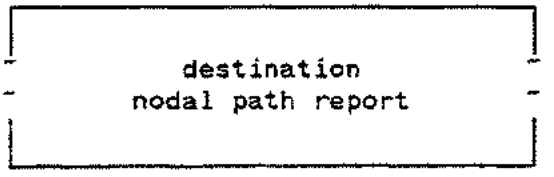
Name Request (type 16), Proxy Name Request (type 23)



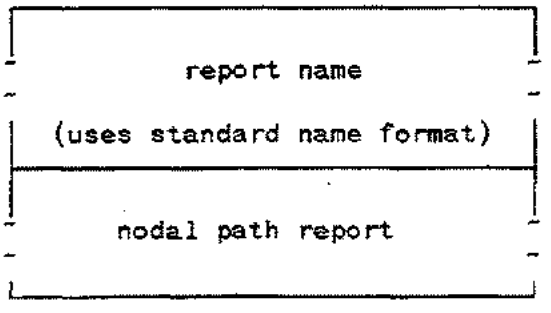
Virtual Address Request (type 17)



Name Reply (type 19), Proxy Name Reply (type 24)



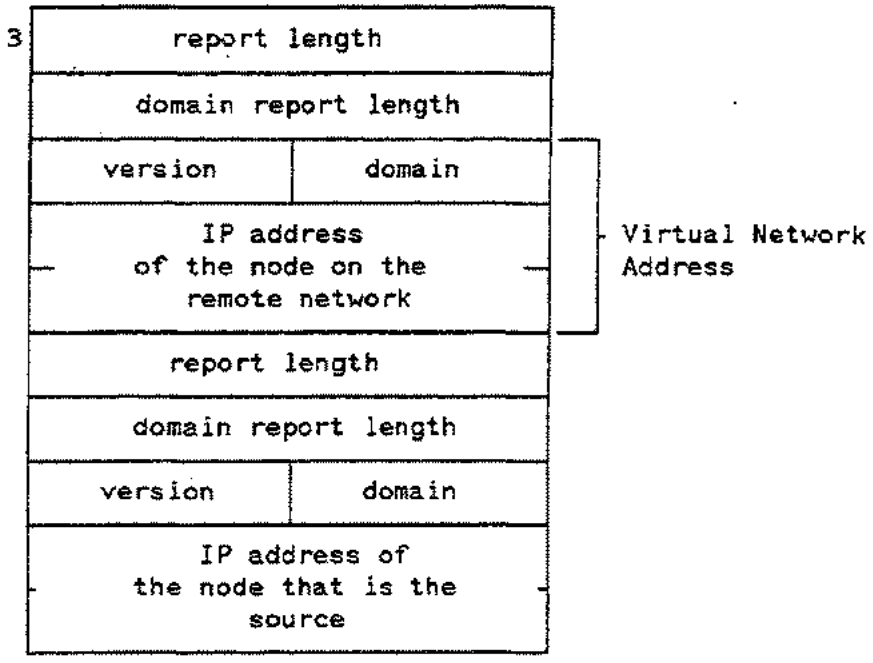
Unsolicited Reply (type 21)



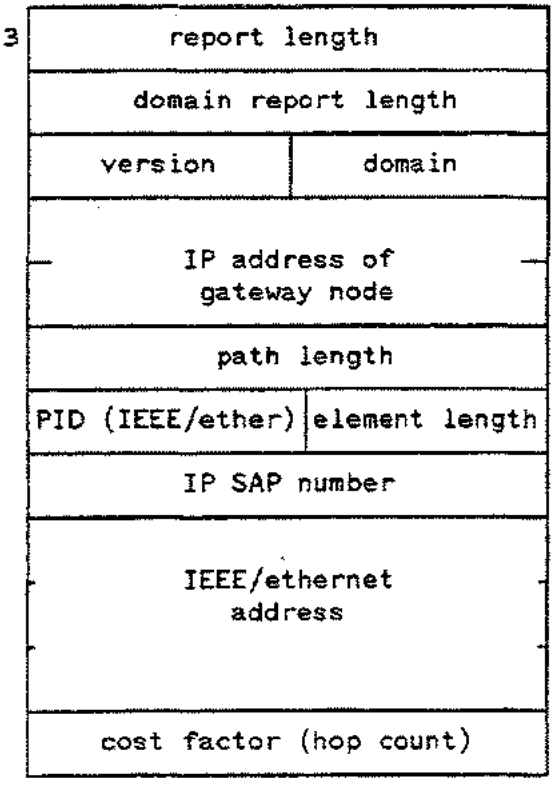
Virtual Address Reply (type 20)

The Virtual Address Reply is not pictured because it consists of only the Common header. Its message length=6.

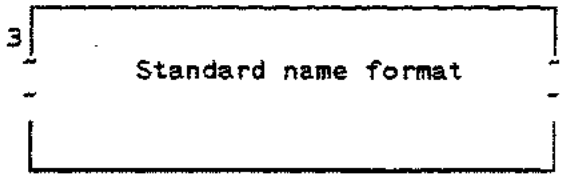
Gateway Request (type 18)



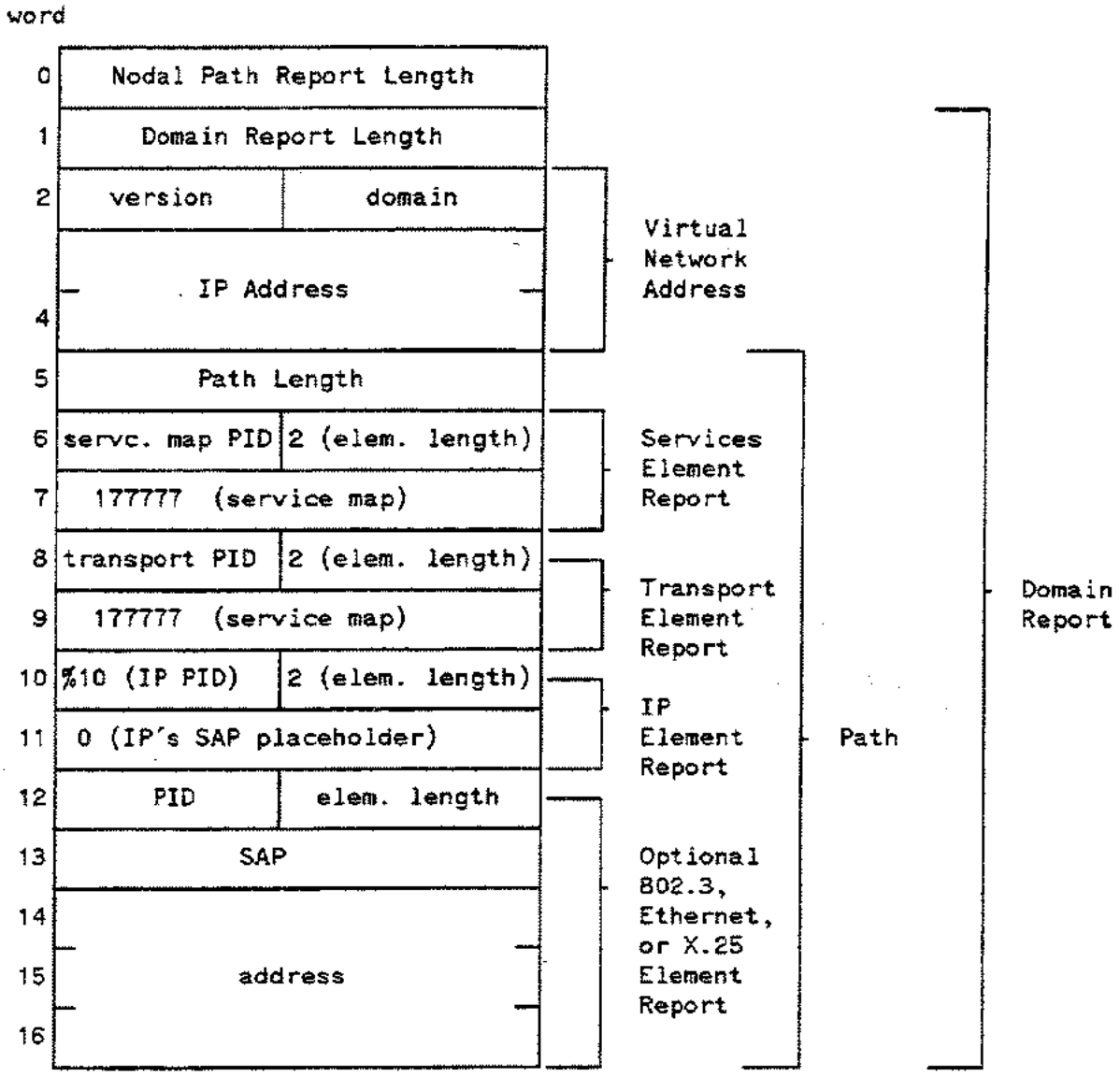
Gateway Reply (type 22)



Node Down (type 25)



Nodal Path Report



A Nodal Path Report (NPR) consists of one or more domain reports. The NPR includes one domain report for each internetwork that the node belongs to.

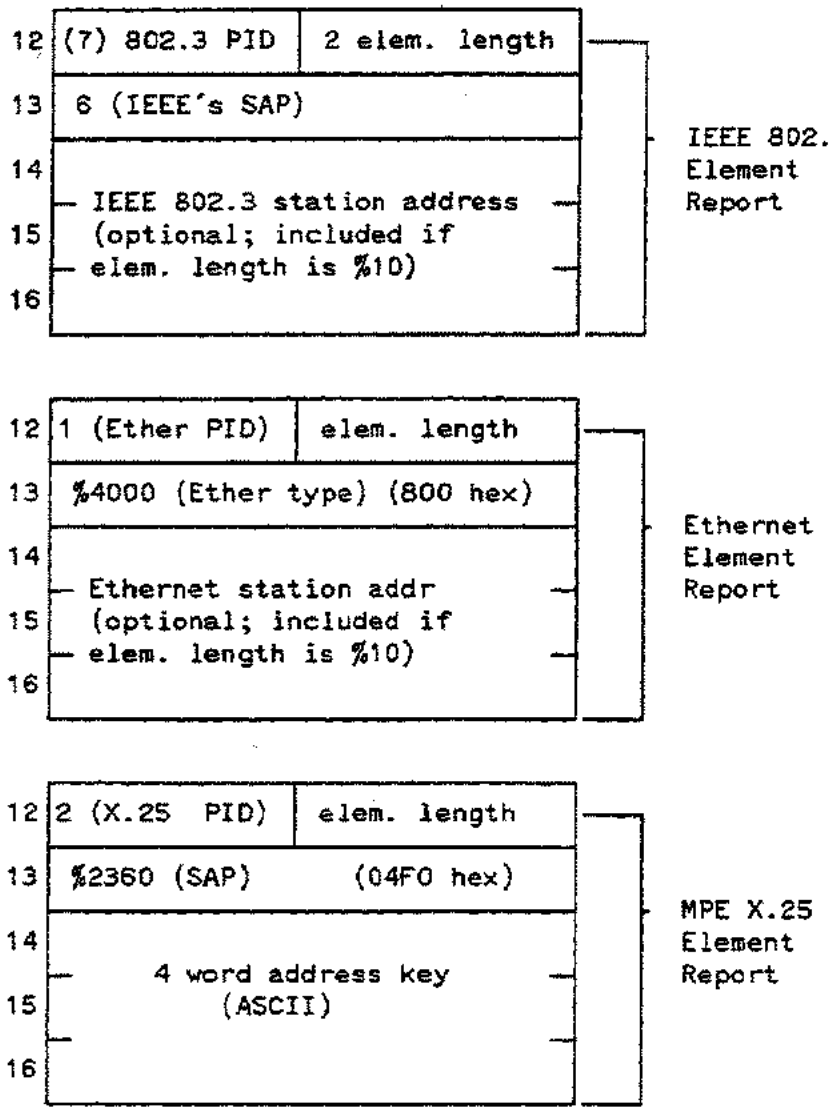
A domain report consists of the following:

- a virtual network address and
- a path report.

A path report consists of

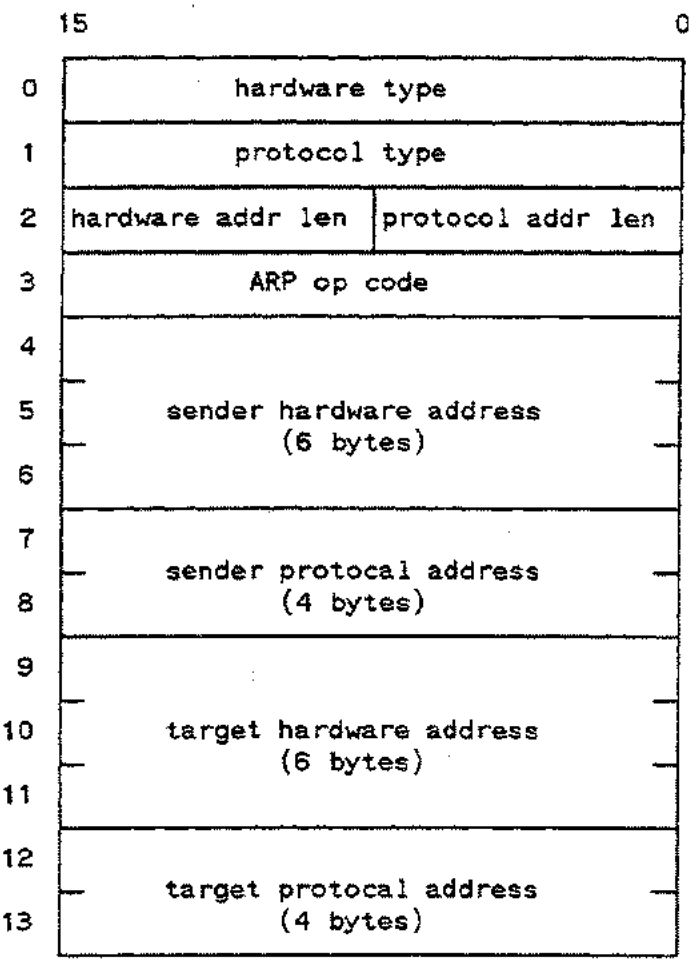
- a service element report,
- a transport element report, and
- an IP element report, and

- (optional) an IEEE 802.3 element report, or an Ethernet, or MPE X.25 element report (see below).



domain	The domain=1, for HPDSN.
elem. length	The length of each element report, in bytes. (This does not include the element length and PID fields.)
servc. map PID	The service map protocol ID, which is 377 octal, indicating NS services.
transport PID	The transport protocol ID, which is 376 octal, indicating NS transport.
IEEE 802.3 PID	The IEEE 802.3 protocol ID, which is 7 octal.
IP PID	The internet protocol ID, which is 10 octal.

ARP Request/Reply Header



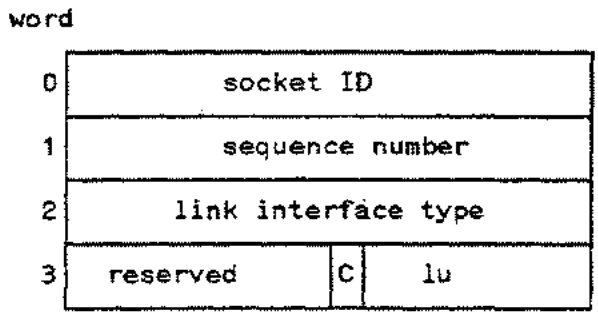
hardware type	specifies the kind of hardware used. (Ethernet = 0001)
protocol type	specifies the upper level protocol type to hand the packet to. (Protocol type IP = 2048)
hardware addr len	specifies the length of the station address, in number of bytes. (Default station address is 6 bytes.)
protocol addr len	specifies the length of the IP or network address, in number of bytes. (Default IP address is 4 bytes.)
ARP op code	specifies if the message is an ARP request or ARP reply. "1" = ARP request. "0" = ARP reply.

sender hardware address	hardware address of the sender of this packet (requester or server hardware address). The length of the address is determined by the "hardware addr len" field. (Default length is 6 bytes.)
sender protocol address	protocol address of the sender of this packet (requester or server hardware address). The length of the address is determined by the "protocol addr len" field. (Default length is 4 bytes.)
target hardware address	hardware address of target node. If the messages is an ARP request, this field will contain 0's. If it is an ARPA reply, it will contain the original requester's station address.
target protocol address	protocol address of target node. If the message is an ARP request, this field will contain the target node's IP address. If the message is an ARP reply, this field will contain the original requester's IP address.

Control Buffer Messages

The messages described in this section appear in the "Control Buffer" section of NS-ARPA/1000 trace files formatted by FMTRC.

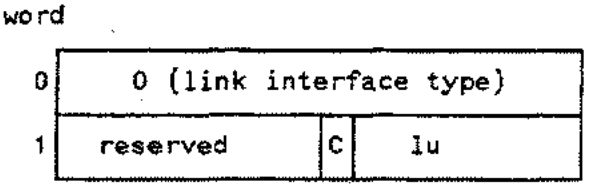
Non-Router Control Buffer Message



The Non-Router Control Buffer Message is appended to messages to be transmitted over non-router (IEEE 802.3 or gateway half) links. The Control Buffer Message itself is not transmitted, however.

socket ID	For outbound messages, this is the identifier of the socket's outbound socket buffer. For inbound messages, this field =0.
sequence number	The trace record's sequence number.
link interface type	The type of link interface is indicated by the following values: 802.3 =7 GWY =6
C (C bit in word 3)	The C bit is 0 if the message was received from a communications link driver. The C bit is 1 if the message was received after being requeued.

Router Control Buffer Message



The Router Control Buffer Message is appended to NS service messages to be transmitted over Router (RTR) links.

link interface type	A value of 0 indicates a Router/1000 link.
C (C bit in word 3)	The C bit is 0 if the message was received from a communications link driver. The C bit is 1 if the message was received after being requeued.

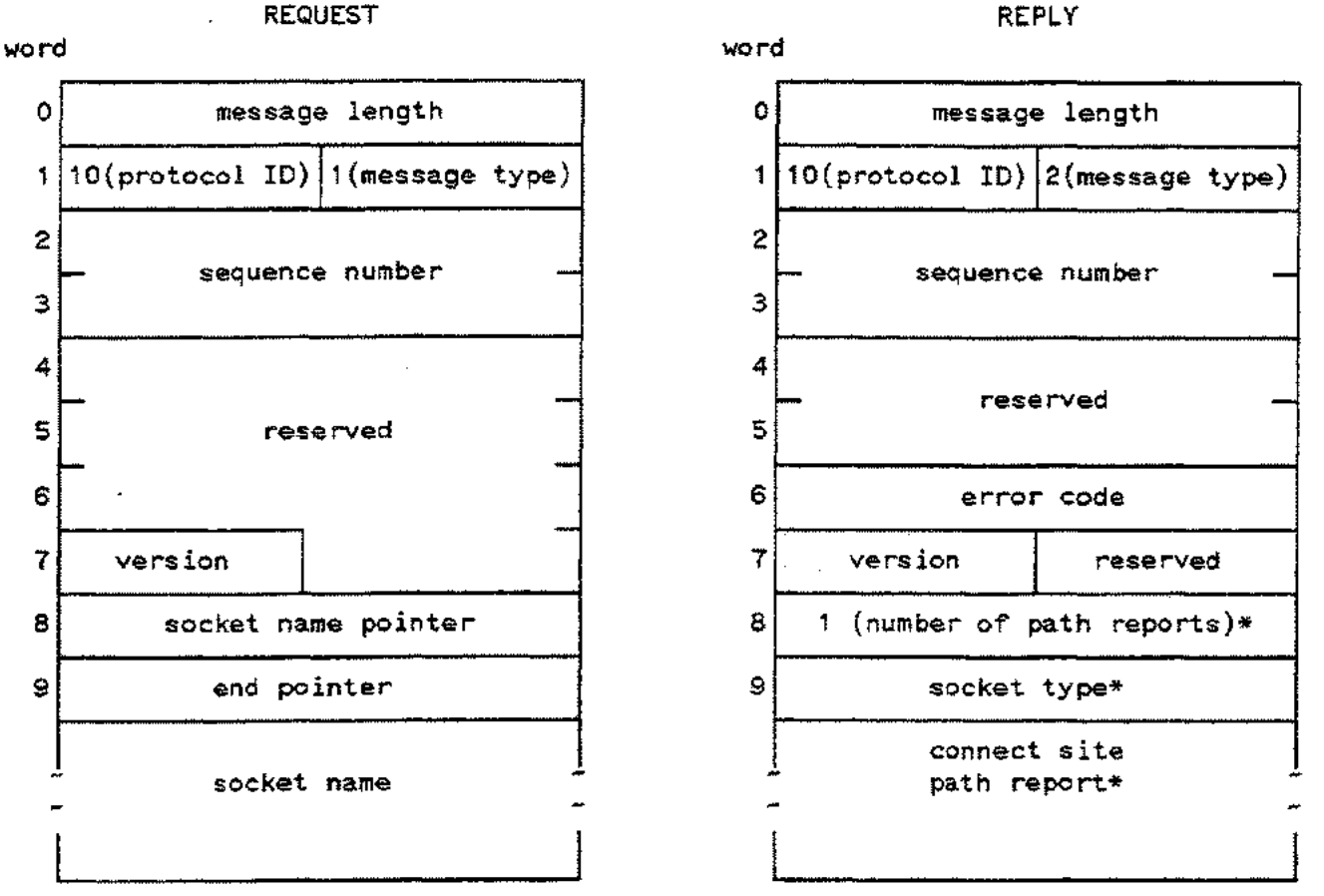
Note: Refer to the "Local Appendage" subsection of Section 2 of this manual for a description of a similar message appended to messages generated by DS/1000-IV services.

SOCKET REGISTRY MESSAGES

SECTION 2

The diagrams in this section illustrate messages generated by Socket Registry software. These messages include the Socket Registry Header and the Socket Registry Connect Site Path Report.

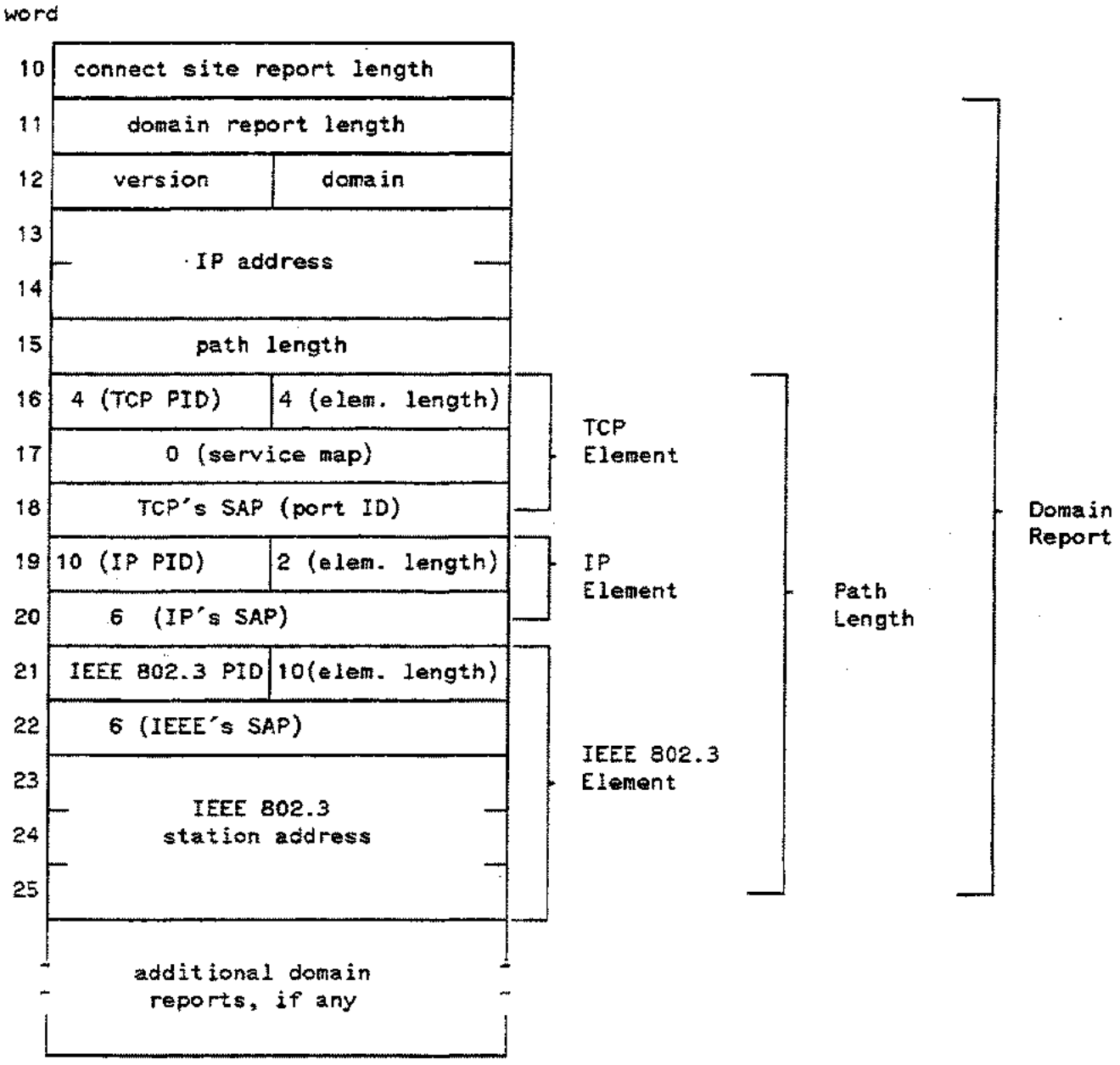
Socket Registry



* Fields marked with an asterisk (*) do not appear if an error is detected (if error code ≠0).

version is the socket registry version number.

Socket Registry Connect Site Path Report



All lengths are in bytes.

PID = Protocol ID

SAP = service access point

domain = 1, for HPDSN

IEEE 802.3 Protocol ID (left byte of word 20) = 7

elem. length (element length) = length of the current element starting from the following word (TCP element, IP element, or IEEE 802.3 element)

The IP SAP (word 22) might not appear in the connect site path report, in which case word 22 will contain a placeholder of 0.

* The IEEE 802.3 Element may or may not be present.

Note: Although the IEEE 802.3 and Internet Protocol (IP) service access points might both be 6, these values are unrelated. The IEEE 802.3 SAP indicates IP, and the IP SAP indicates TCP.

NFT MESSAGES

SECTION 3

This section contains illustrations of NFT messages.

NFT is available on NS-ARPA/1000, NS/3000, NS/9000, NS for the DEC VAX computer systems, and OfficeShare on the PC.

For each NFT transaction there is an **Initiator** node, **Producer** node and **Consumer** node. The Initiator node is the node where the NFT request originates. The Producer node is the the node at which the file to be copied exists. The Consumer node is the node to which the copied file will be sent.

Accordingly, NFT requires the following modules:

- DSCOPY, which is required at the Initiator node. DSCOPY is scheduled at user request.
- PRODC and PRDC1, which are required at the Producer node.
- CONSM, which is required at the Consumer node.
- NFTMN (NFT Monitor), which is required at the Producer and Consumer nodes. NFTMN is scheduled at node initialization by NSINIT in all nodes that support NFT. Each NFTMN has a well-known call socket. The address of these call sockets is the same on all nodes, and is known by all NFT modules.

NOTE

On other HP systems, the modules that perform these tasks may have other names. This subsection refers to the modules according to the HP 1000 module names.

NFT General Flow

This subsection describes the general sequence of events of a file transfer. The message flow is summarized at the end of the subsection, and each NFT message is briefly described in Table 3-1.

1. At the Initiator node, the Initiator Process (DSCOPY) is scheduled by the user.
2. DSCOPY parses the command.
3. If the command is a copy request, DSCOPY attempts to establish an NFT connection with the Producer node. An NFT connection is a connection established between two NFT modules, and is based on a NetIPC connection. DSCOPY calls IPCDest to get a path report for the Producer node's well-known NFTMN call socket. IPCDest is an HP-Internal NetIPC call that NFT calls with NFTMN's well-known call socket address and node name as call parameters. IPCDest returns a path report descriptor to that socket. With this path descriptor, DSCOPY calls

IPCConnect, to initiate a VC connection with NFTMN at the Producer node. NFTMN (Producer node) calls IPCRecvCn, establishing a VC connection. (See Figure 3-1A.)

4. DSCOPY uses IPCSend to send an RINIT (Request Initialization) message to NFTMN (Producer node), to request an NFT connection initialization. The RINIT message also specifies a logon string (or key to an existing session) that the Producer node is to use to produce the source file.

5. NFTMN then attempts to schedule a PRODC clone. Once PRODC is scheduled, NFTMN passes the VC socket descriptor to PRODC, so that PRODC now has a VC socket connected to DSCOPY. With this VC socket descriptor, PRODC uses IPCSend to send an AINIT (Acknowledge Initialization) message to DSCOPY, and the NFT connection between the Initiator (DSCOPY) and the Producer (PRODC) is initialized. (See Figure 3-1B.)

6. DSCOPY sends an RNFT to PRODC. The RNFT message specifies file characteristics, the Consumer node name, the target file, and a logon string (or a key to an existing session) to use to access the target file. PRODC then initiates a VC connection with NFTMN at the Consumer node, using IPCDest and IPCConnect. NFTMN (Consumer node) calls IPCRecvCn, establishing a VC connection to PRODC. (See Figure 3-1C.)

7. PRODC sends an RINIT message to NFTMN (Consumer node). NFTMN (Consumer node) attempts to schedule a CONSM clone. Once CONSM is scheduled, NFTMN passes the VC socket descriptor to CONSM, so that CONSM now has a VC socket connected to PRODC. With this VC socket descriptor, CONSM uses IPCSend to send an AINIT message to PRODC, and the NFT connection between the Producer (PRODC) and the Consumer (CONSM) is initialized. (See Figure 3-1D.)

8. If the user specified a file mask (wildcards in the file name), PRODC schedules the file lister program, PRDC1, passing the file mask in the scheduling parameters. From the file mask, PRDC1 generates a list of files and directories and places them in a scratch file for PRODC to access.

9. PRODC and CONSM negotiate the characteristics of the file to be transferred. If the Producer node and the Consumer node are homogeneous (for example, both HP 1000s), PRODC sends an OFFERT message for transparent mode to CONSM, specifying the source and target file names and file attributes, such as file size.

   If the Producer node and the Consumer node are heterogeneous (for example, an HP 1000 and an HP 3000), PRODC sends an OFFERI message for interchange mode, specifying the source and target file names, data type, record type, file type, record length, record file length (the copy descriptor *fsize* parameter). For information on interchange file formats, refer to *NS-ARPA/1000 User/Programmer Reference Manual*.

10. PRODC sends the results of the file negotiation to DSCOPY in an INFO message. DSCOPY prints the results to the list device.

11. PRODC opens the source file and sends it to CONSM in DATA messages. PRODC continues to send DATA messages until it reaches the end of the source file. PRODC then sends an EOD message. Note that PRODC does not wait for acknowledgements to DATA messages. CONSM only sends ADATA (Acknowledge Data) messages if CONSM encounters an error while trying to store the transferred file.

12. CONSM acknowledges the EOD with an AEOD. PRODC closes the source file.

13. PRODC sends the result of the transfer to DSCOPY. If the user specified a file mask, PRODC reads the name of the next source file from PRDC1's scratch file, and returns to step 9. Otherwise, PRODC and CONSM wait for NFT messages on their VC sockets.

14. If the user enters another copy descriptor that can use the same NFT connections (same source and target nodes, same logon IDs), DSCOPY returns to step 8.

15. If the user specified a different Producer node or logon ID, DSCOPY closes the VC connection to PRODC. When PRODC notes that DSCOPY has closed its connection, PRODC closes the VC connection to CONSM and terminates. CONSM notes that PRODC has closed the connection and terminates. Execution continues from step 3.

16. If the user specified a different Consumer node or logon ID, but is using the same Producer node and logon ID, DSCOPY sends an RNFT to PRODC, PRODC closes the VC connection to CONSM. CONSM notes that PRODC has closed the connection and terminates. Execution continues from step 6.

17. If the user wants to quit, DSCOPY closes the VC connection to PRODC and terminates. When PRODC notes that DSCOPY closed its connection, PRODC closes the VC connection to CONSM and terminates. CONSM notes that PRODC has closed the connection and terminates.

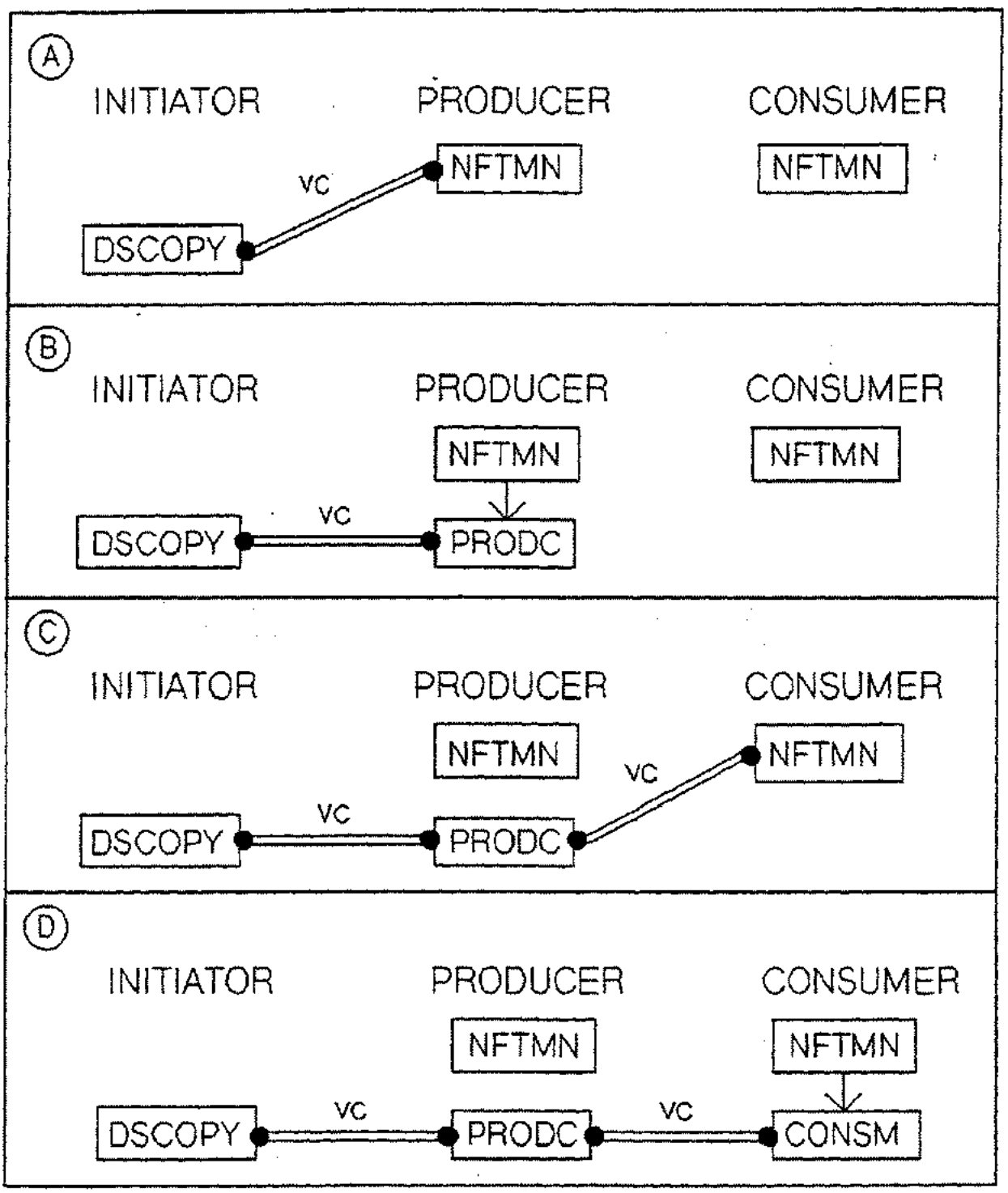


Figure 3-1. NFT Connection Establishment.

Message Flow Summary

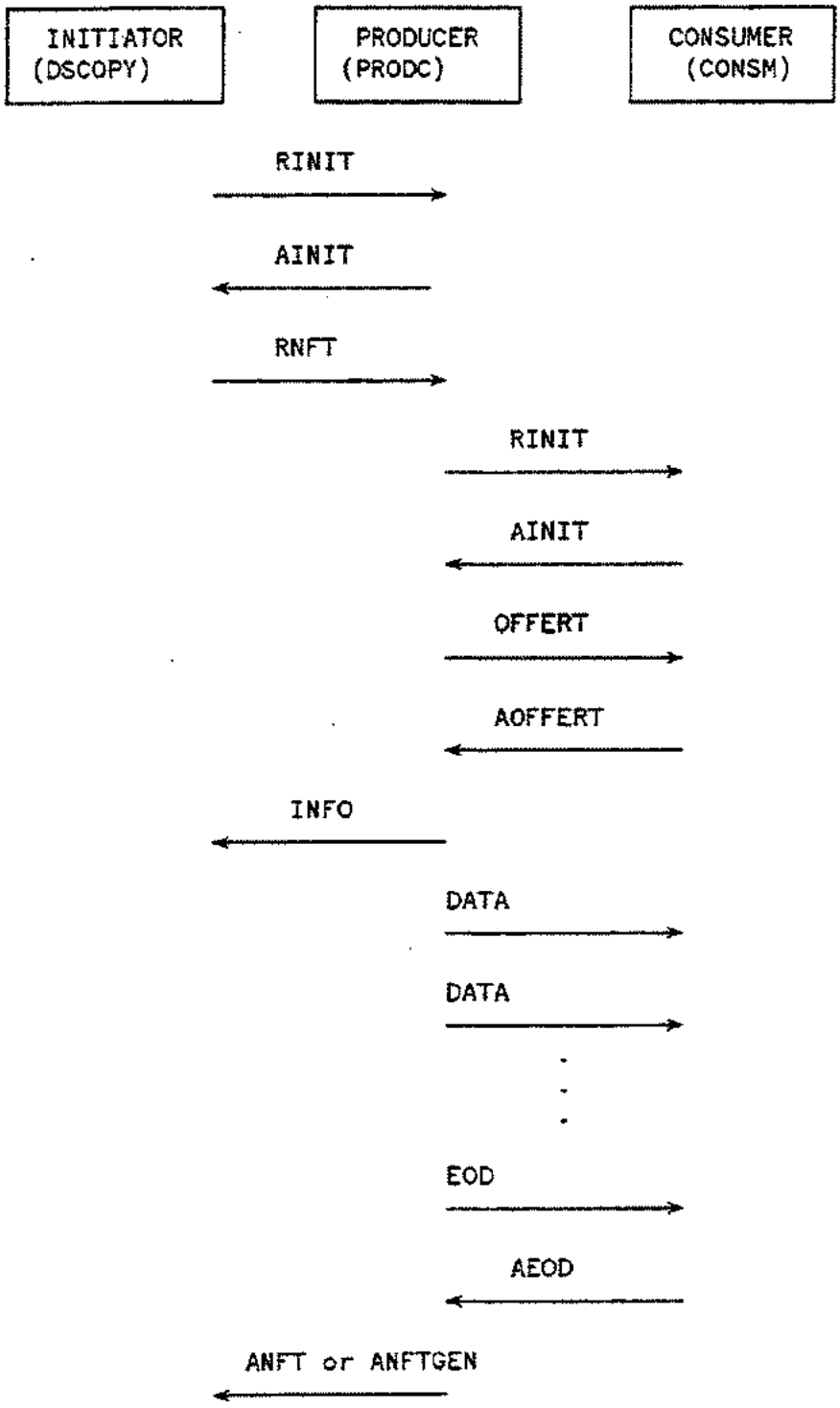


TABLE 3-1. NFT MESSAGES.

Message	Purpose
RINIT	Request Initialization. Request to initialize NFT connection. Sent by DSCOPY (Initiator) to PRODC and by PRODC to CONSM.
AINIT	Acknowledge Initialization (RINIT). Sent back by PRODC or CONSM.
RNFT	Request Network File Transfer. Sent by DSCOPY (Initiator) to ask PRODC to begin a file transfer to CONSM.
ANFT	Acknowledge Network File Transfer (RNFT). Signal end of successful file transfer or to report an error. Sent by PRODC to DSCOPY (Initiator) when a file transfer is complete, or when the last file/directory of a group transfer is complete, or when PRODC has encountered an error.
ANFTGEN	Acknowledge Network File Transfer (generic). Sent by PRODC to DSCOPY (Initiator) when a single file transfer of a group transfer is complete (except for the last file transfer of a group; PRODC sends an ANFT in that case). Also sent by PRODC to DSCOPY to indicate that PRODC has encountered an error.
OFFERT	Offer Transparent. Sent by PRODC to CONSM; makes an offer for transparent format file transfer.
AOFFERT	Acknowledge Offer Transparent (OFFERT). Sent by CONSM to PRODC; acknowledges transparent format file transfer offer and returns an error if offer was unacceptable.
OFFERI	Offer Interchange. Sent by PRODC to CONSM; makes an offer for interchange format file transfer. Proposes file format for interchange.
AOFFERI	Accept Offer Interchange. Sent by CONSM to PRODC; acknowledges interchange format file transfer offer. It also proposes a counter-offer or returns error if offer was unacceptable.
DIRECTORY	Create Target Directory. Sent by PRODC to CONSM; requests creation of a new directory and gives characteristics for that directory. (Only valid when CONSM supports a hierarchical file system.)
ADIRECTORY	Acknowledge Directory Creation. Sent by CONSM to PRODC; reports result of directory creation requested by DIRECTORY message.

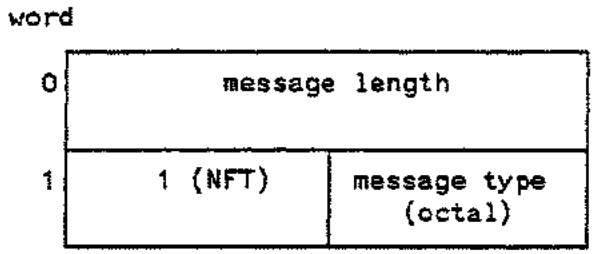
TABLE 3-1. NFT MESSAGES (Continued).

RPROGRESS	Request Progress. Sent by DSCOPY (Initiator) to PRODC; requests progress report on present file transfer.
PROGRESS	Progress. Sent by PRODC to DSCOPY (Initiator) to respond to RPROGRESS. Contains status of current file transfer.
INFO	Information. Sent by PRODC to DSCOPY (Initiator) after file transfer negotiations. Includes source and target file names.
ABORT	Abort Transfer. Sent by DSCOPY (Initiator) to PRODC; requests that PRODC abort the current file transfer to CONSM. Also sent by PRODC to CONSM; notifies CONSM that PRODC encountered an error, and therefore it is prematurely terminating current file transfer.
CANCEL	Cancel Transfer. Same as ABORT, but any files created during the current transfer are purged (the new target file, any scratch files).
AABORT	Acknowledge Abort. Sent by CONSM to acknowledge ABORT and CANCEL messages sent by PRODC. (However, PRODC does not send AABORT messages to DSCOPY to acknowledge ABORT and CANCEL messages sent by DSCOPY. In those cases, PRODC will send an ANFT.)
DATA	File Data. Sent by PRODC to CONSM. Contains file data.
ADATA	Acknowledge File Data. Sent by CONSM to PRODC; signifies that CONSM encountered an error while trying to store file data.
EOD	End of Data. Sent by PRODC to CONSM; indicates the end of the source file. May or may not include file data.
AEOD	Acknowledge End of Data. Sent by CONSM to PRODC; acknowledges End of Data after CONSM has successfully closed target file.

NFT Header

All NFT messages begin with a header that has the following format:

NFT HEADER



Each message begins with the message length (word 0). All message lengths in this section are in bytes. As shown, word 1 of this header consists of two fields. The left byte, which contains 1, identifies the message as an NFT message. The right byte, which contains the message type, identifies the specific kind of NFT message that is being sent. Together, these fields contain octal values ranging from 401 to 437. These values are what the diagrams in this section show as the contents of word 1.

NFT Errors

Most NFT messages contain an NFT error code field. The error codes in this field correspond to NFT errors described in the *NS-ARPA/1000 Error Message and Recovery Manual* and the *NS/3000 Error Message and Recovery Manual*. Some messages contain an error code enhancement parameter, which may contain error qualification, such as a file name or other explanation.

Pointers

Many NFT messages described in this section contain pointers to message parameters. For example, the ADATA message contains a local error message pointer, an NFT enhancement pointer, and an end pointer.

Fields labeled as pointers contain the byte offset of the first byte of the parameter from the beginning of the message. Using the ADATA message as an example (refer to the ADATA message later in this section), word 7 is the local error message pointer. Because the local error message is the first parameter contained in the variable length parameter area, the local error message pointer will point to the first byte of the parameter area. In the case of ADATA, the local error message pointer will contain 22 octal (18 decimal) to indicate that the local message parameter begins with the 19th byte from the beginning of the message. (The first byte of the message has an offset of 0).

The parameter indexed by each subsequent pointer in the message will be appended to the parameter indexed by the previous pointer. For example, for ADATA, the NFT error code enhancement field will occur in the message immediately after the local error message. The local error message pointer will contain the offset of the local error message field from the beginning of the message.

Each message that contains one or more parameter fields also contains an end pointer. The end pointer is the byte offset of the first byte following the message; therefore, the `end pointer` value will be the same value as the `message length`.

NFT MESSAGE FORMATS

Table 3-2 summarizes NFT messages. The diagrams of each message appear in this chapter in ascending numerical order according to message type, as listed in the table.

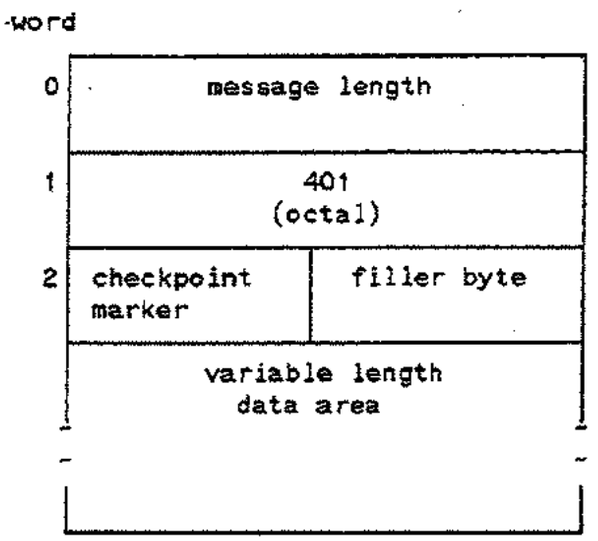
Table 3-2. NFT Message Types

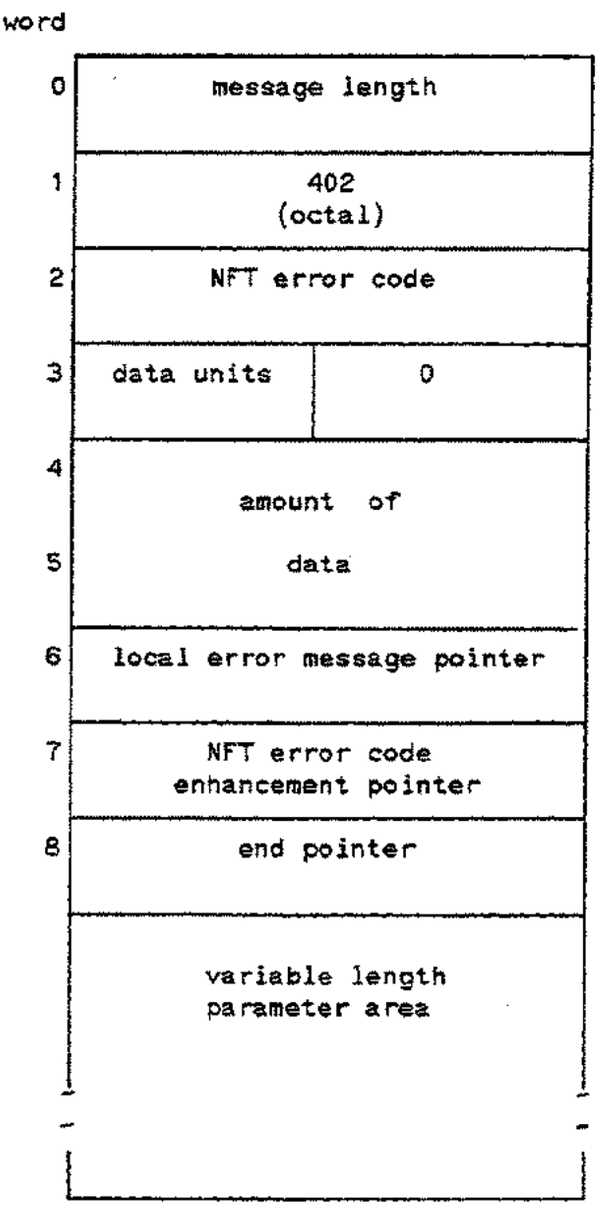
Word 1 octal	(hex)	Message Type octal	(hex)	Message - Meaning
401	(101)	1	(1)	DATA - file data
402	(102)	2	(2)	ADATA - acknowledge file data (error)
403	(103)	3	(3)	MARKER - marker (data checkpoint)
404	(104)	4	(4)	AMARKER - acknowledge marker
405	(105)	5	(5)	EOD - end of data
406	(106)	6	(6)	AEOD - acknowledge end of data
407	(107)	7	(7)	COMPDATA - compressed data
410	(108)	10	(8)	COMPEOD - compression end of data
411	(109)	11	(9)	COMPMARKER - compressed data with marker (data checkpoint)
413	(10B)	13	(B)	RNFT - request NFT
414	(10C)	14	(C)	ANFT - acknowledge NFT
415	(10D)	15	(D)	RNFTR - request NFT restart
416	(10E)	16	(E)	ANFTGEN - acknowledge NFT generic
417	(10F)	17	(F)	RPROGRESS - request progress
420	(110)	20	(10)	PROGRESS - progress
421	(111)	21	(11)	ABORT - abort transfer
422	(112)	22	(12)	INFO - information

Table 3-2. NFT Message Types (cont'd)

Word 1 octal	(hex)	Message Type octal	(hex)	Message - Meaning
423	(113)	23	(13)	CANCEL - cancel transfer
424	(114)	24	(14)	AABORT - acknowledge abort
425	(115)	25	(15)	OFFERT - offer transparent
426	(116)	26	(16)	AOFFERT - acknowledge offer transparent format
427	(117)	27	(17)	OFFERI - offer interchange format
430	(118)	30	(18)	AOFFERI - acknowledge offer interchange format
431	(119)	31	(19)	RINIT - request initialization
432	(11A)	32	(1A)	AINIT - acknowledge initialization
433	(11B)	33	(1B)	OFFERR - offer restart
434	(11C)	34	(1C)	AOFFERR - acknowledge offer restart
435	(11D)	35	(1D)	DIRECTORY - create target directory
436	(11E)	36	(1E)	ADIRECTORY - acknowledge directory creation
437	(11F)	37	(1F)	WARN - warning

DATA/ADATA





NFT error code — 0 = successful transfer; refer to NFT error numbers for other values

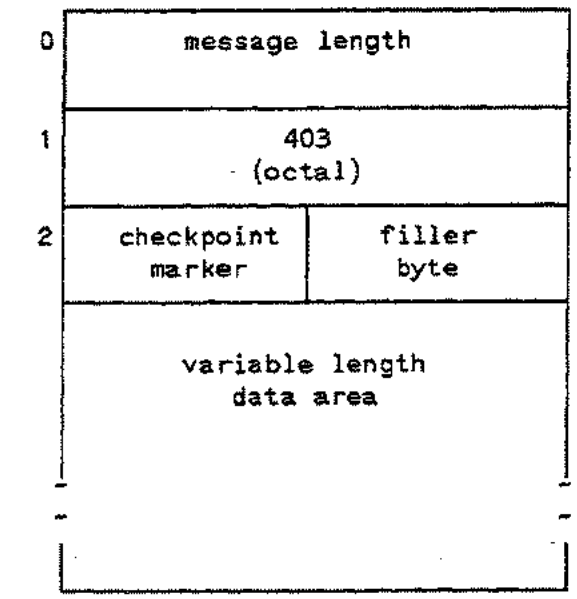
checkpoint marker — word 2 is checkpoint marker if both systems support checkpoint/restart. If not, data starts at word 2.

checkpoint marker identifies the current checkpoint record of the system that originates this parameter

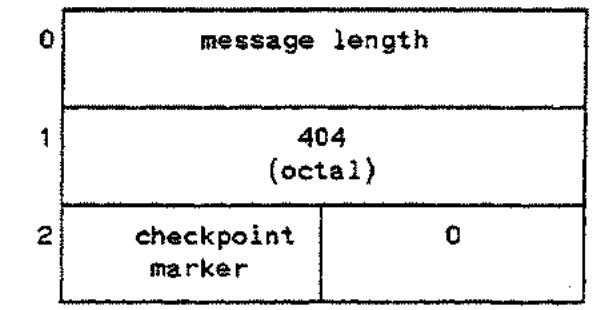
filler byte	extra byte added as filler so that packet field will start on word boundary
data units	units for "amount of data" parameter: 0 = N/A 1 = bytes 2 = 16 bit words 3 = 32 bit words 4 = logical records 5 = physical records (blocks) 6 = percentage of file

MARKER/AMARKER

MARKER



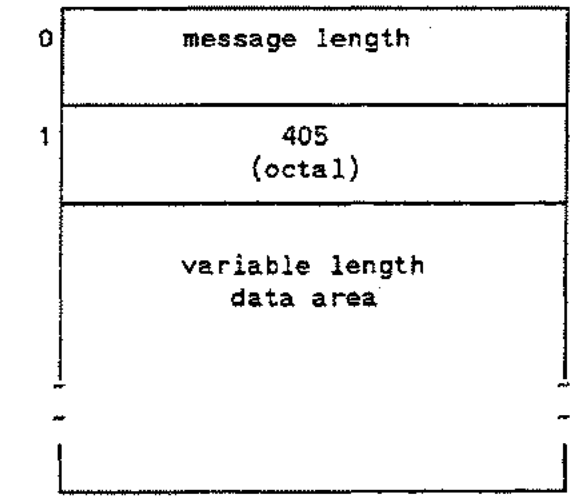
AMARKER



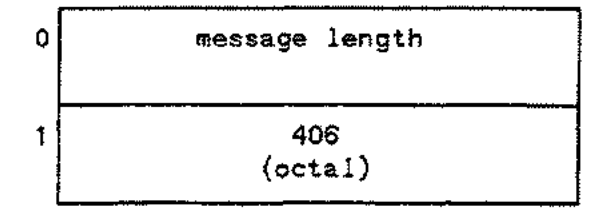
checkpoint marker	identifies the current checkpoint record of the system that originates this parameter
filler byte	extra byte added as filler so that packet field will start on word boundary

EOD/AEOD

EOD

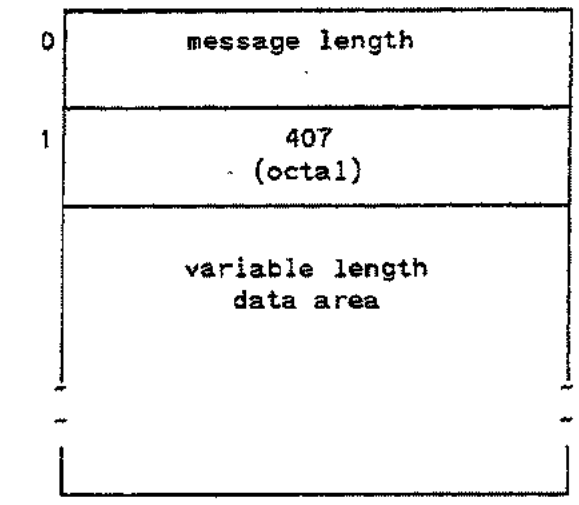


AEOD

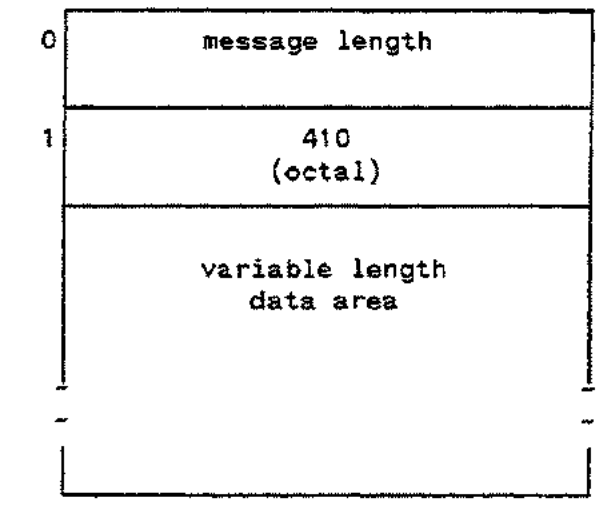


COMPDATA/COMPEOD

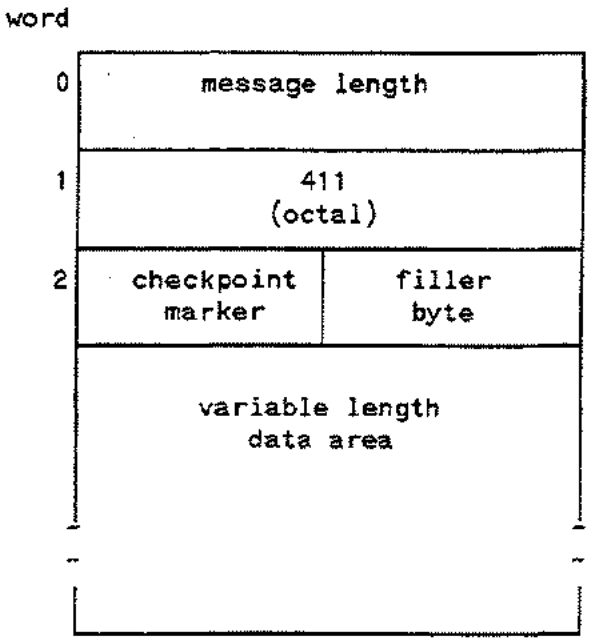
COMPDATA



COMPEOD

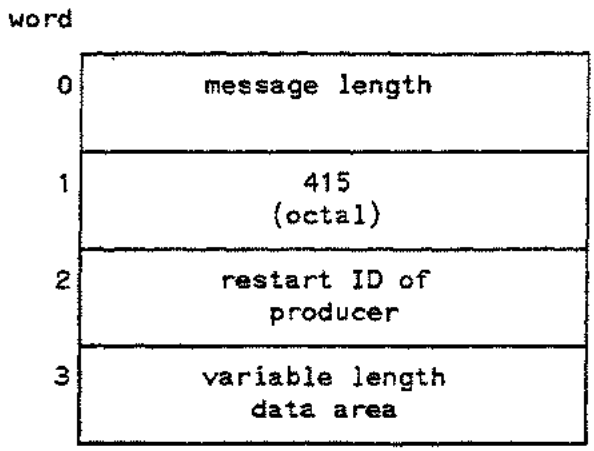


COMPMARKER



checkpoint marker	identifies the current checkpoint record of the system that originates this parameter
filler byte	extra byte added as filler so that packet field will start on word boundary

RNFTR



restart ID of producer	specifies the restart identifier of the current file transfer on the system that originates the message that contains this parameter. Zero indicates no checkpointing will be done.
checkpoint interval	indicates the period of time in seconds between the sending of checkpoint markers. Zero indicates that the checkpointing facility should not be used.

RNFT/ANFT

RNFT

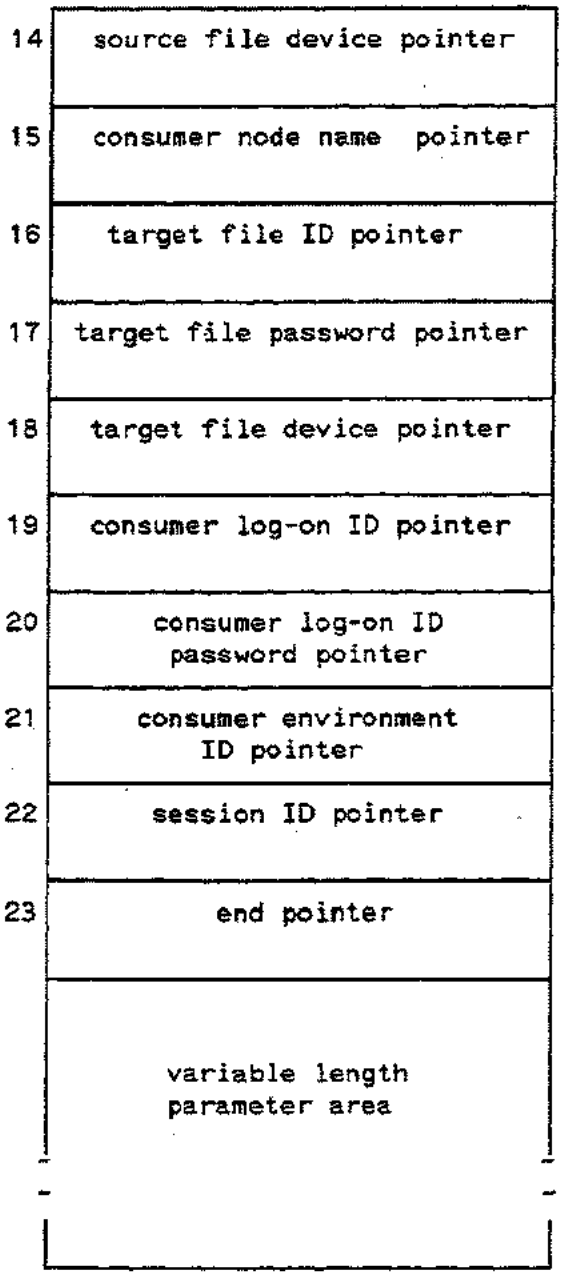
word	
0	message length
1	413 (octal)
2	force store options / data type
3	record type / file type
4–5	record length
6–7	record file length
8	consumer storage / producer options
9	checkpoint interval
10	search character pointer
11	insertion character pointer
12	source file ID pointer
13	source file password pointer

(continued on next page)

ANFT

word	
0	message length
1	414 (octal)
2–3	amount of data
4	data units / misc ANFT flags
5	NFT error code
6	local error message pointer
7	NFT error code enhancement pointer
8	source file ID pointer
9	target file ID pointer
10	end pointer
	variable length parameter area

(RNFT, continued)



force store options

0 = use default transfer
1 = force interchange transfer
2 = force transparent transfer and transient storage on consumer node

data type

type of pad character to use if necessary:

0 = data type of source
1 = ASCII (space)
2 = Binary (null)

record type

for interchange transfer:

0 = record type of source
1 = fixed length records
2 = variable length records

file type

for interchange transfer:

0 = file type of source
1 = sequential
2 = direct (or random)

record length

number of bytes or 0 if record length of source

record file length

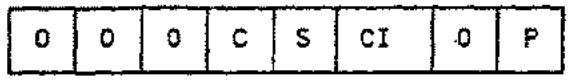
number of records or 0 if file length of source

consumer storage

0	0	0	CC	T	A	O	P

P (if set) = purge existing file
O (if set) = overwrite existing file
A (if set) = append to existing file
T (if set) = force transient storage
CC (if set) = carriage control in first byte of source

producer options



P (if set) = purge source
CI (if set) = consumer node is the same as initiator node
S (if set) = strip trailing blanks or zeros
C (if set) = compress file data

checkpoint internal

indicates the period of time in seconds between the sending of checkpoint and markers. Zero indicates that the checkpointing facility should not be used.

search character pointer

points to the beginning of a character array defining character(s) that have been inserted at the end of each logical record stored on an HP-UX system.

insertion character pointer

points to the beginning of a character array defining character(s) to be inserted at the end of each logical record stored on an HP-UX system.

session ID pointer

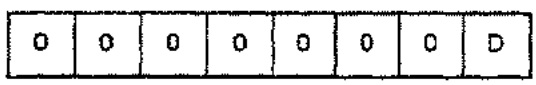
points to an internal number which identifies a previously-created environment to be used by NFT

data units

units for "amount of data" parameter:

0 = N/A
1 = bytes
2 = 16 bit words
3 = 32 bit words
4 = logical records
5 = physical records (blocks)
6 = percentage of file

misc ANFT flags

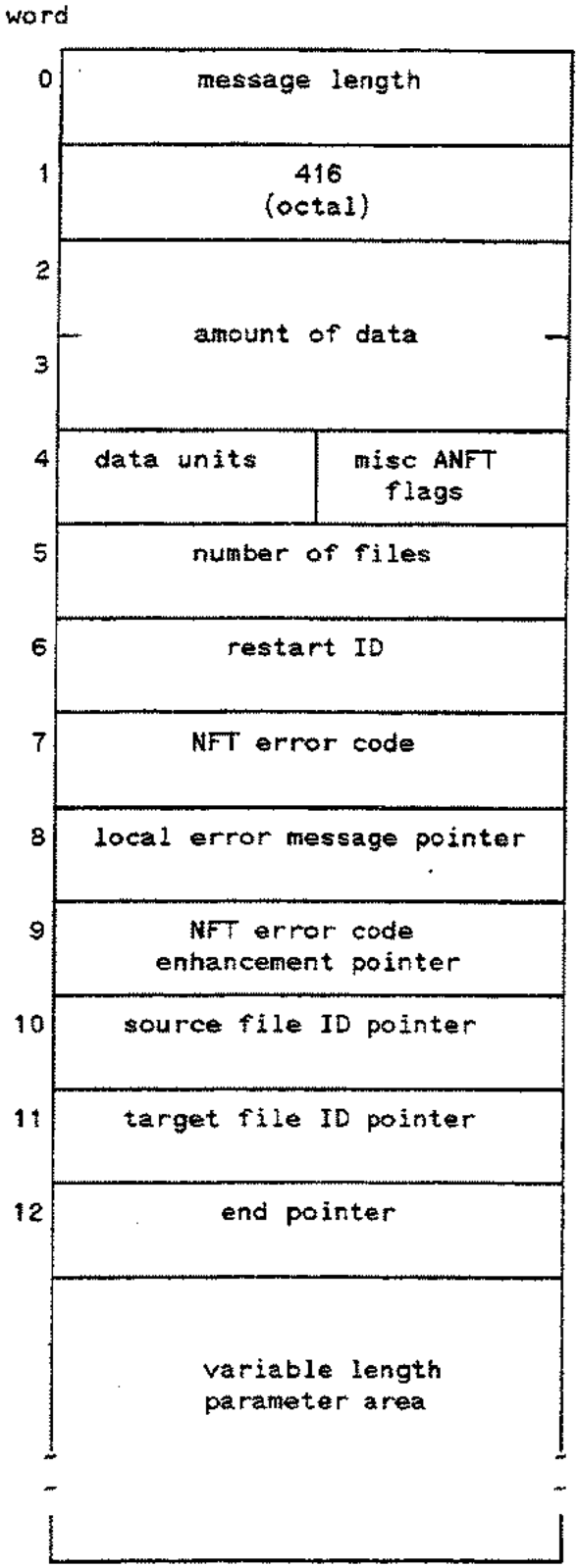


D (if set) = source and/or target strings in this message identify a directory

NFT error code

0 = successful transfer; refer to NFT error numbers for other values

ANFTGEN


word
0
message length
1
416
(octal)
2
amount of data
3
4
data units
misc ANFT
flags
5
number of files
6
restart ID
7
NFT error code
8
local error message pointer
9
NFT error code
enhancement pointer
10
source file ID pointer
11
target file ID pointer
12
end pointer
variable length
parameter area

amount of data

total number of data units that have been transferred between the producer and the consumer nodes

data units

units for "amount of data" parameter:

0 = N/A
1 = bytes
2 = 16 bit words
3 = 32 bit words
4 = logical records
5 = physical records (blocks)
6 = percentage of file

misc ANFT flags

0	0	0	0	0	0	0	D

D(if set) = source and/or target strings in this message identify a directory

number of files

number of files in a generic transfer; filled in only for the first file in the set. This field contains 0 for subsequent transfers.

restart ID

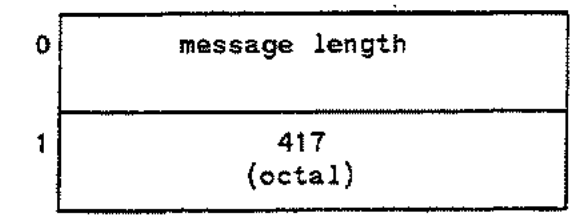
specifies the restart identifier of the current file transfer on the system that originates the message which contains this parameters. Zero indicates no checkpoint will be done.

NFT error code

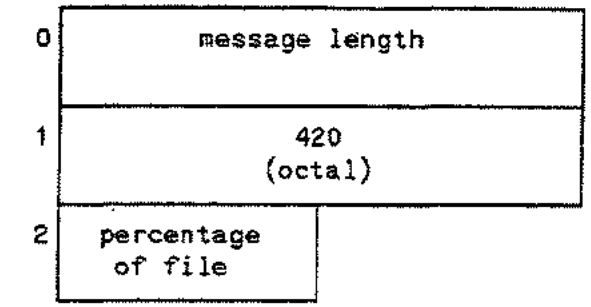
0 = successful transfer; refer to NFT error numbers for other values

RPROGRESS/PROGRESS

RPROGRESS



PROGRESS

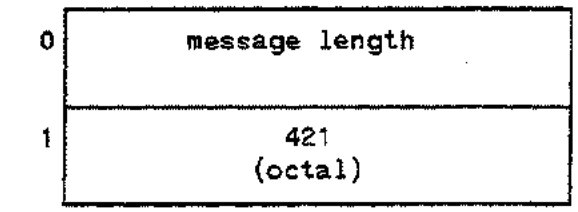


percentage of file

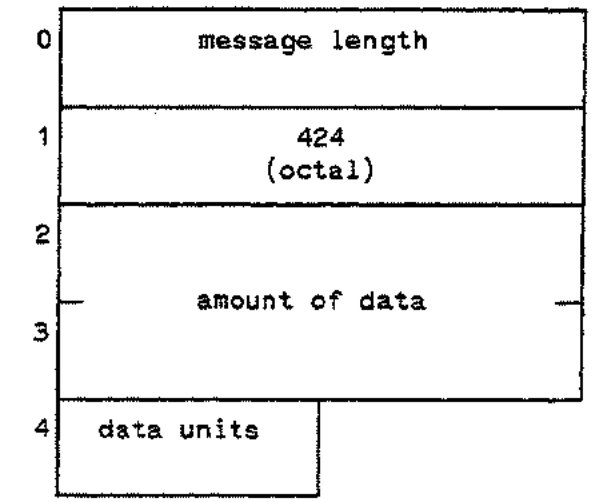
percentage of file transferred

ABORT/AABORT

ABORT



AABORT

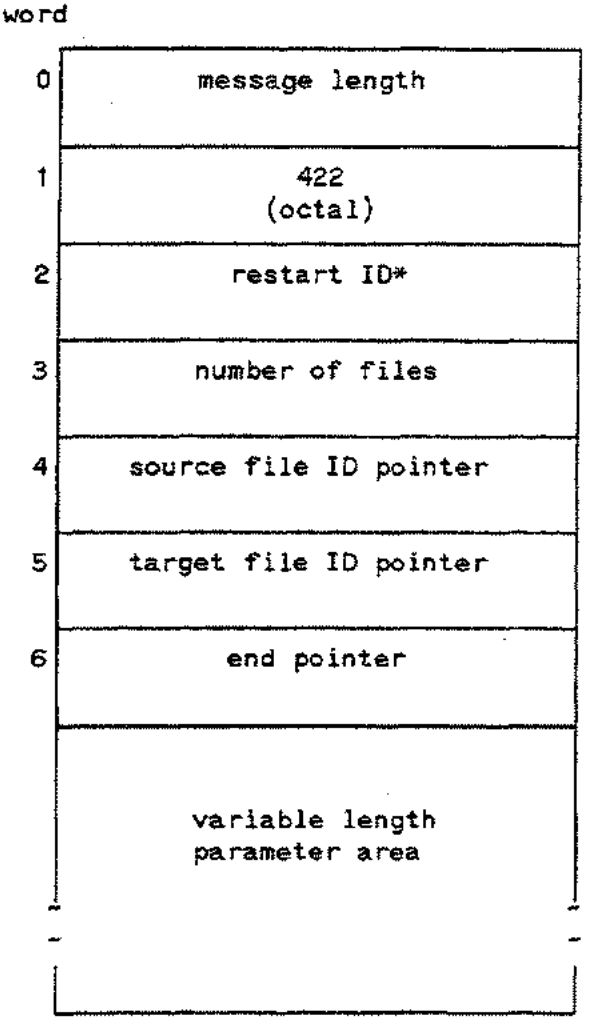


amount of data — number of data units transferred between producer and consumer nodes

data units — units for "amount of data" parameter:

0 = N/A
1 = bytes
2 = 16 bit words
3 = 32 bit words
4 = logical records
5 = physical records (blocks)
6 = percentage of file

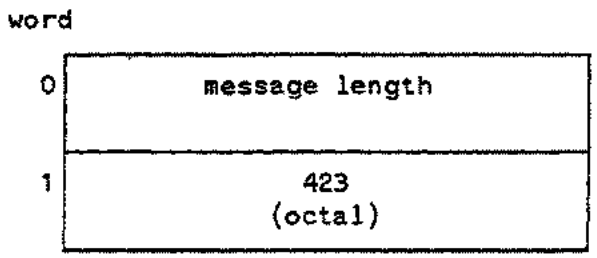
INFO



number of files — number of files in a generic transfer; filled in only for the first file in the set. Field contains 0 for subsequent transfers.

* = not currently used

CANCEL



OFFERT/AOFFERT

OFFERT

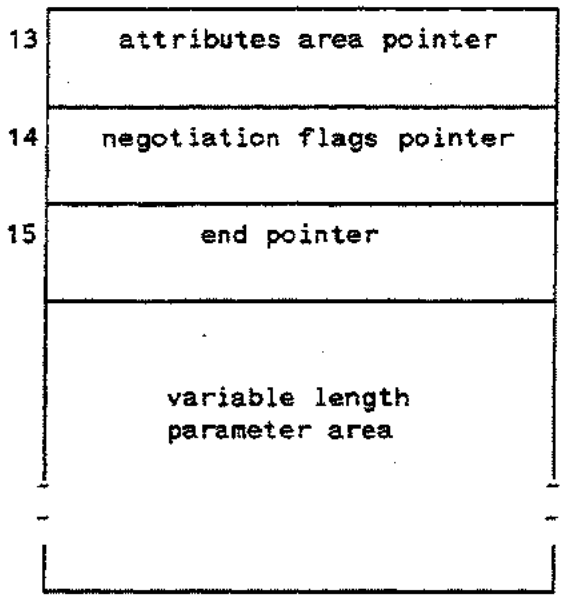
word	
0	message length
1	425 (octal)
2 – 3	file length
4	home sys type
5 – 6	home capability mask
7	consumer storage | num buffers
8	restart ID
9	source file name pointer
10	target file ID pointer
11	target file password pointer
12	target file device pointer

(continued on next page)

AOFFERT

word	
0	message length
1	426 (octal)
2	restart ID
3	NFT error code
4	target file ID pointer
5	local error message pointer
6	NFT error code enhancement pointer
7	system specific message pointer
8	negotiation flags pointer
9	end pointer
	variable length parameter area

(OFFERT, continued)

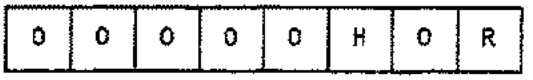


file length — maximum number of bytes in source file (for transient storage only)

home sys type

1 = MPE
12 (octal) = RTE
24 (octal) = HP-UX and VAX/VMS Systems **
36 (octal) = MS-DOS
50 (octal) = other systems

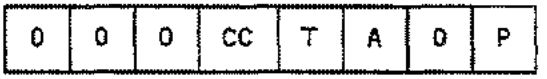
home capability mask — last byte:



R (if set) = may change roles from transfer to transfer (for example, initiator-producer can be used as producer-consumer).

*H (if set) = sending system has hierarchical file system that supports multilevel directories.

consumer storage



P (if set) = purge existing file
O (if set) = overwrite existing file
A (if set) = append to existing file
*T (if set) = force transient storage
*CC (if set) = carriage control in first byte of source

attributes area pointer — points to a system-specific attributes area.

negotiation flags pointer — points to an array of bytes
last byte:

C (if set) = compression is desired or allowed

NFT error code — 0 = successful transfer; refer to NFT error numbers for other values

restart ID — specifies the restart identifier of the current file transfer on the mainframe that originates the message which contains this parameters. Zero indicates no checkpointing will be done.

* = not currently used

** VAX/VMS is a U.S. registered trademark of Digital Equipment Corporation.

OFFERI/AOFFERI

OFFERI

word		
0	message length	
1	427 (octal)	
2	consumer storage	data type
3	record type	file type
4–5	record length	
6–7	record file length	
8	misc interchange flags	number of buffers
9	restart ID	
10	insertion character pointer	
11	source file name pointer	
12	target file ID pointer	

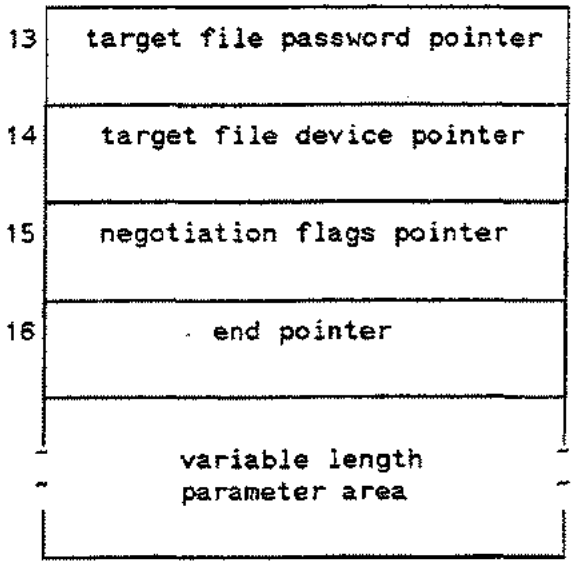
(continued on next page)

AOFFERI

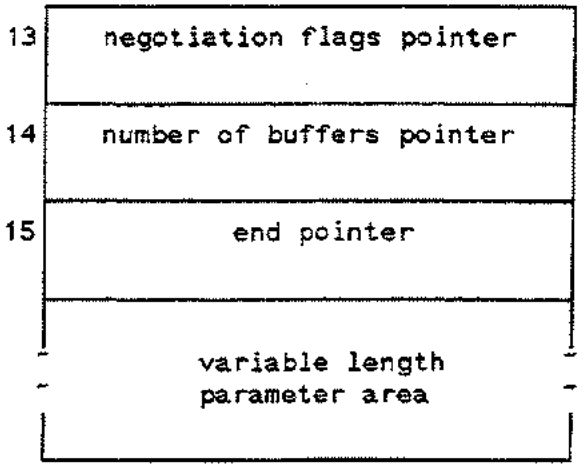
word		
0	message length	
1	430 (octal)	
2	data type	record type
3	file type	filler byte
4–5	record length	
6–7	record file length	
8	restart ID	
9	error code	
10	target file ID pointer	
11	local error message pointer	
12	error code enhancement pointer	

(continued on next page)

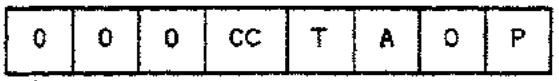
(OFFERI, continued)



(AOFFERI, continued)



consumer storage



P (if set) = purge existing file
O (if set) = overwrite existing file
*A (if set) = append to existing file
*T (if set) = force transient storage
*CC (if set) = carriage control in first byte of source

data type — indicates whether data is ASCII or binary. If data type=1, data is ASCII. If data type=2, data is binary.

record type — indicates type of data records sent to consumer node during an Interchange Format transfer. If record type=1, records are fixed length. If record type=2, records are variable length.

file type — indicates whether file organization is sequential or direct. Sequentially written records=1, directly written records=2.

record length — length of data records, in bytes, for fixed length records. Maximum record length for variable length records.

record file length — target file length, in records.

misc interchange flags

R (if set) = record length value is forced.
F (if set) = file length value is forced.
E (if set) = target file length given in record file length field is an estimate only.

number of buffers — number of outstanding packets permitted

restart ID — specifies the restart identifier of the current file transfer on the system that originates the message which contains this parameter. Zero indicates no checkpointing will be done.

insertion character pointer — points to the beginning of a character array defining character(s) to be inserted at the end of each logical record stored on an HP-UX file system.

RINIT/AINIT

RINIT

word	
0	message length
1	431 (octal)
2	debug flags* | misc flags
3	system type
4	op system version
5	buffer size
6	capability mask
7	
8	system specific capability mask
9	log-on ID pointer
10	log-on password pointer
11	environment ID pointer
12	saession ID pointer

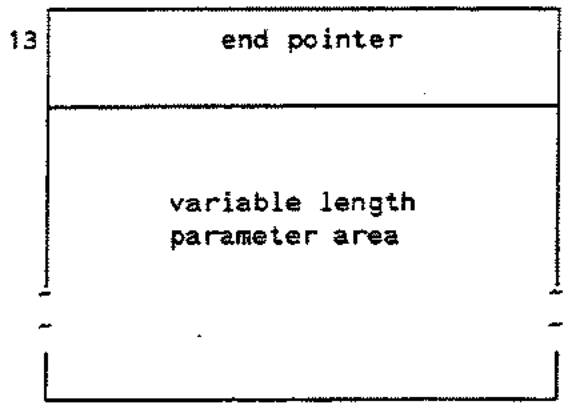
(continued on next page)

AINIT

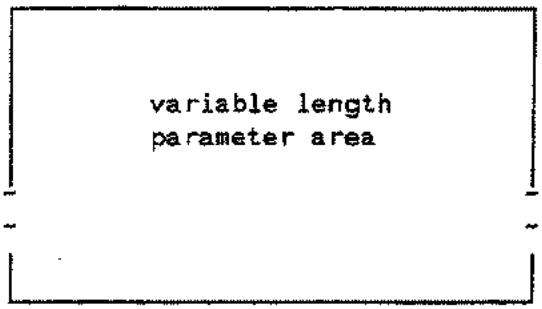
word	
0	message length
1	432 (octal)
2	debug flags* | misc flags
3	system type
4	op system version
5	buffer size
6	capability mask
7	
8	system specific capability mask
9	NFT error code
10	local error message pointer
11	NFT error code enhancement pointer
12	end pointer

(continued on next page)

(RINIT, continued)



(AINIT, continued)



misc flags

0	0	0	0	0	0	0	C/P

C/P (if set) = receiving end is initially the consumer (otherwise, the producer)

system type

1 = MPE
12 (octal) = RTE
24 (octal) = HP-UX and VAX/VMS Systems
36 (octal) = MS-DOS
50 (octal) = other systems

buffer size

maximum number of bytes that originating node can handle

capability mask

last byte:

0	0	0	CR	0	H	0	R

R = may change roles from transfer to transfer (for example, initiator-producer connection can be used as producer-consumer)

H = sending system has hierarchical file system that supports multilevel directories.

CR = system supports checkpoint/restart. Data packets wil have 6-byte NFT header

session ID pointer

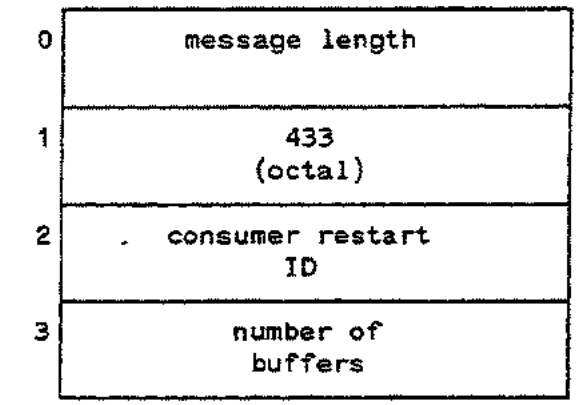
points to an internal number which identifies a previously-created environment to be used by NFT

NFT error code

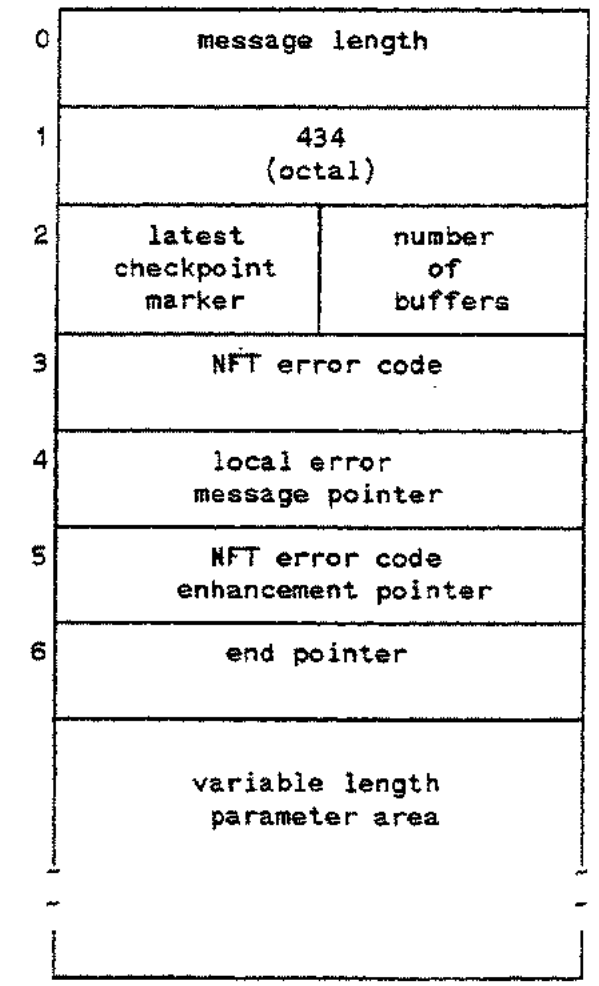
0 = successful transfer; refer to NFT error numbers for other values

OFFERR/AOFFERR

OFFERR



AOFFERR

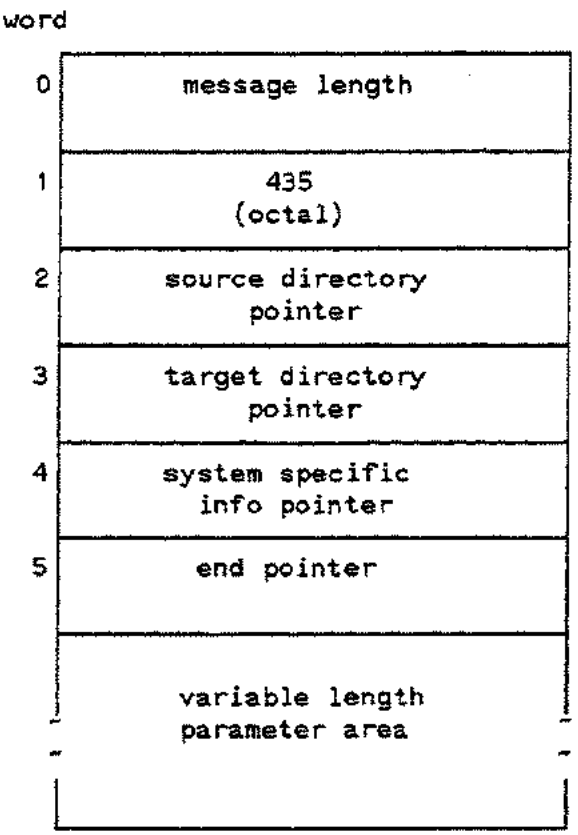


consumer restart ID	specifies the restart indentifier of the current file transfer on the system that originates the message which contains this parameter. Zero indicates no check pointing will be done.
number of buffers	number of outstanding packets permitted
latest checkpoint marker	identifies the current checkpoint record of the mainframe that originates this parameter
NFT error code	0 = successfull transfer; refer to NFT error numbers for other values
local error message pointer	points to string (ASCII) more fully explaining local error message

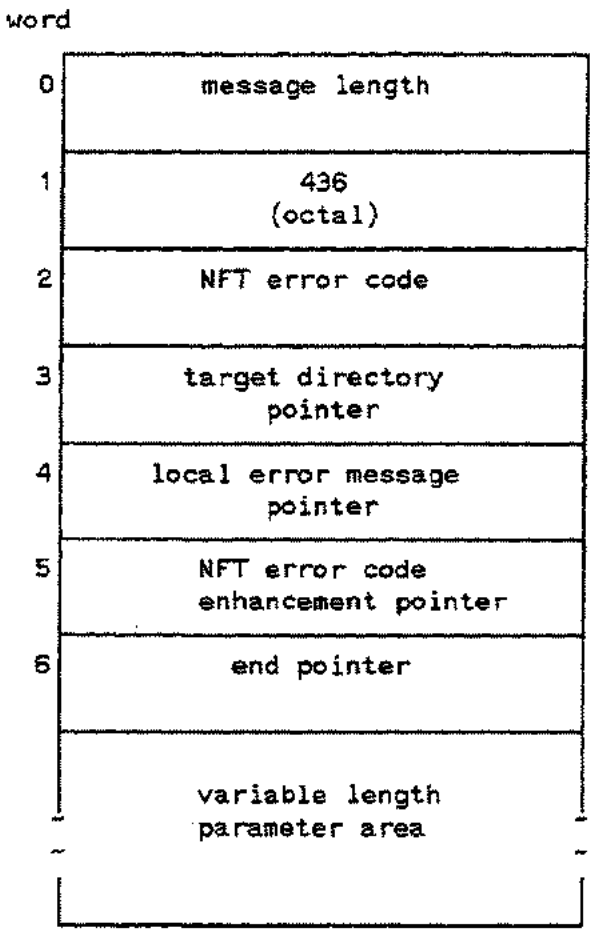
NFT error code enhancement pointer	points to string (ASCII) more fully explaining NFT error message
end pointer	points to the end of the packet

DIRECTORY/ADIRECTORY

DIRECTORY

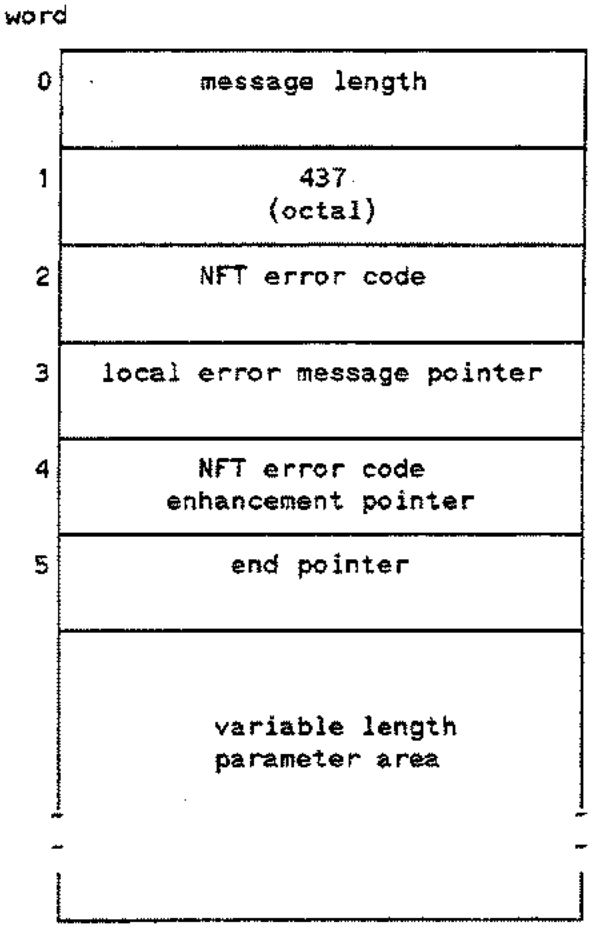


ADIRECTORY



source directory pointer	points to source directory pathname
target directory pointer	points to target directory name
system specific info pointer	points to area containing system-specific information for the directory consumer
NFT error code	0=successful transfer; refer to NFT error numbers for other values
local error message pointer	points to string (ASCII) more fully explaining local error message
NFT error code enhancement pointer	points to string (ASCII) more fully explaining NFT error message

WARN



NFT error code — 0 = successful transfer; refer to NFT error numbers for other values

NS-ARPA/1000 FTP MESSAGES

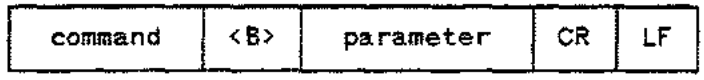
SECTION 4

This section contains illustrations of FTP messages.

FTP on the HP 1000 was developed in accordance with industry standards, with specifications from the Department of Defense's ARPANET.

FTP Commands

FTP commands use the following message format:

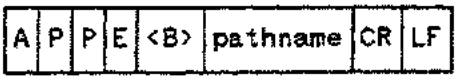


command	FTP command, character string of variable length, terminated by a blank (B), or by carriage return (CR), line feed (LF) if the command does not have a parameter.
<B>	indicates the blank character (ASCII set, 40 octal).
parameter	command parameter, character string of variable length, terminated by carriage return (CR), line feed (LF).
CR	indicates the carriage return character (ASCII set, 15 octal).
LF	indicates the line feed character (ASCII set, 12 octal).

FTP Command Messages

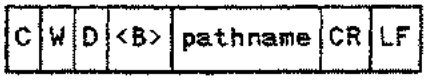
The FTP commands implemented on the HP 1000 are listed below, in alphabetical order:

ABOR	Tells the server to abort the previous FTP service command and any associated transfer of data.
CR,LF	Carriage return, line feed.

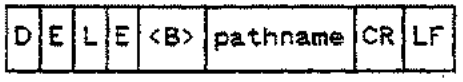


APPE	Accepts data transferred via the data connection and appends the data to the specified file. If the file does not exist, it will be created.
<B>	Blank character.
pathname	Pathname of file to append to.
CR,LF	Carriage return, line feed.

CDUP	Changes working directory to the parent directory of the current working directory.
CR,LF	Carriage return, line feed.

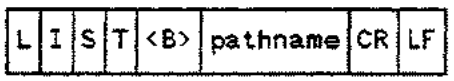


CWD	Changes working directory.
<B>	Blank character.
pathname	Pathname of new working directory.
CR,LF	Carriage return, line feed.

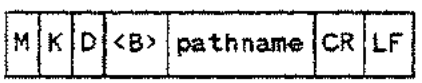


DELE	Deletes the specified file.
<B>	Blank character.
pathname	File to be deleted.
CR,LF	Carriage return, line feed.

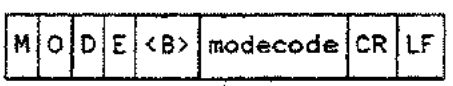
HELP	Displays help information.
<B>	Blank character.
string	Command for which help is requested.
CR,LF	Carriage return, line feed.



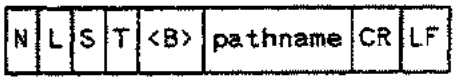
LIST	Gives listing of specified directory.
<B>	Blank character.
pathname	Directory to be listed.
CR,LF	Carriage return, line feed.



MKD	Makes a new directory.
<B>	Blank character.
pathname	Pathname of new directory.
CR,LF	Carriage return, line feed.

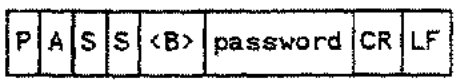


MODE	Specifies data transfer mode.
<B>	Blank character.
modecode	Transfer mode.
CR,LF	Carriage return, line feed.



NLST	Causes a directory listing to be sent from server site to user site.
<B>	Blank character.
pathname	Directory to be listed.
CR,LF	Carriage return, line feed.

NOOP	Specifies no action other than that the server sends an OK reply.
CR,LF	Carriage return, line feed.



PASS	Specify the user's password. This command must be immediately preceded by the user name command (USER), and for some sites, completes the user's identification for access control.
<B>	Blank character.
password	Password.
CR,LF	Carriage return, line feed.

PORT	Specifies data connection port. There are defaults for both the user and server data ports.
<B>	Blank character.
hostport	Host port for the data connection. It is a concatenation of a 32-bit IP host address and a 16-bit TCP port

address. This address information is broken down into 8-bit fields and the value of each field is transmitted as a decimal number (in character string representation). The fields are separated by commas. A PORT command would be:

PORT *h1, h2, h3, h4, p1, p2*

where, *h1* is the high order 8 bit of the internet host address.

CR,LF — Carriage return, line feed.

P	W	D	CR	LF

PWD — Displays the name of the current working directory.

CR,LF — Carriage return, line feed.

Q	U	I	T	CR	LF

QUIT — Terminates the current FTP session and closes the control connection. If file transfer is in progress, the connection will remain open for result response and the server will then close it.

CR,LF — Carriage return, line feed.

R	E	T	R	<B>	pathname	CR	LF

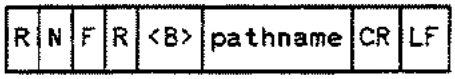
RETR — Causes the server to transfer a copy of the file to the user at the other end of the data connection.

<B> — Blank character.

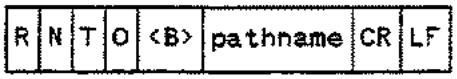
pathname — File to be retrieved.

CR,LF — Carriage return, line feed.

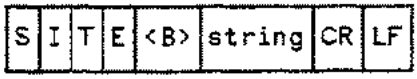
RMD	Removes a directory.
<B>	Blank character.
pathname	Directory to be removed.
CR,LF	Carriage return, line feed.



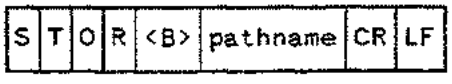
RNFR	Specifies the old pathname of the file which is to be renamed. This command must be immediately followed by a RNTO (rename to) command.
<B>	Blank character.
pathname	Old pathname of file to be renamed.
CR,LF	Carriage return, line feed.



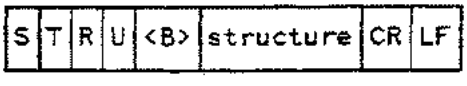
RNTO	Specifies the new pathname of the file for renaming. This command must be immediately preceded by the RNFR (rename from) command.
<B>	Blank character.
pathname	New pathname used for renaming.
CR,LF	Carriage return, line feed.



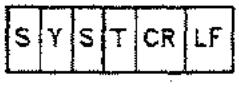
SITE	Provides services specific to a system that are essential to file transfer but not sufficiently universal to be included as commands in the protocol.
<B>	Blank character.
string	Service provided at a server site.
CR,LF	Carriage return, line feed.



STOR	Causes the server to accept the data transferred via the data connection and to store the data as a file at the server site.
<B>	Blank character.
pathname	File to store the data.
CR,LF	Carriage return, line feed.



STRU	Specifies data transfer structure.
<B>	Blank character.
structure	Structure to be used.
CR,LF	Carriage return, line feed.



SYST	Displays the remote system type.
CR,LF	Carriage return, line feed.

T	Y	P	E	<B>	typecode	CR	LF

TYPE	Specifies data transfer type.
<B>	Blank character.
typecode	Type to be used.
CR,LF	Carriage return, line feed.

U	S	E	R	<B>	username	CR	LF

USER	Specifies user name. This command is normally the first command transmitted by the user after the control connections are made.
<B>	Blank character.
username	User name to be used.
CR,LF	Carriage return, line feed.

FTP Reply Codes

FTP Reply Codes uses the following message format:

rcode	<B>	message	CR	LF

rcode — FTP reply code, character string of variable length, terminated by a blank (B).

<B> — indicates the blank character (ASCII set, 40 octal).

message — reply code message, character string of variable length, terminated by carriage return (CR), line feed (LF).

CR — indicates the carriage return character.

LF — indicates the line feed character.

The following table lists the FTP reply codes and their accompanying messages currently implemented on the HP 1000.

FTP Command Reply Codes.

Command Code	Meaning
125	Data connection already open; transfer starting.
150	File status okay; about to open data connection.
200	Command okay.
202	Command not implemented, superfluous at this site.
214	FTP Server commands currently implemented.
220	FTP Server ready for new user.
221	Service closing control connection.
226	Closing data connection.
230	User logged in, proceed.
250	Requested file action okay, completed.

FTP Command Reply Codes (continued).

Command	Code
251	*directory_name* is the current working directory.
257	*directory_name* created.
331	User name okay, need password.
350	Requested file action pending further information.
425	Cannot open data connection.
426	Connection closed; transfer aborted.
450	Requested file action not taken.
451	Requested action aborted; local error in processing.
452	Requested action not taken.
500	Syntax error, command unrecognized.
501	Syntax error in parameter or arguments.
502	Command not implemented.
503	Bad sequence of commands.
504	Command not implemented for that parameter.
530	Not logged in.
532	Need account for storing files.
550	Requested action not taken.
552	Requested file action aborted.
553	Requested action not taken.

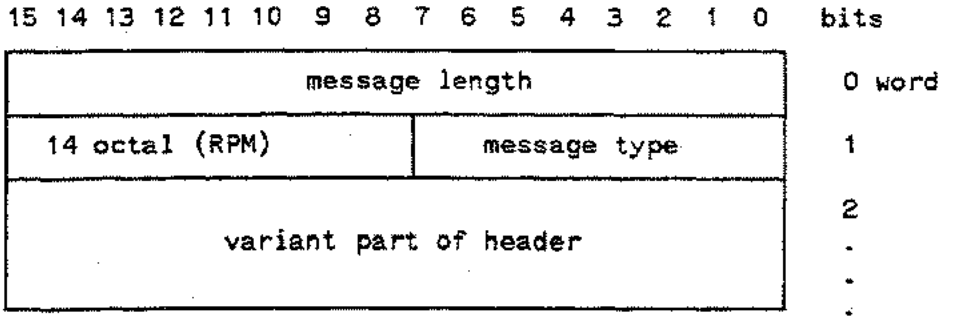
NS-ARPA/1000 RPM MESSAGES

SECTION 5

This Remote Process Management (RPM) sections apply to RPM on NS-ARPA/1000 only.

Communication in NS-ARPA/1000 between the RPM parent program and the RPM monitor takes place with RPM messages sent by NetIPC calls across the network. These messages are comprised of two portions: header and variable data. The header in turn consists of a common header to identify the RPM protocol and message type, and other fields specific to that message type.

The RPM message header consists of three common fields and other variant fields depending on the message type as shown below:



`message length` Byte length of message, including this word.

`14 octal` For RPM, the protocol ID is 14 octal (C hexadecimal or 12 decimal).

`message type` Message type. There are 10 message types in RPM as shown in Table 5-1.

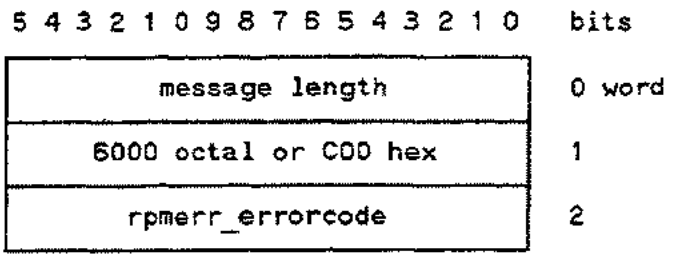
TABLE 5-1. RPM Message Types

Word 1 (octal)	Word 1 (hex)	Message Type (octal)	Message
6000	C00	0	RPMError Reply
6001	C01	1	RPMCreate Request
6002	C02	2	RPMCreate Reply
6003	C03	3	RPMKill Request
6004	C04	4	RPMKill Reply
6005	C05	5	RPMLength Request
6006	C06	6	RPMLength Reply
6007	C07	7	RPMSon Complete
6010	C08	10	RPMControl Request
6011	C09	11	RPMControl Reply

The structure of each RPM message header is given in the following subsections.

RPMError Reply Message

The RPMError reply message is sent by the RPM monitor to the parent if an unexpected message type (a message other than one of those specified in Table 5-1) is received on an open virtual circuit connection.

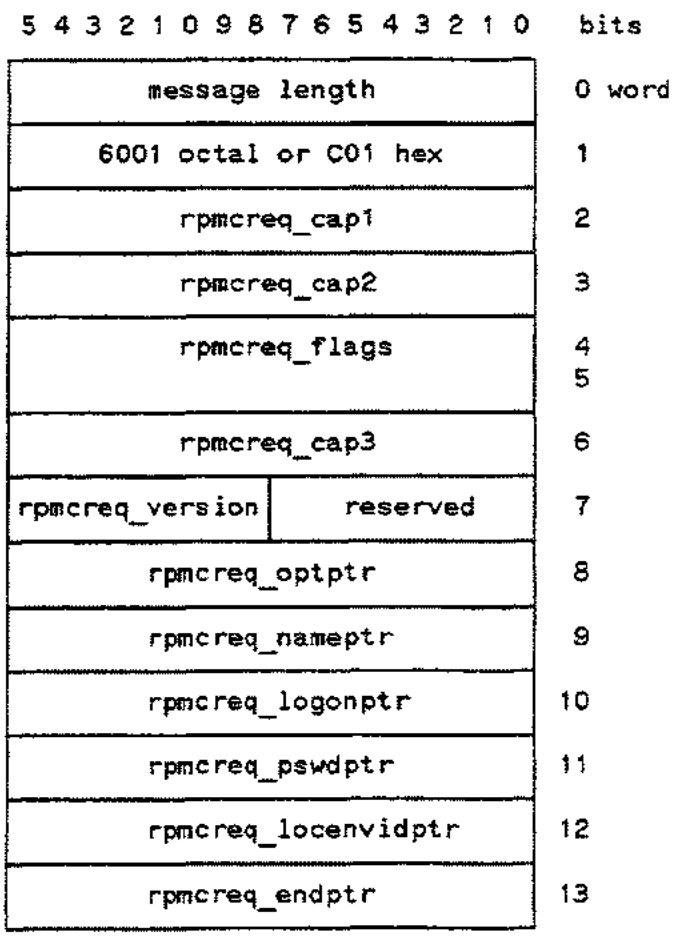


rpmerr_errorcode Refer to the *NS-ARPA/1000 Error Message and Recovery Manual.*

RPMCreate Request Message

The RPMCreate request message is sent from the parent program to the RPM monitor. This message contains all the information needed to locate or set up the child program environment and to create the child program.

The RPMCreate request message follows the standard NS message format. For each variable length field there is a byte offset from the beginning of the message.



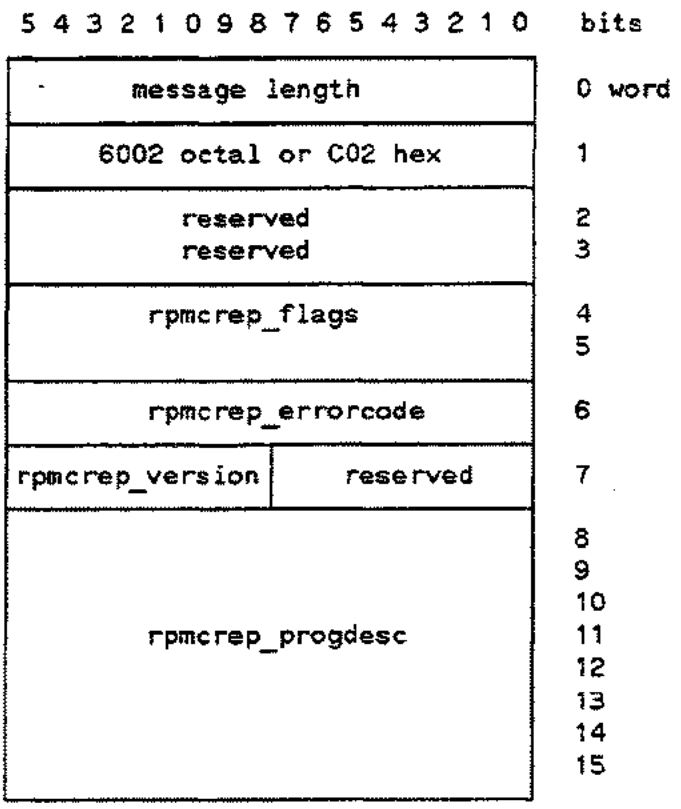
rpmcreq_cap1 — This, with rpmcreq_cap2 and rpmcreq_cap3, is a six-byte session identifier to uniquely identify the local session. This field consists of a random number unique within this node, while rpmcreq_cap2 and rpmcreq_cap3 together constitutes the IP address of the local node, so that the session identifier will be unique within the entire catenet.

rpmcreq_cap2 — This is the second part of the session identifier and contains the first two bytes of the IP address for this node.

rpmcreq_flags	Currently three flags are defined: wait: wait for child program. dependent: child program is in dependent mode. session: child program is shareable.
rpmcreq_cap3	Third part of the session identifier containing the third and fourth bytes of the IP address for this node.
rpmcreq_version	For NS-ARPA/1000 RPM, the version number is 1.
rpmcreq_optptr	Byte offset to start of opt array.
rpmcreq_nameptr	Byte offset to start of program name.
rpmcreq_logonptr	Byte offset to start of logon string.
rpmcreq_pswdptr	Byte offset to start of password string.
rpmcreq_locenvidptr	Byte offset to remote session ID. This is only included when a specific remote session is desired. This is not used for NS-ARPA/1000 RPM.
rpmcreq_endptr	Byte offset to one byte after end of message.

RPMCreate Reply Message

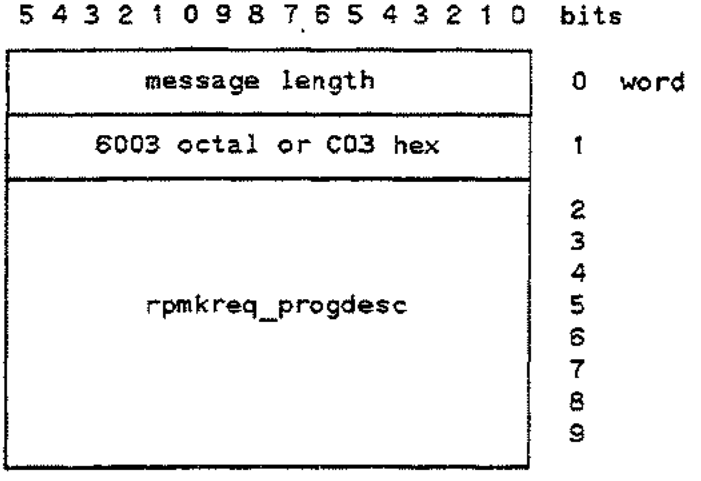
The RPMCreate reply message is sent from the RPM monitor to the parent program to indicate the success or failure of an RPMCreate request. If the request succeeded, this message returns the program descriptor which uniquely identifies the child program. If the request failed, it returns the RPM error code for the failure.



rpmcrep_flags	This will be set to the flags specified in the RPMCreate request message that are successfully executed.
rpmcrep_errorcode	Error code returned by the RPM monitor. If the child program was successfully created, 0 is returned.
rpmcrep_version	For NS-ARPA/1000 RPM, the version number is 1.
rpmcrep_progdesc	An eight-word number for the program descriptor to uniquely identify the child program in the catenet. This is valid only if rpmcrep_errorcode = zero.

RPMKill Request Message

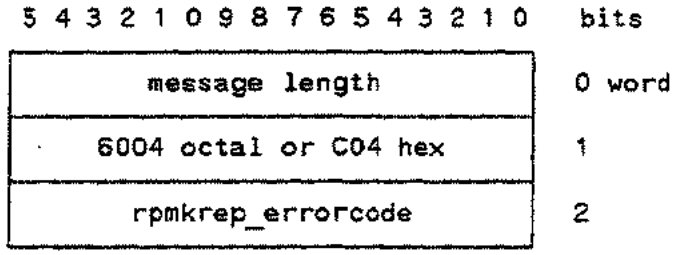
The RPMKill request message is sent from the caller of the RPMKill call to the RPM monitor. This message contains the program descriptor of the child program to be terminated.



`rpmkreq_progdesc` The eight-word program descriptor of the child program to kill. This was returned by the RPM monitor in the RPMCreate reply message.

RPMKill Reply Message

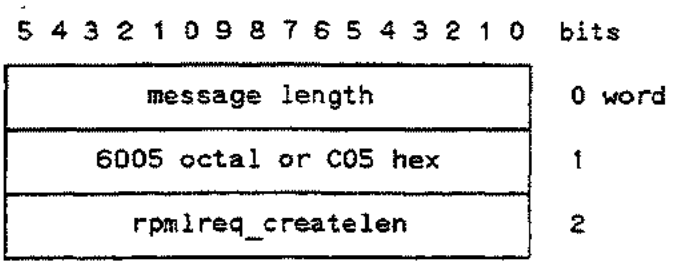
The RPMKill reply message is sent from the RPM monitor to the parent program to indicate the success or failure of the RPMKill request.



rpmkrep_errorcode — The RPM error code. If the RPMKill request was successful, then the code will be zero.

RPMLength Request Message

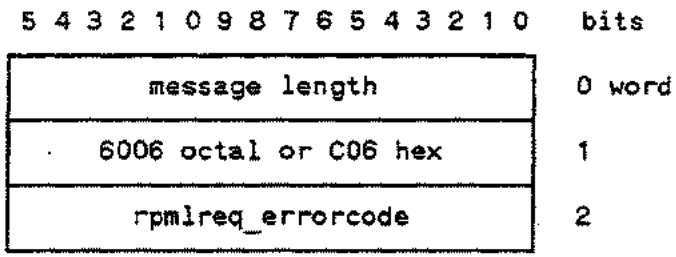
The RPMLength request message is sent from the parent program to the RPM monitor. This message is sent when the length of an RPMCreate request message to be sent exceeds a normal maximum (256 bytes). When the RPM monitor receives this message, it attempts to expand its message buffer and NetIPC parameters in anticipation of the long RPMCreate request message. NS-ARPA/1000 RPM returns an error code of zero as long as the length requested in the RPMLength request message is less than 1024 bytes.



rpmlreq_createlen The byte length of the RPMCreate or RPMControl request message to be sent.

RPMLength Reply Message

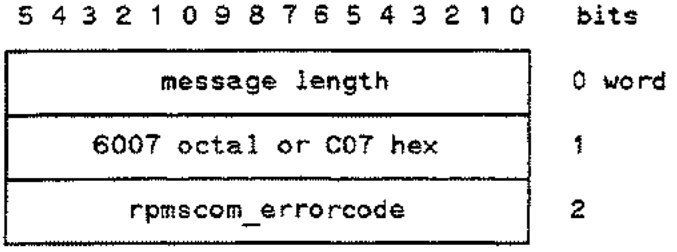
The RPMLength reply message is sent from the RPM monitor to the parent program after an RPMLength request has been received. It indicates if the RPM monitor can receive long RPMCreate message.



rpmlrep_errorcode Error code indicating the success or failure of the RPMLength request. If the request is granted, the error code is zero.

RPMSonComplete Reply Message

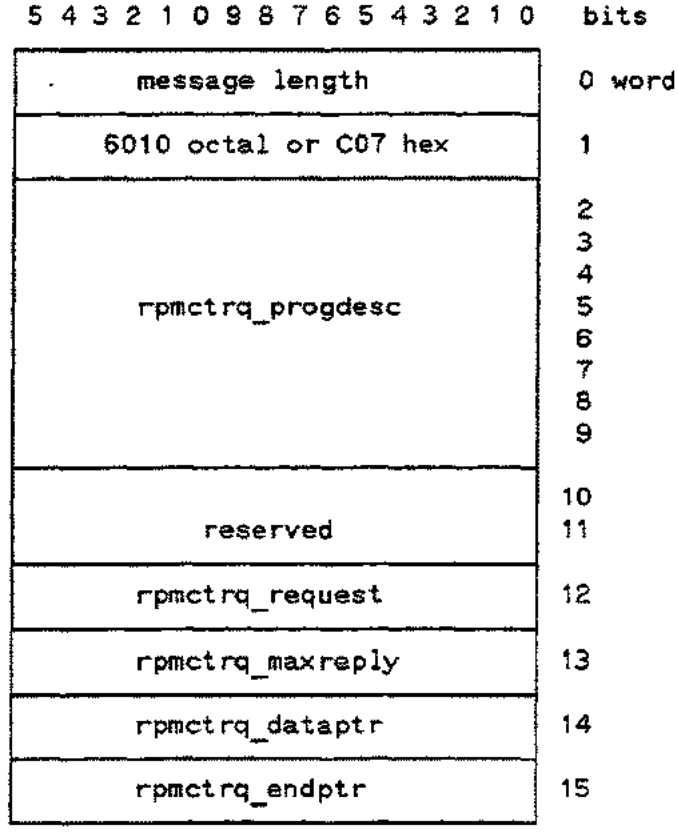
The RPMSonComplete reply message is sent from the RPM monitor to the parent program to inform the latter that the child program has terminated.



rpmscom_errorcode — Error code indicating the completion status of the child program. If the child program completed successfully, the error code is zero.

RPMControl Request Message

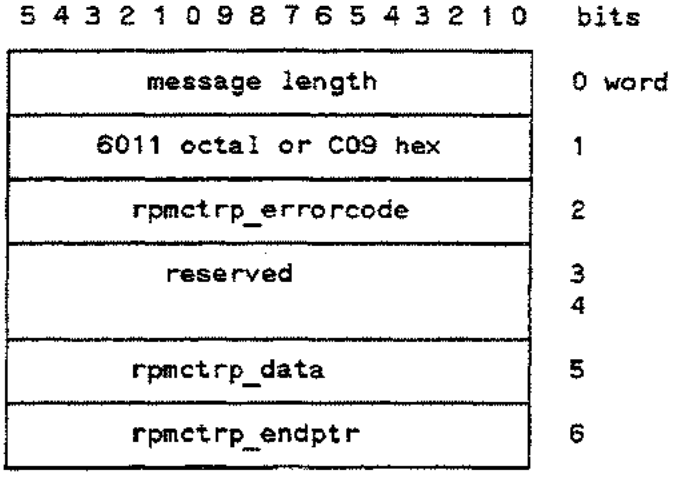
The RPMControl request message is sent by the parent program to the RPM monitor to request for certain control operations to be performed on the child program specified in the program descriptor field. This may involve common operations such as suspending and resuming the program or system-dependent options.



rpmctrq_progdesc	16-byte program descriptor to identify the child program on which the control operations are to be performed.
rpmctrq_request	The request code number for this control message.
rpmctrq_maxreply	Length, in bytes, of the maximum amount of data to be returned in the reply message.
rpmctrq_dataptr	Byte offset to the start of the data array for the control request.
rpmctrq_endptr	Offset to end of message. Points to one byte after the end of the message.

RPMControl Reply Message

The RPMControl reply message is sent by the RPM monitor to the parent program to inform it of the status of the RPMControl request.



`rpmctrp_errorcode` Return code of the control request. This will be zero if the control request was successfully executed, or an appropriate error code otherwise.

`rpmctrp_data` Offset to start of data field, if the control reply returns some data. If no data is returned, this field will be set to the value of `rpmctrp_endptr`.

`rpmctrp_endptr` Offset to one byte after end of message.

VIRTUAL TERMINAL ACCESS MESSAGES

SECTION 6

Virtual Terminal (VT) Access allows a user from a terminal on a local HP 3000 system to log on to a remote HP 3000 and conduct an interactive session as if the terminal were directly connected to the remote system. Character-mode applications and block-mode applications using VPLUS are supported. For PCs connected to an HP 3000 over IEEE 802 LAN, Virtual Terminal Access allows terminal emulation access to most HP 3000 applications. The PC must be using HP OfficeShare Network software and hardware.

This section covers Virtual Terminal Access messages.

There are two logical components in a Virtual Terminal (VT) connection. Virtual Terminal (VT) Access requires the following modules:

- TM (Terminal Monitor). Logical components which resides on the same physical system as the physical terminal (i.e. at the local node). It receives VT messages through the network from the Application Monitor and translates them back to terminal/file control requests for the physical terminal.

- AM (Application Monitor). Logical components which resides on the same system as the application (i.e. at the remote node which contains CI, VPLUS, etc). It receives terminal control requests from the application (i.e., the CI) and translates these requests into VT messages and sends them to the Terminal Monitor.

Application Monitor and Terminal Monitor are implemented as parts of the VTServer and DSServer programs.

VT General Flow

This subsection describes the general sequence of events of a Virtual Terminal access session on an MPE XL system.

ON LOCAL NODE.

1. At the local node, the user invokes VT to acces CI on the remote node. CXREMOTE sends message to DSDAD and causes CI to block on port.

2. DSDAD creates TM.

3. TM adopts under JSMAIN.

4. TM FOPENS $STDIN/$STDLIST.

5. TM calls IPCConnect to connect with remote DSDAD.

ON REMOTE NODE.

6. Remote DSDAD calls IPCRecvcn to get TM's connection request.

7. DSDAD creates AM and hands off connection.

8. Remote AM and local TM negotiates.

9. AM allocates VT LDEV and VTLDM is created.

10. VTLDM calls START.DEV.REL, sending message to session.

11. Session creates JSMAIN.

12. AM adopts under JSMAIN.

13. JSMAIN FOPENS LDEV, and sends logon prompt via STARTLOGON.

14. JSMAIN creates CI.

TERMINATING VT.

15. After the session, the agreed termination goes as follows.

16. JSMAIN calls NS ADOPT OUT SERVERS, sending message to DSDAD and waits for reply.

17. DSDAD tells AM to "stop" and "adopt out" and wait for reply.

18. AM telss VTLDM to "halt" and wait for DCLOSE.

19. After DSDAD tells JSMAIN AM has adopted out, JSMAIN calls KILL_CHILD to kill CI.

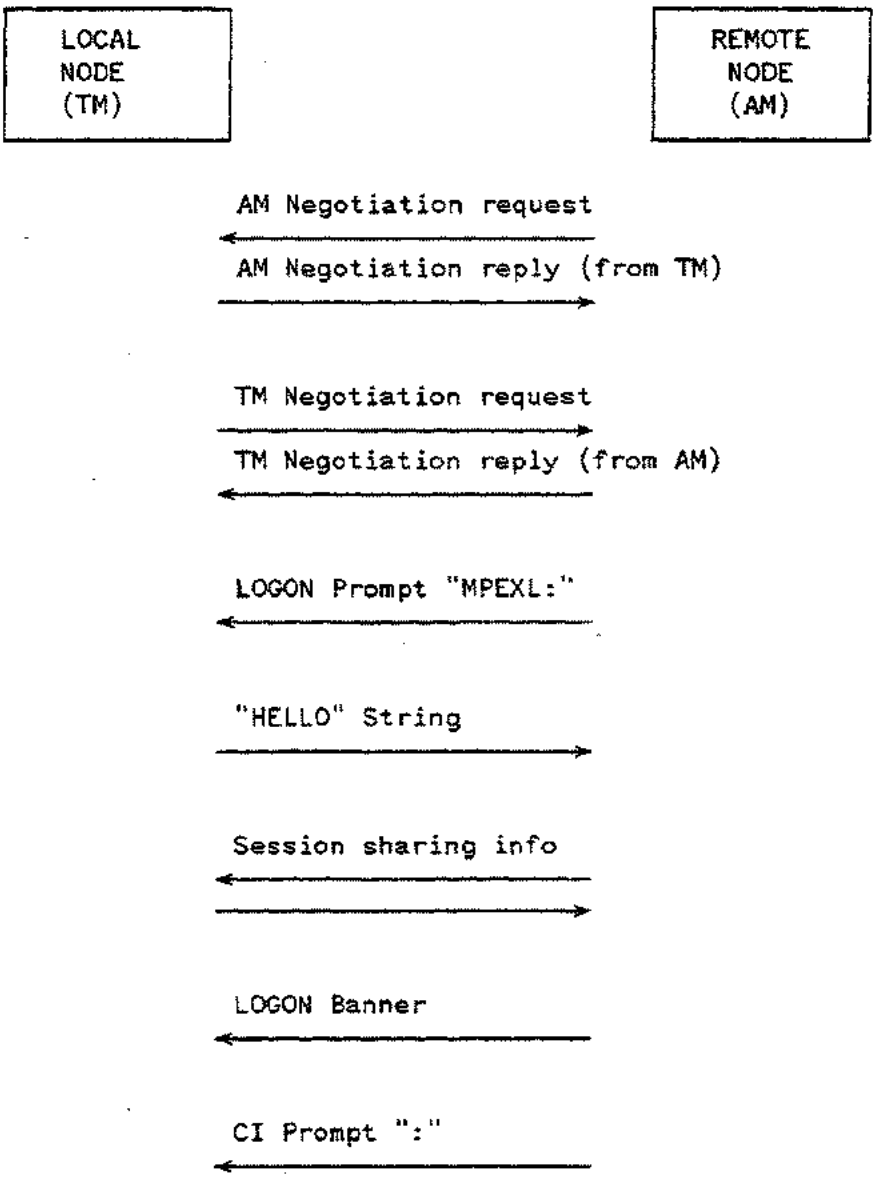
20. JSMAIN sends DCLOSE to VTLDM.

21. VTLDM tells AM to deallocate VT LDEV.

22. AM releases VT LDEV and sends "ENV_TERM" requesst to TM.

23. TM sends "ENV_TERM" reply and cleans up. TM must adopt out from JSMAIN and FCLOSE $STDIN/$STDLIST.

Typical Virtual Terminal Logon Sequence



NOTE

Logon UDCs may generate additional terminal control messages.

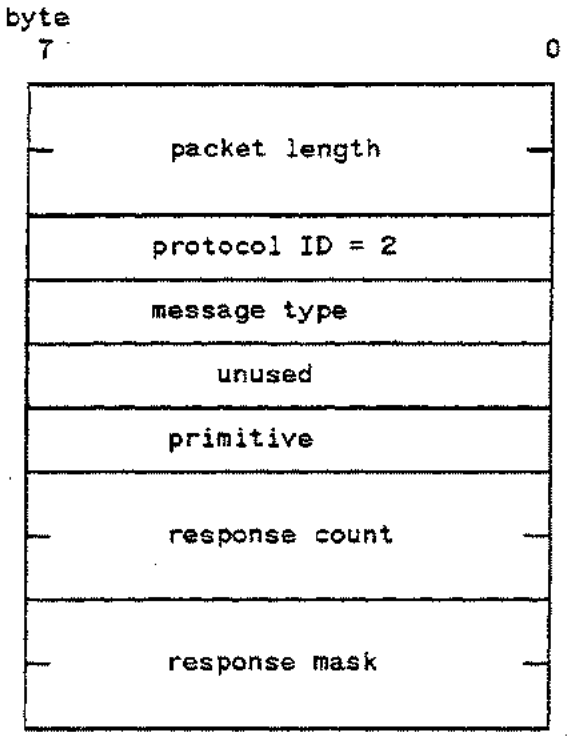
Virtual Terminal Message Types and Primitives

Messages are defined by message type and primitive field, according to the following table.

Table 6-1. Virtual Terminal Message Types/Primitives

VT Messag Types	Primitives
0 - env__cntl__req	0 - terminal__negotiate 1 - application__negotiate 2 - terminate 3 - logon__info
1 - env__cntl__rsp	same as above
2 - terminal__io__req	0 - read 1 - write 2 - write__read 3 - abort
3 - terminal__io__rsp	none
4 - term__cntl__req	0 - tcs__break__set 1 - tcs__driver__set
5 - term__cntl__rsp	none
6 - application__cntl__req	0 - application__invoke__break
7 - application__cntl__rsp	none
8 - hpe__cntl__req	0 - hpe__driver__set 1 - hpe__get__info
9 - hpe__cntl__rsp	none

VT Message Header and Generic Response



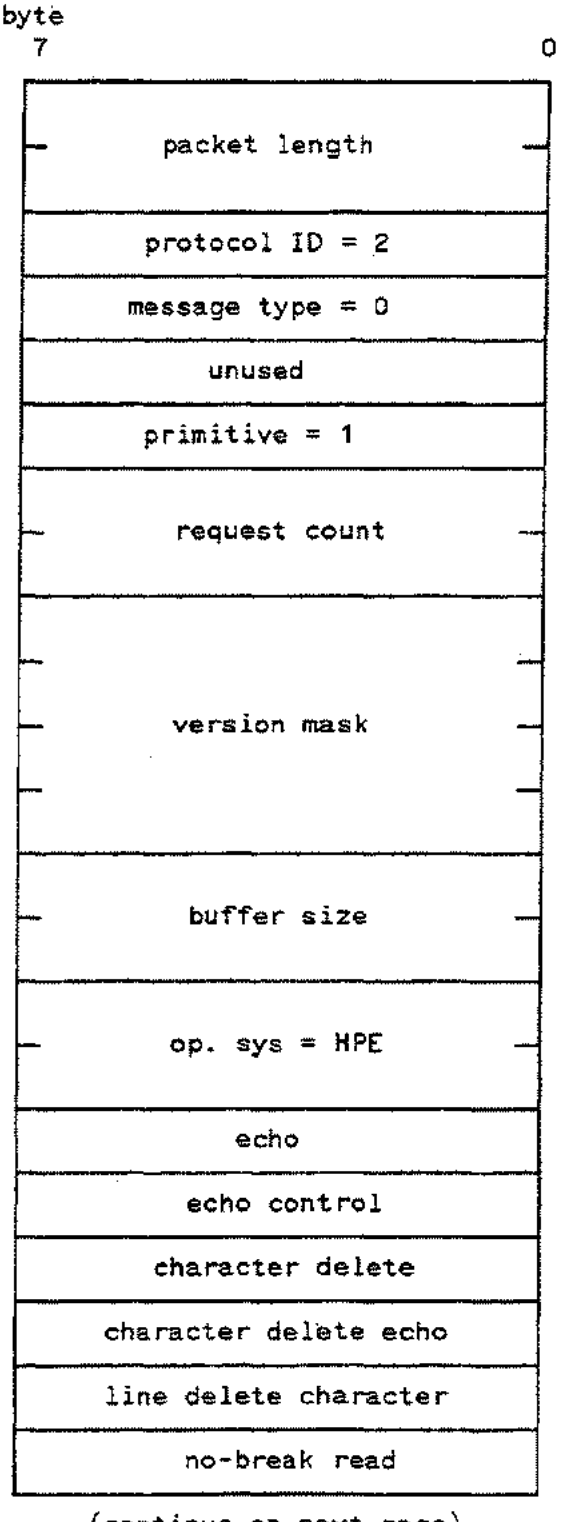
message type	0 = env_cntl_req (environment control) 1 = env_cntl_rsp 2 = terminal_io_req (terminal i/o) 3 = terminal_io_rsp 4 = term_cntl_req (terminal control) 5 = term_cntl_rsp 6 = appln_cntl_req (application control) 7 = appln_cntl_rsp 8 = hpe_cntl_req (HPE only messages) 9 = hpe_cntl_rsp
primitive	used with message type to define the message
response count	sequence ID used to match requessts and replies
response mask	bitmask of type below:

bit number	code
00	no_error
01	request_failed
02	bad_msg_format
03	bad_primitive
04	bad_parameter
05	rsp_extra03
06	rsp_extra02

07	rsp_extra01
08-15	rsp_extras

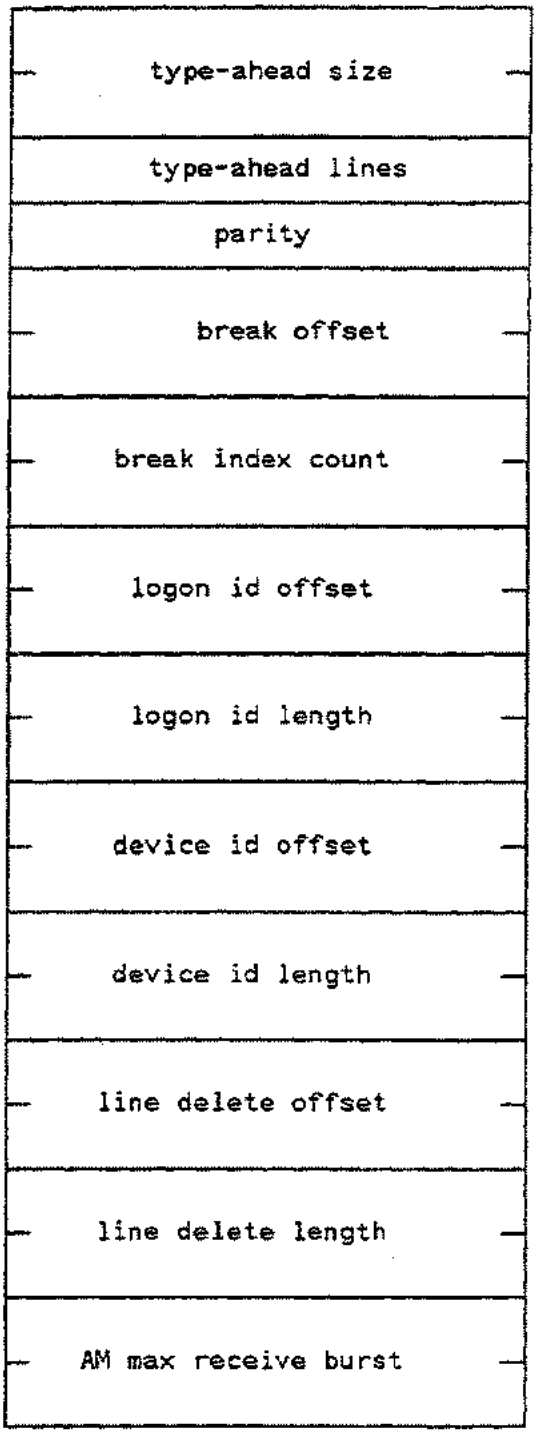
Application Monitor Negotiation Request

(Type 0, Primitive 1)



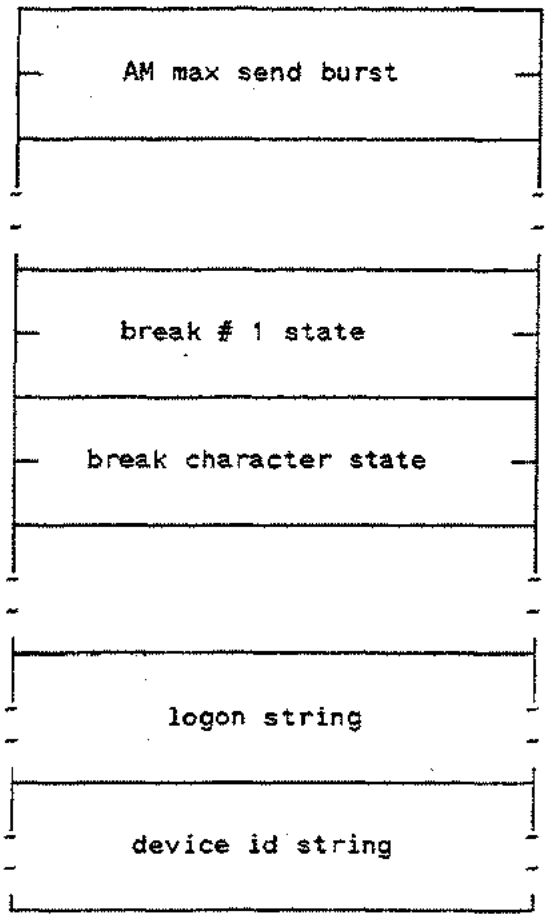
(continue on next page)

AM's Negotiation Request (continued)



(continue on next page)

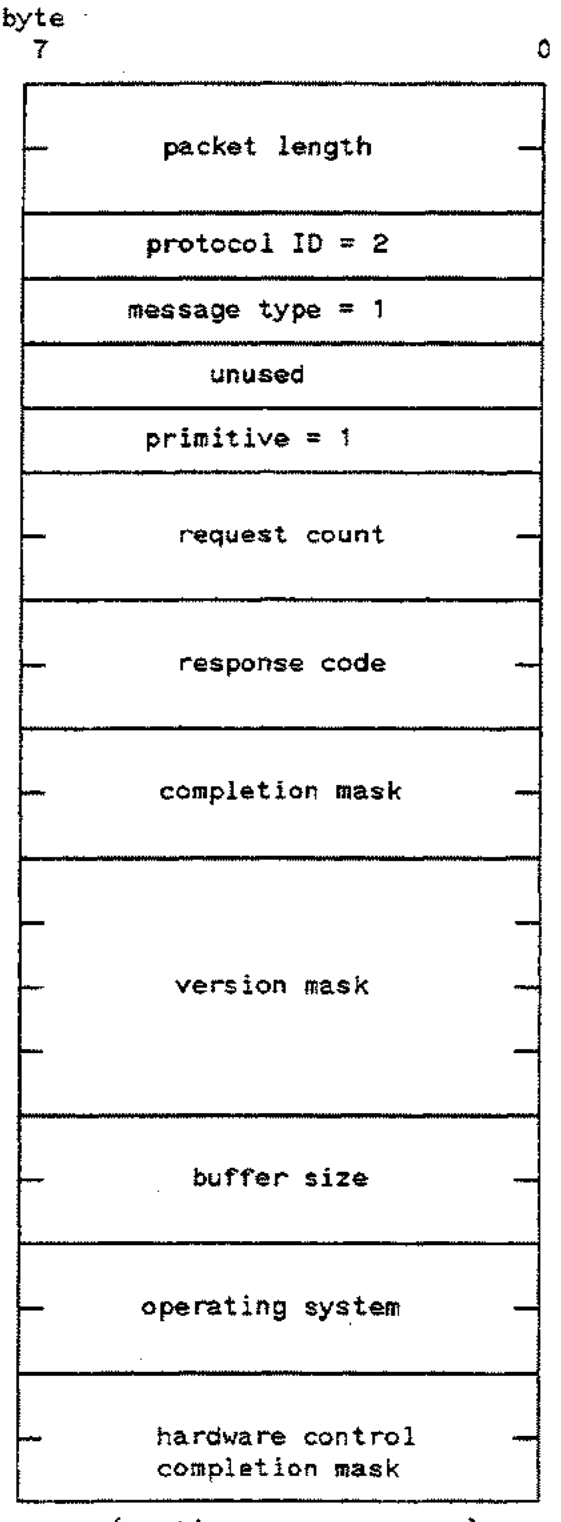
AM's Negotiation Request (continued)



parity	reserved
echo character deletion string	0 = dont_care 1 = echo_backspace 2 = echo_bs_slash 3 = echo_bs_sp_bs
echo control	0 = dont_care 1 = dont_allow 2 = switch_ok 3 = notify_on_switch

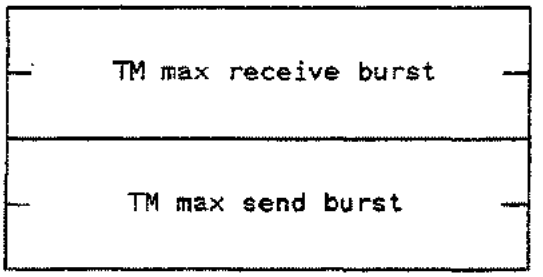
Application Monitor Negotiation Reply (sent by TM)

(Type 1, Primitive 1)



(continue on next page)

AM's Negotiation Reply (continued)



completion mask

bit number	code
00	successfull
01	echo_failed
02	break_failed
03	no_device
04	reserved
05	reserved
06	reserved
07	reserved
8-15	break_index

operating system

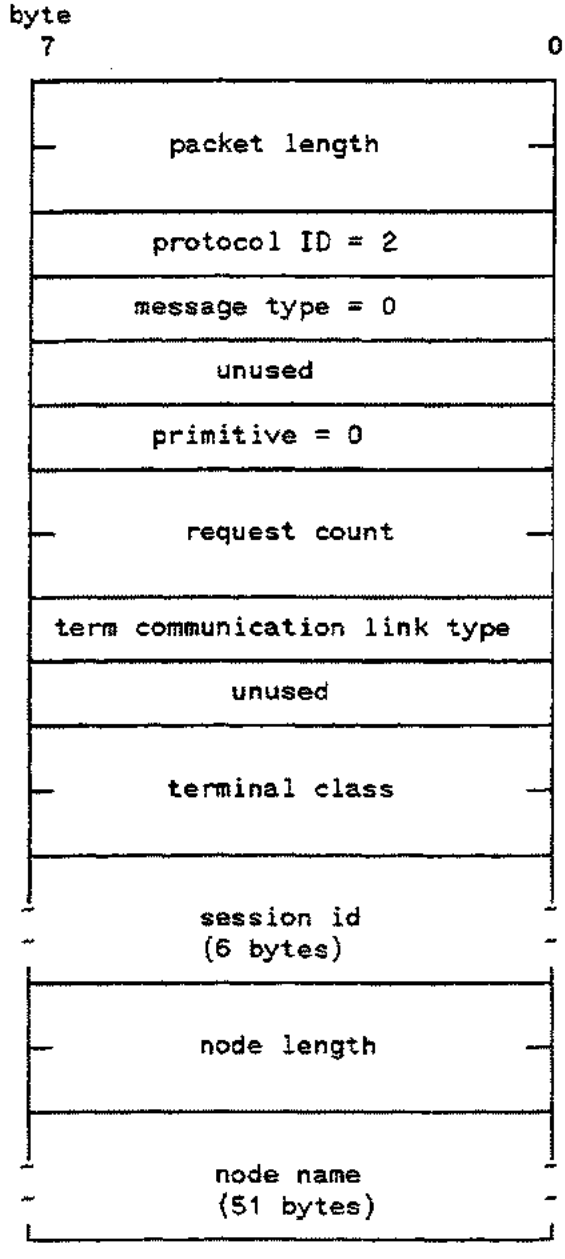
first byte = operating system type
- 1 = MPE XL
- 2 = MPE V
- 4 = HP-UX
- 6 = MS DOS

second byte = operating system internal version

hardware control completion mask (if set, TM cannot support)

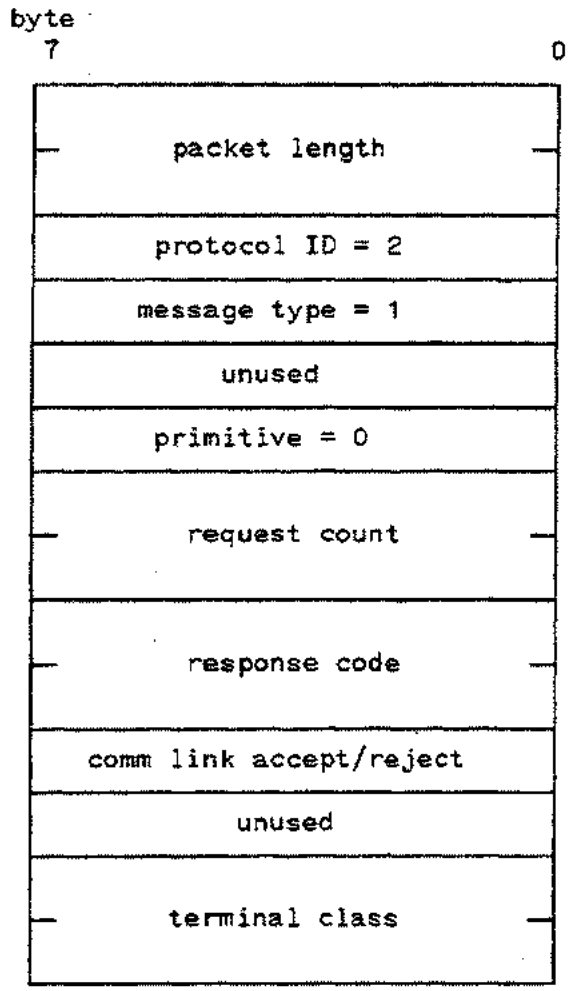
bit number	code
00	echo_cntl
01	char_del
02	char_echo
03	line_del
04	line_char
05	type_ahead
06	no_read_brk
07	parity
8-15	reserved

Terminal Monitor Negotiation Request (Type 0, Primitive 0)



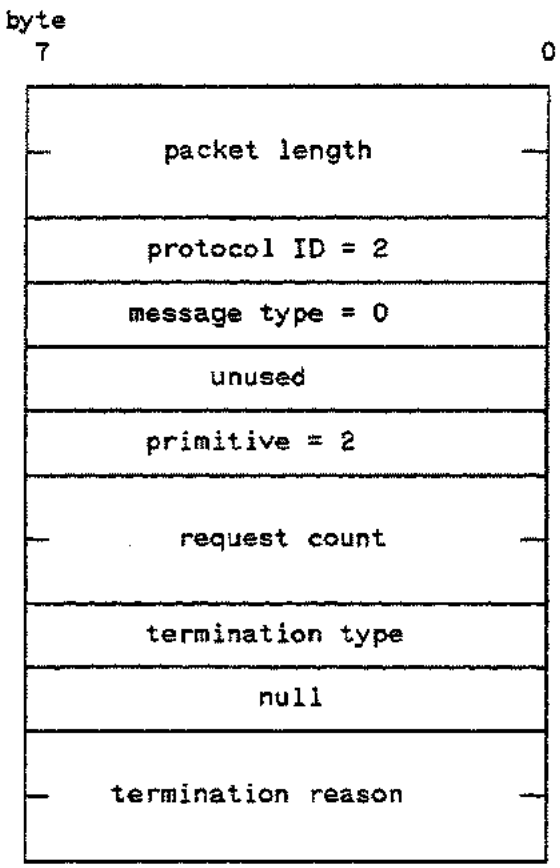
Terminal Monitor Negotiation Reply (sent by AM)

(Type 1, Primitive 0)



Terminate Message Request

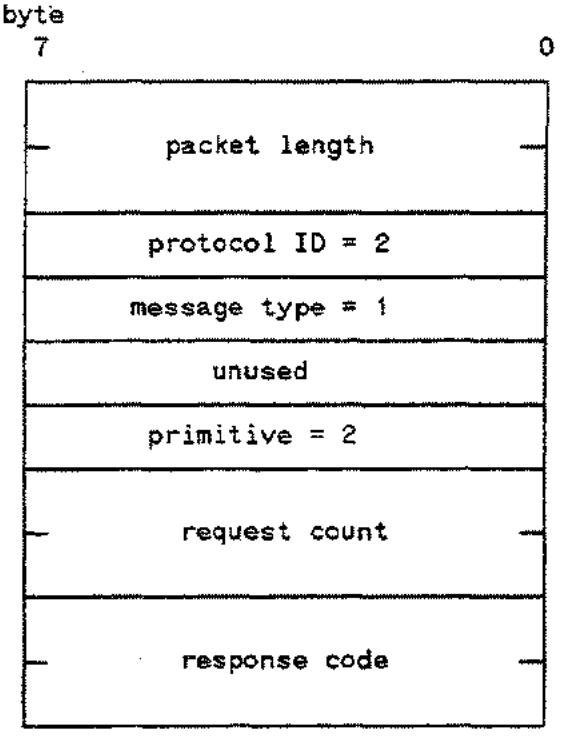
(Type 0, Primitive 2)



termination type
0 = tk_agreed
1 = tk_forced

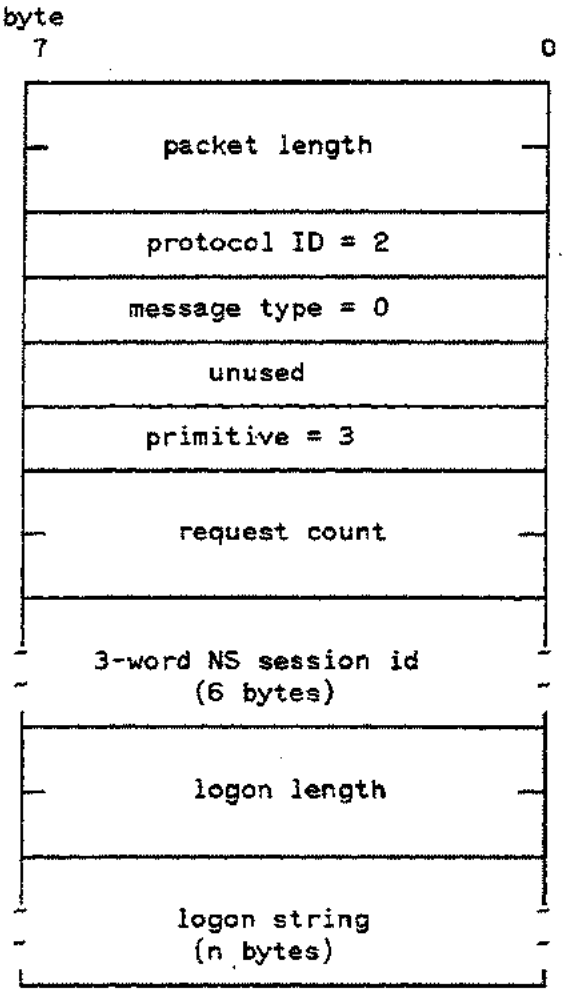
termination reason
0 = term_normal
1 = term_user_abort
2 = term_control_req
3 = term_pgm_error
4 = term_bad_negotiate
5 = term_VTS_error
6 = term_logon_fail
7 = term_init_fail
8 = term_no_device

Termination Reply
(Type 1, Primitive 2)



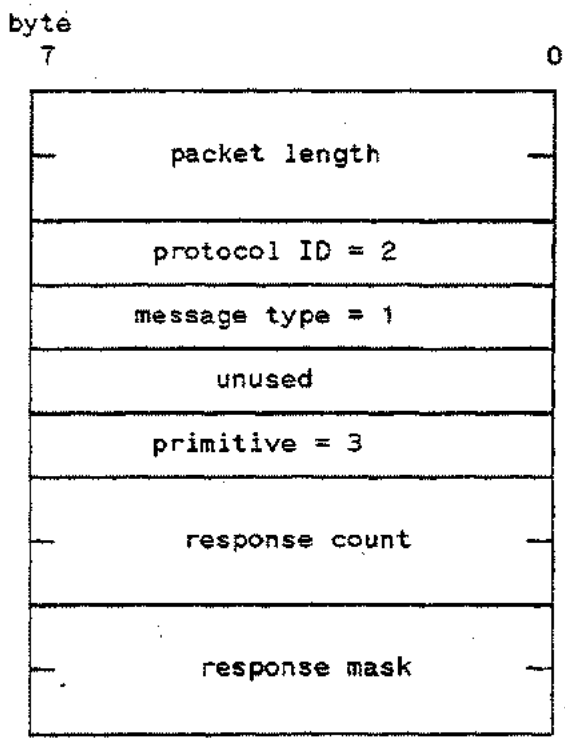
Logon Info Message Request & Response

Session Sharing Request (Type 0, Primitive 3)

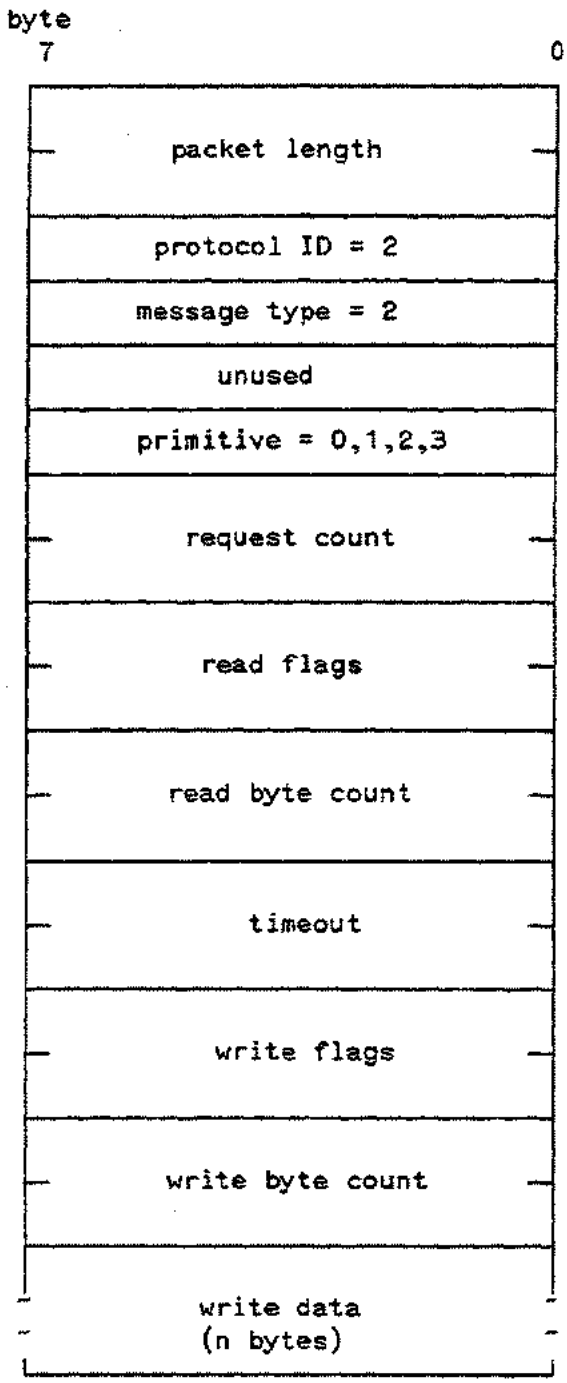


Logon Info Message Request & Response

Session Sharing Response (Type 1, Primitive 3)



Terminal I/O Request



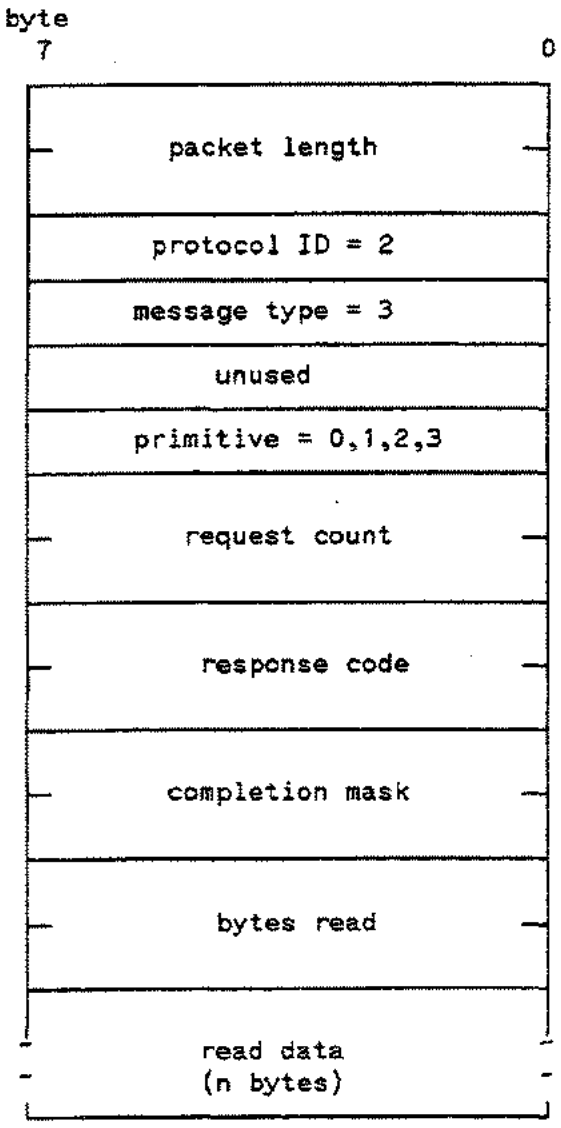
i/o primitive numbers

bit number	code
0	read
1	write
2	write_read
4	abort

read_mask	bit number	code
	0	type_ahead
	1	prompt
	2	cr_lf_off
	3	bypass type_ahead buffer
	4	flush type_ahead buffer
	5-15	reserved

write_mask	bit number	code
	0	cc_in_data
	1	prespace
	2	prompt
	3	needs_rsp
	4	premptive
	5	clr_flush
	5-15	reserved

Terminal I/O Reply



i/o primitive numbers

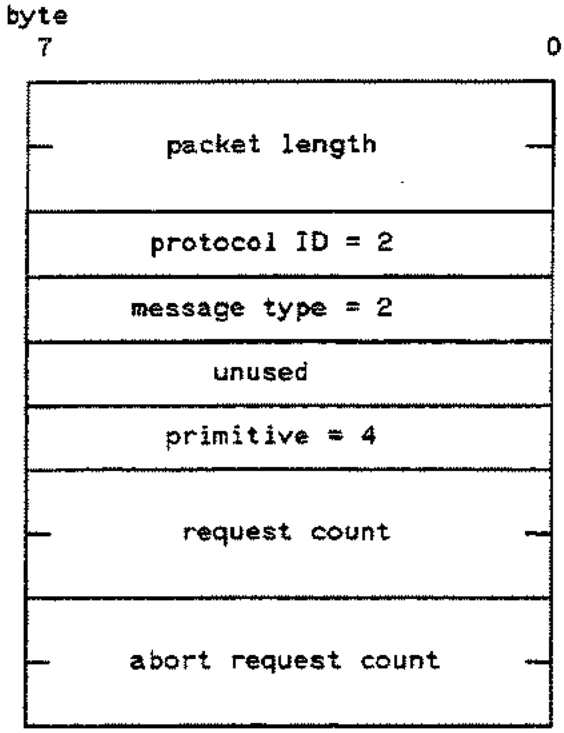
```
0 = read
1 = write
2 = write_read
4 = abort
```

io_completion

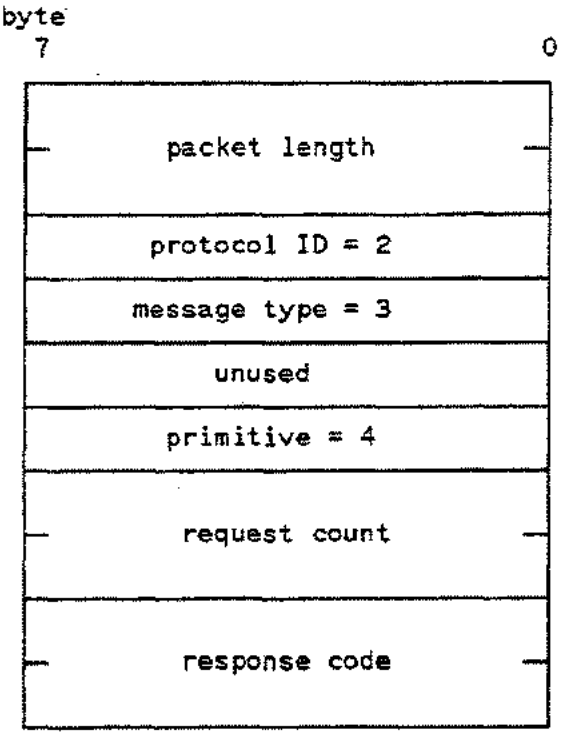
bit number	code
00	io_successful
01	io_eof
02	io_eof_error
03	io_bad_op
04	io_timeout
05	io_aborted
06	io_data_lost
07	io_parity_err

```
08      io_dev_unavail
09      io_unit_fail
10      io_buff_short
12      io_break_read
13      io_eor_read
14      io_extra02
15      io_extra01
```

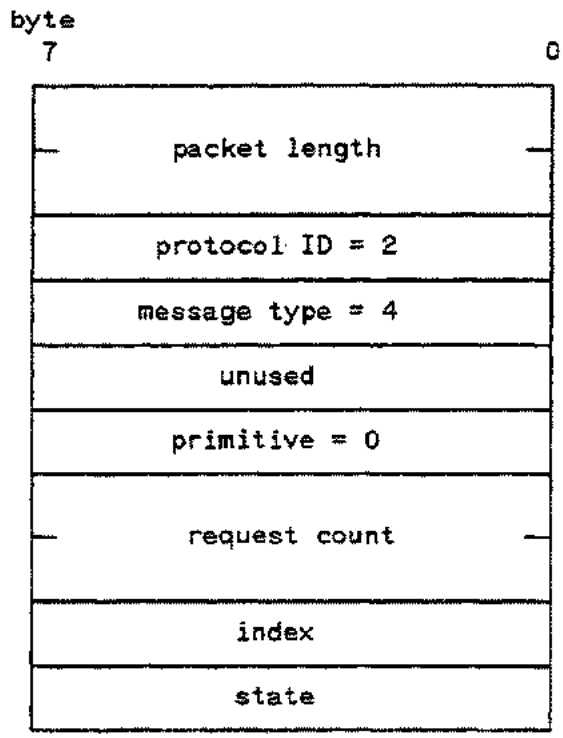
Abort I/O Request
(Type 2, Primitive 4)



Abort I/O Response
(Type 3, Primitive 4)



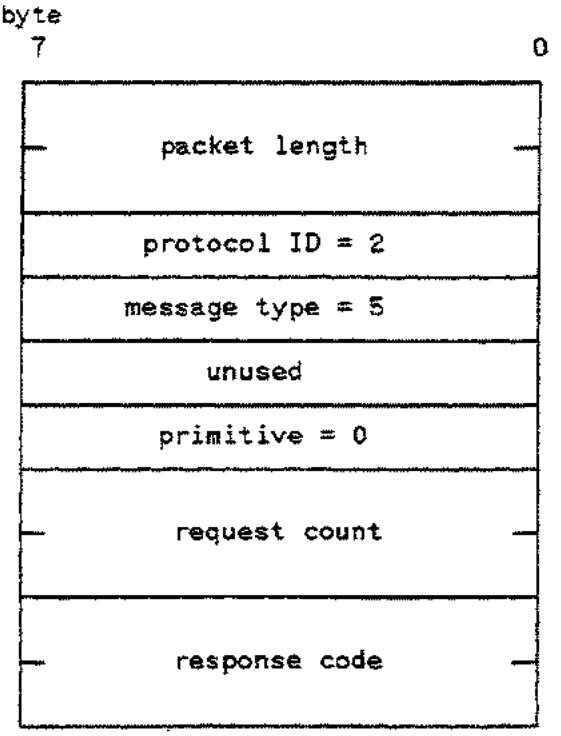
Set Break Request
(Type 4, Primitive 0)



index
0 = subsystem break
1 = system break

state
0 = off
1 = on

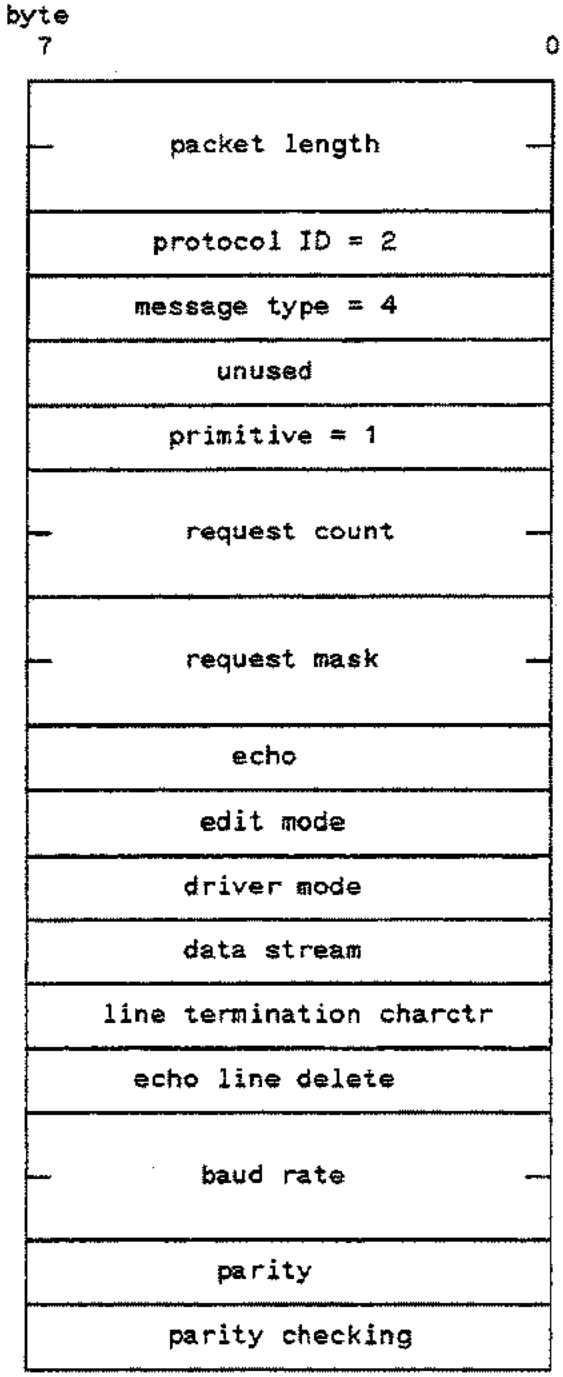
Set Break Response
(Type 5, Primitive 0)



response code 0 = successful

Terminal Driver Control Request

(Type 4, Primitive 1)



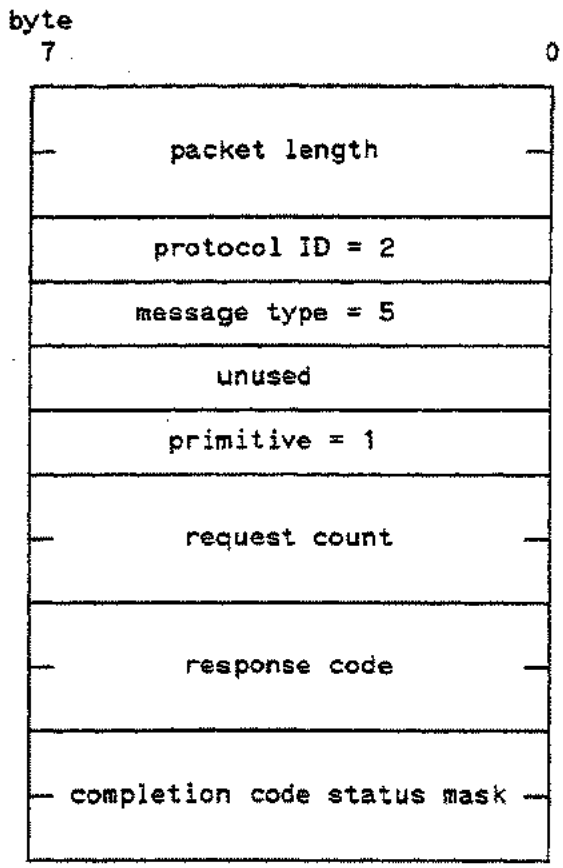
request mask

bit number	code
0	echo
1	e_mode
2	d_mode

```
                    3   term_char
                    4   data_strm
                    5   echo_line
                    6   baud_rate
                    7   parity
                    8   parity_sw

edit mode           1 = edit
                    2 = unedit
                    3 = binary
                    4 = disable_binary

driver mode         1 = "vanilla"
                    2 = block
                    3 = user block

echo                1 = echo_off_all
                    2 = echo_off_next
                    3 = echo_on_all
                    4 = echo_on_next
```

Set Driver Control Response

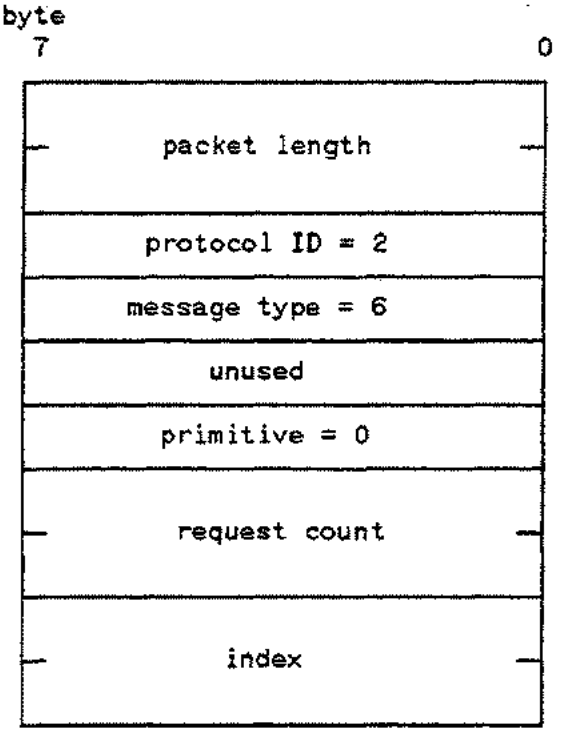
(Type 5, Primitive 1)



completion code status mask

bit number	code
00	successful
01	echo_fail
02	e_mode_fail
03	d_mode_fail
04	ln_trm_fail
05	stream_fail
06	echo_ld_fail
07	baud_fail
08	parity_fail
09	prtysw_fail

Invoke Break Request
(Type 6, Primitive 0)

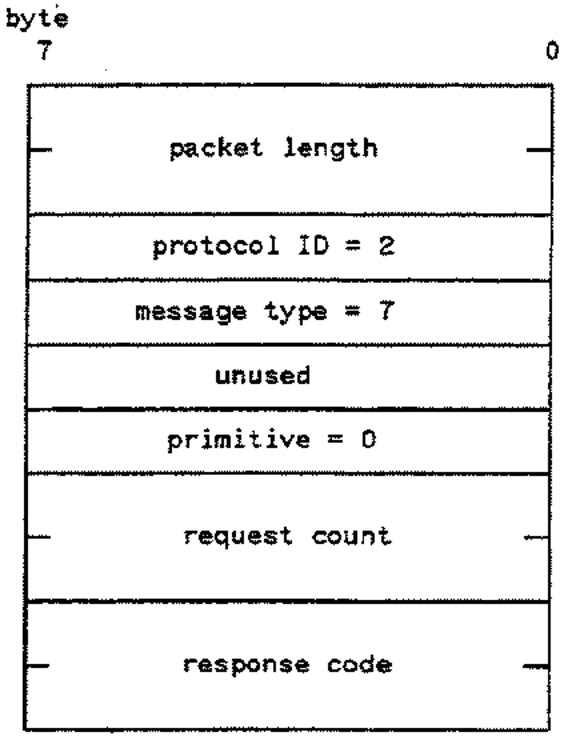


index

0 = subsystem break
1 = system break

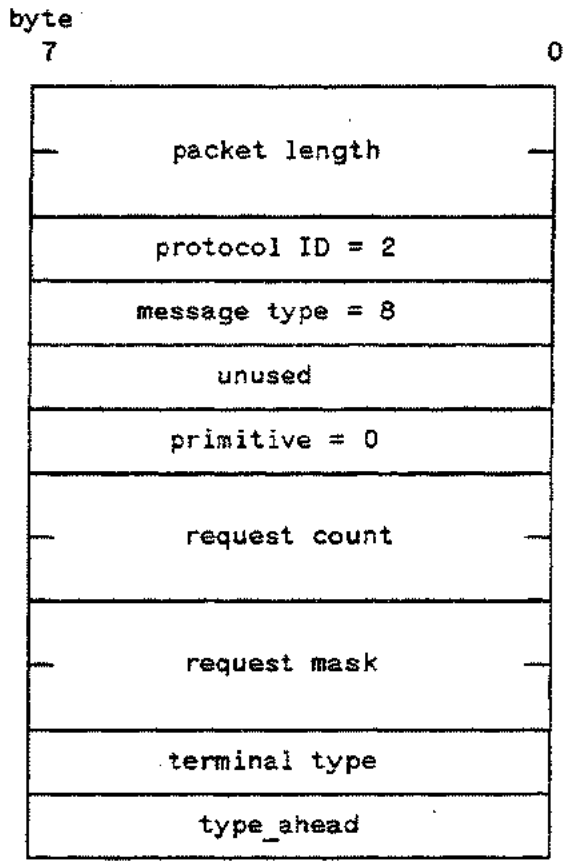
Invoke Break Response

(Type 7, Primitive 0)



MPE Specific Control Request

(Type 8, Primitive 0)



primitive

0 = hpe_driver_set
1 = hpe_get_info

request mask

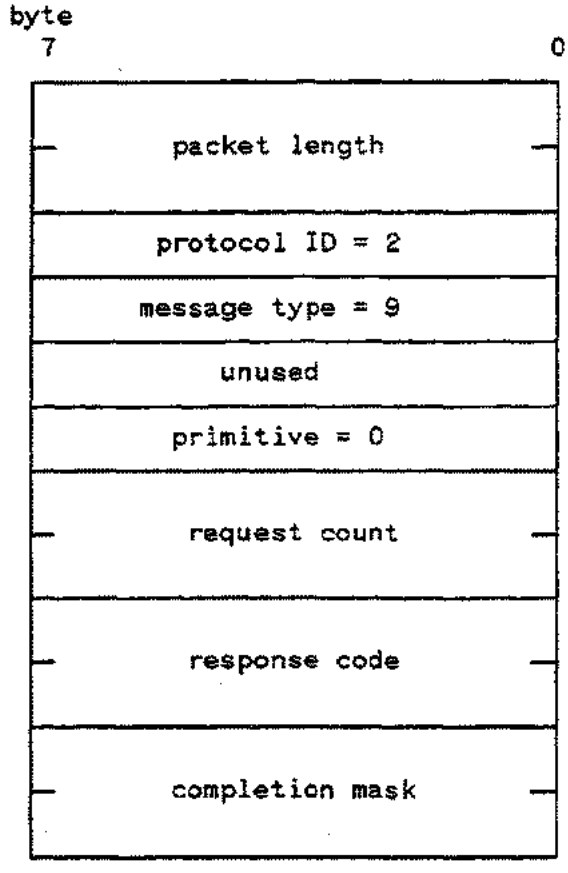
bit number	code
0	set_t_type
1	do_no_op
2	clr_flush
3	type_ahead

type_ahead

0 = diable type_ahead
1 = enable type_ahead

MPE Specific Control Response

(Type 9, Primitive 0)

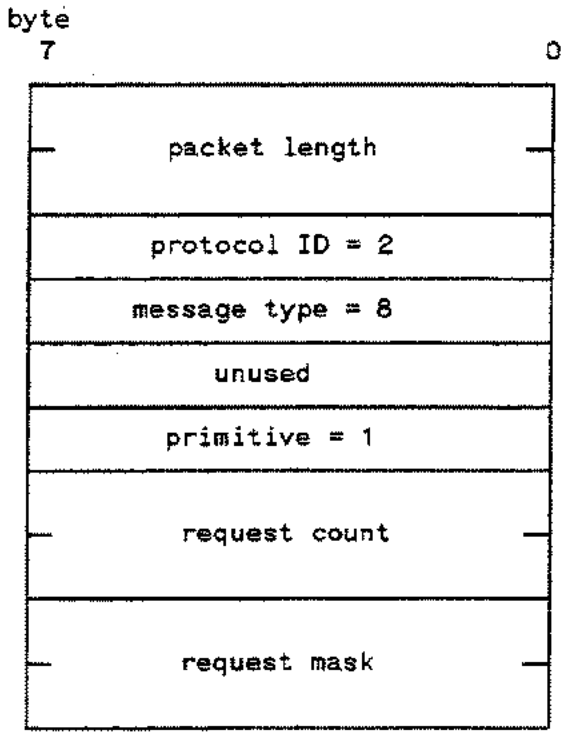


completion mask

bit number	code
0	successful
1	t_type_fail
2	no_op_fail

MPE Get Information Request

(Type 8, Primitive 1)

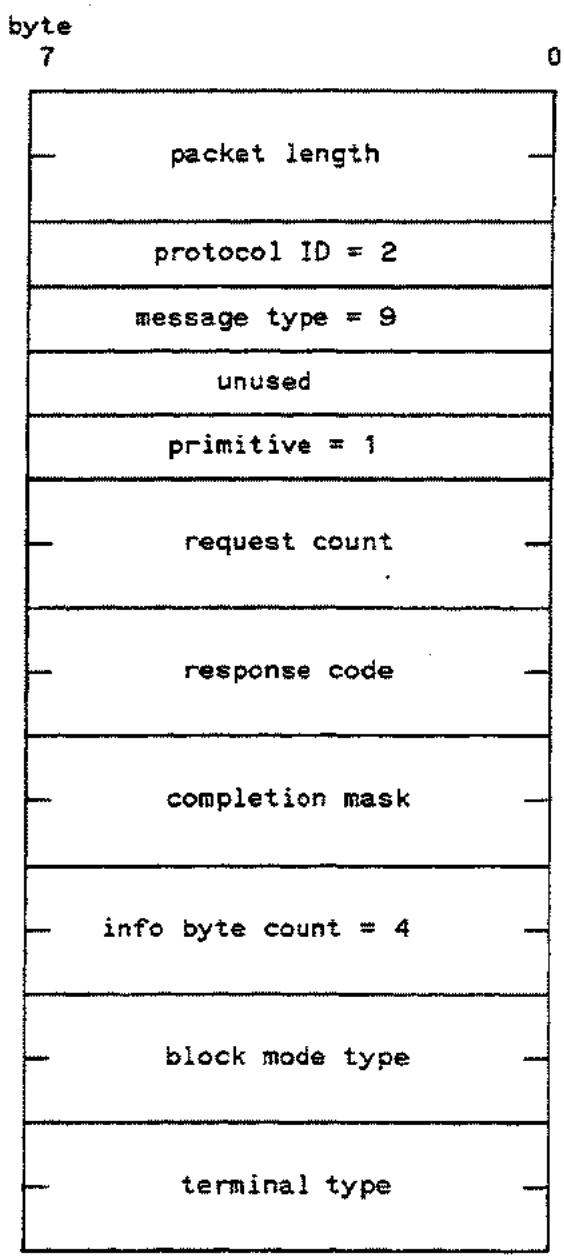


request mask

bit number	code
0	block_mode
1	get_t_type

MPE Get Information Response

(Type 9, Primitive 1)



completion mask

bit number	code
0	successful
1	bock_mode_fail
2	get_t_type_fail

block mode type

0 = no_block
1 = line_block
2 = ATP_page_block

```
  3 = ATP_page_line
  6 = AVESTA_page_block
  7 = AVESTA_page_line
 15 = PAD_terminal
```

RTE-RTE DS/1000-IV COMPATIBLE TRANSPORT AND SERVICES HEADERS

SECTION 7

User-initiated DS/1000-IV requests and commands are passed through RTE systems as class I/O buffers that are divided into two parts: A part containing user data, if any, and a part containing the Router/1000 header.

This chapter illustrates the Router/1000 headers that accompany user data. The Router/1000 header will be appended to messages generated by the following:

- NS Common Services transmitted over a Router/1000 link from an NS-ARPA/1000 system.
- DS/1000-IV Compatible Services transmitted over any other link type from an NS-ARPA/1000 system.

The first section of this chapter describes the format of the Router/1000 header in detail, particularly the meanings of fields in a standard "fixed header area."

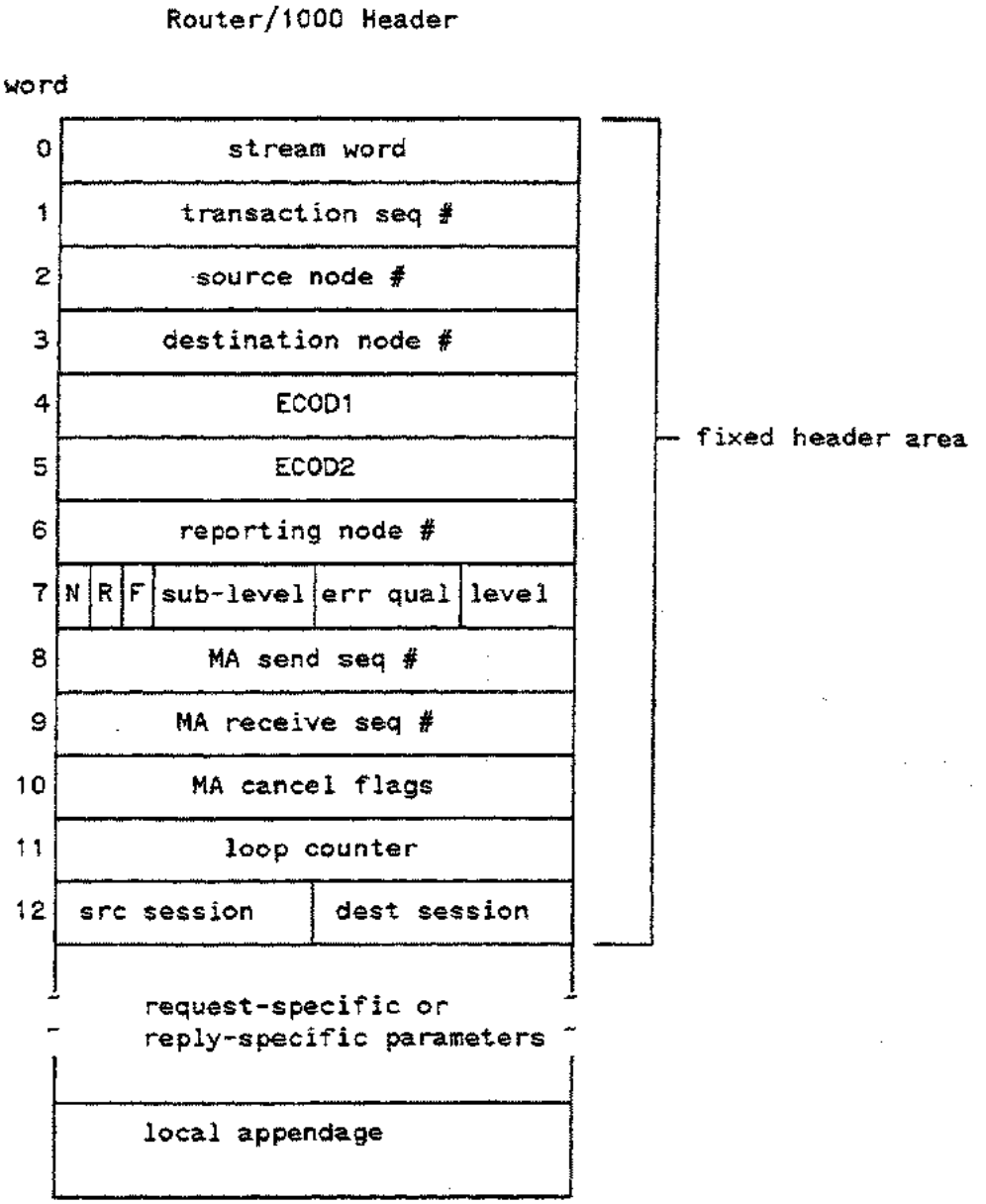
The second section contains illustrations of individual headers and messages generated by DS/1000-IV Compatible Services. These include DS/1000-IV transport messages and DS/1000-IV services messages.

ROUTER/1000 HEADER

The Router/1000 header consist of three parts:

- A thirteen-word fixed header
- A variable-length area containing parameters specific to the request or reply
- An area called the local appendage that contains information used by the local system. Because this information has no meaning to other network nodes, the local appendage is never transmitted. Although the local appendage is described early in this section, it is not repeatedly pictured with individual header illustrations.

The following illustration shows the format of the Router/1000 header:

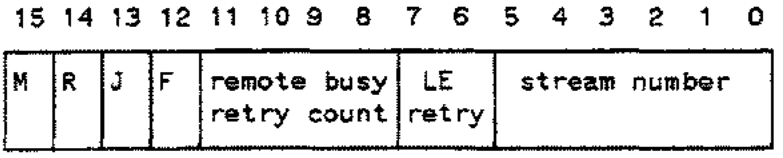


The remainder of this subsection describes the individual fields comprising the fixed header area and the local appendage. The contents of the request- or reply-specific parameter area are described for each command or intrinsic in the "DS/1000-IV Services" section of this chapter.

Fixed Header Area

Stream Word

The stream word (word 0) has the following format:



M — HP 3000 master bit. This bit is set if the message is a request from an HP 3000 or is a reply to an HP 3000 request.

R — Reply bit. This bit is set to indicate a reply.

J — Reject bit. This bit is set if the request or reply has been rejected because a remote busy condition has been encountered. A remote busy condition can be caused by any of the following situations:

- o No SAM available
- o No TCBs available
- o QUEUE is busy
- o Remote node is quiescent

F — Format level bit. Message level is Level 1 (DS/1000-IV or NS-ARPA/1000) if set; Level 0 (DS/1000) if cleared.

remote busy retry count

The value in this field represents the number of times a message is rejected because of a remote busy condition. The initial value of the field is the complement of the value specified in reply to the DSMOD \T menu prompt:

REMOTE-BUSY RETRIES [1 to 10]?

The initial value is then incremented each time a message is rejected because of a remote busy condition. When the count reaches 1111 binary, GRPM abandons retry attempts.

LE retry

Line error retry count. This field is used by the DVA65 driver (DS/1000-IV) only. (Note: The DVA65 driver is not supported by NS-ARPA/1000. It is supported by M-, E-, or F-Series nodes only. Initially zero, this count is incremented each time the driver reports an irrecoverable parity error upon transmission. The driver makes seven retry attempts before reporting an error; therefore, 21 retries are possible.

stream number

The contents of the stream field indicate the type of request made. Refer to Table 2-1 for the correspondence between the request types and stream numbers for headers shown in this section. Possible other stream numbers are the following:

Stream Number (octal)	Type of Request
0	Message Accounting or Router/1000 message, handled by GRPM
1	DLIST (Directory List/Cartridge List)
2	CNSLM ($STDLIST)
3	DEXEC (EXECW) (Remote Schedule with Wait)
4	PTOP (Program to Program Communications)
5	EXECM (Remote Exec Requests)
6	RFAM (Remote File Access)
7	OPERM (Operator Request)
11*	PROGL (System Cold Load Requests)
12	RDBAM (Remote Database Access)
13*	APLDX (APLDR for memory-only RTE-L)
14	TRFAS (Transparent File Access Server)
15	INPRO (Request for NS transport)

* The PROGL and APLDX request message headers are not shown in this chapter because they cannot be examined. Because QUEUE sends download requests directly to PROGL, these requests are never logged by NSTRC and will not appear in traces.

Transaction Sequence Number

The transaction sequence number is a number assigned by #RSAX to identify a unique request. This number is used to match the returning reply to the waiting master requestor.

Source Node Number and Destination Node Number

The source and destination node number fields (word 2 and word 3, respectively) contain the Router/1000 addresses of either the node that originated the message (source node number) or the node that is to receive the message (destination node number).

GRPM uses the reply bit in the stream word to determine the node to which messages should be forwarded. If the reply bit is set, the message is a reply, so the message is sent to the node identified by the source node number. If the reply bit is cleared, the message is an outbound message, so it is sent to the node identified by the destination node number.

The destination node number will always be non-negative; if the original master call specifies a negative LU, #MAST converts it to a node number.

Error Reporting

Words 4, 5, and 6 of each Router/1000 header indicate errors, if any, that occur when a message is passed between nodes. Errors reported in this way may have occurred at the sending node or during transmission. Note that if the node reporting the error (indicated by the value in word 6, the reporting node number field) is the same as the destination node (indicated by the value in word 3), it means that that although the message was delivered, it may not have been executable.

The ECOD1 and ECOD2 fields (word 4 and word 5, respectively) are error code words. The values in these fields indicate the error that occurred. For details on the meaning of a returned error code, refer to the *NS-ARPA/1000 Error Message and Recovery Manual.*

ECOD1. ECOD1 contains an ASCII value that indicates the type of error that occurred, as follows:

- DS indicates a DS/1000-IV Compatible Services error
- IO indicates an I/O error
- FM indicates an error returned by the File Manager
- RF indicates a Remote File Access error
- RS indicates a remote session error
- SC indicates a scheduling error
- SM indicates a Session Monitor error

ECOD2. ECOD2 can contain an ASCII value or a numeric value; this value indicates the error number. If the contents of ECOD2 are ASCII, the most significant bit (the A-bit) of the following word (the reporting node number word, word 6) is set.

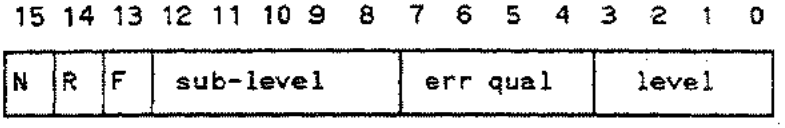
Note that it is up to master routines to convert the error codes in ECOD1 and ECOD2 to a form expected by the application program. The error words can be obtained by using the utility call DSERR.

Reporting Node Number. The reporting node number field (word 6) indicates the node that reported the error.

Qualifier and Message Level

Word 7 of the fixed header shows the software upgrade level and the error qualifier. Error qualifiers differentiate errors having the same error code.

This word has the following format:



N	Indicates a no-reply request, if set In reply messages, this bit is reserved (R)
R	Reserved
F	Flag in replies to indicate that the sub-level is valid
sub-level	Upgrade level within level
err qual	Error qualifier (meaningful only if an error is reported)
level	Software upgrade level

The software upgrade level of the local node is determined by #MAST (for requests) or #SLAV (for replies) from the #LEVL entry point in RESA. The level number in an incoming message is compared with the local node's level number to determine if message format conversion is required. Conversion is required when a message is sent to or received from a node with a lower software upgrade level.

The sub-level number is used when minor changes in protocols are made that do not require message conversion. The sub-level field is valid only when the F bit is set to 1. Messages from nodes with sub-levels of 1 or greater have the F bit and the sub-level field set to 1. Slave replies from nodes with sub-level 0 echo of the sub-level field of the master request. Therefore, a slave reply from a node with sub-level 0 may have a non-zero value in the sub-level field, but the F bit will be set to 0, and the sub-level field would be invalid.

Message Accounting Variables

The MA send seq # (word 8), MA receive seq # (word 9) and MA cancel flags (word 10) are used by Message Accounting software. Refer to *NS-ARPA/1000 Maintenance and Principles of Operation Manual* for more information.

Loop Counter

The loop counter field (word 11) initially contains the two's complement of the Loop Counter. The Loop Counter equals the total number of nodes in the nodal routing vector. The loop counter value in word 11 is incremented each time the message passes through a node via a Router/1000 link.

Session ID Word

The session ID word (word 12) contains src session and dest session, which are the session identifier words for the local and remote sessions.

Parameter-Specific Area

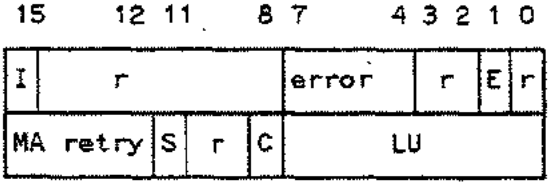
The contents of fields generated by specific DS/1000-IV Service commands and calls are described in the "DS/1000-IV Services" section later in this chapter.

Local Appendage

The Local Appendage contains information needed to process requests in the local node but is meaningless to the remote node, and is not transmitted.

The Local Appendage has the following format:

Local Appendage



Fields labeled r are reserved.

The I bit, if set, indicates an IFP message.

The error is the error code of the error reported by IFPM. The only currently possible code is 1, indicating a line failure. If the error code=0, no error occurred.

The E bit is set if an error occurred. If the error code (error)=0, then E=0.

The MA retry is the number of times Message Accounting has retransmitted an unacknowledged message.

The S bit (MA assignment bit) indicates that the local node's Message Accounting software has assigned valid sequence numbers.

The C bit is 0 if the message was received from a communications link driver; the I/O status contained in the A-register after returning from the class I/O get call is valid. The C bit is 1 if the message was received after being re-queued; the I/O status is ignored.

The LU field is set by the driver (when C is set to 0) to indicate the LU from which the message has been received.

For DS/1000-IV services over Router/1000 links, the first word of the local appendage is *not* meaningful. For DS/1000-IV services over any other link type, the local appendage's first word is meaningful, but only the MA retry, S, r, and C fields (or bits) of the second word are used. The LU field is always set to 0 in this case (for DS/1000-IV services over non-Router/1000 links).

DS/1000-IV COMPATIBLE HEADERS AND MESSAGES

Transport Headers and Messages

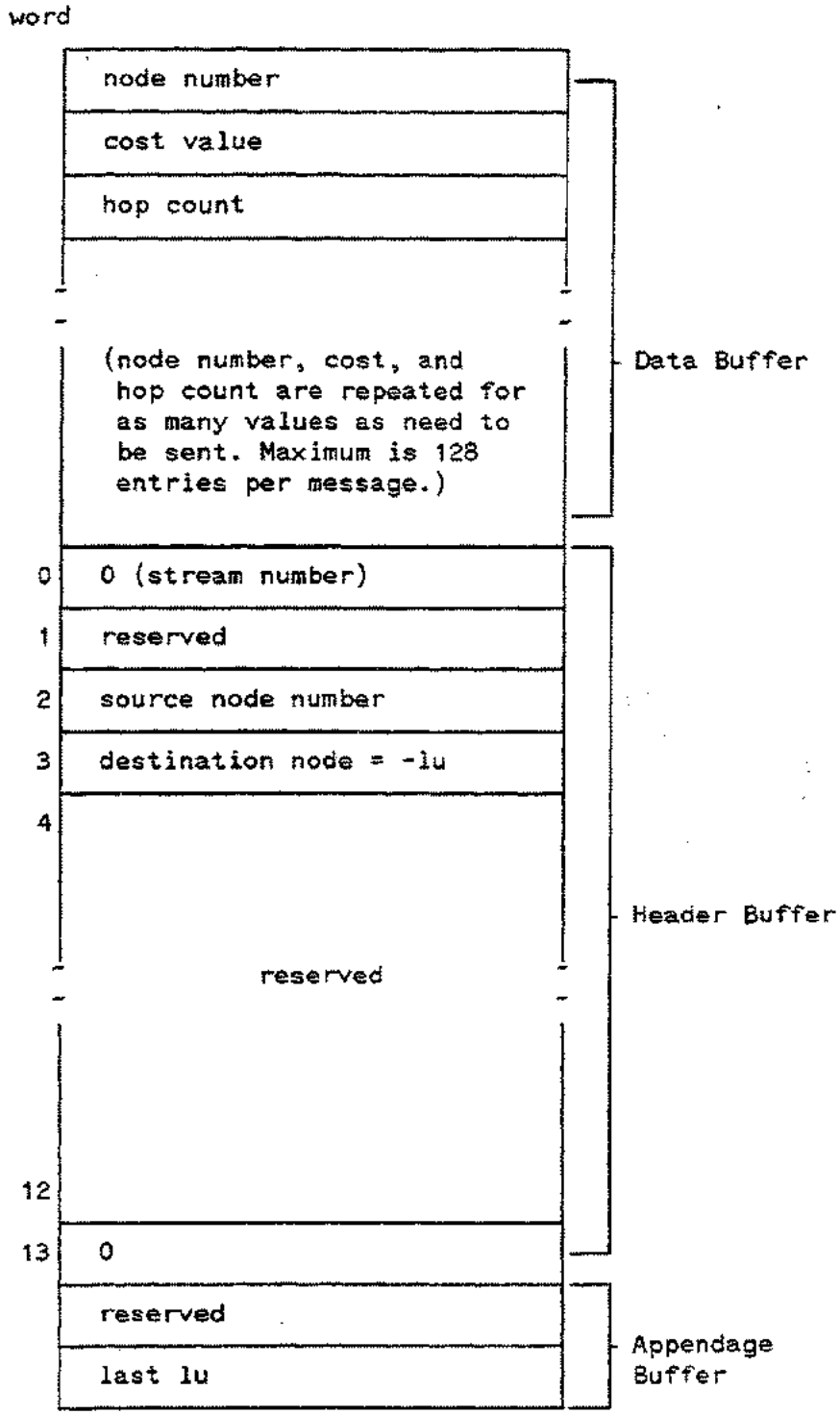
This section contains illustrations of headers for the following types of messages:

- Dynamic Rerouting Update Messages
- Message Accounting Update Messages

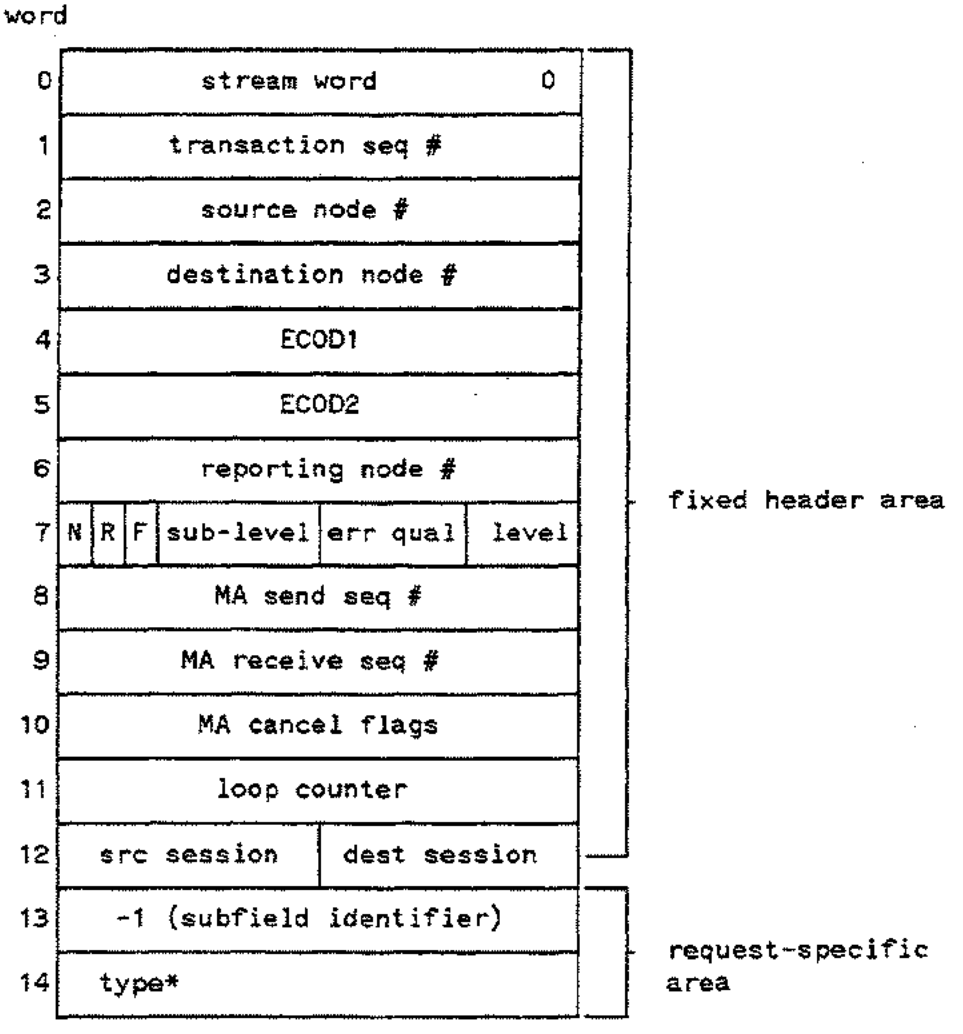
Dynamic Rerouting Update Messages are sent when Router/1000 links are used with Dynamic Rerouting.

Message Accounting Update Messages are sent when DS/1000-IV Compatible Services (RTE-RTE) are used with Message Accounting.

Dynamic Rerouting Update Message



Message Accounting Messages



MA (Message Accounting) Message Types (word 14)

type number (octal)	name
1	(Receiver Ready)
2	(Initialize Request)
4	(Initialize Response)
10	(Cancel Outstanding Message)
20	(No Response)

* Word 14, the type field, is optional. If word 14 is not present, the message is an MA channel down message and is used only within the local node to notify master programs of an error condition. In this case, the local appendage will not be present.

DS/1000-IV Services

The remainder of this chapter illustrates the Router/1000 headers for individual DS/1000-IV Services commands and calls.

Each header diagram shows both the Fixed Format Area and the parameter-specific areas of each call or command.

The names for parameter-specific fields correspond to parameter names defined in the *NS-ARPA/1000 User/Programmer DS/1000 Compatible Services Reference Manual.*

Table 7-1. DS/1000-IV Message Summary

Service Type	Call or Command	Stream Number (octal)
DEXEC	1 (Read)	5
	2 (Write)	5
	3 (Remote I/O control)	5
	6 (Program termination)	5
	9 (Immediate schedule, wait)	3
	10 (Immediate schedule, no wait)	5
	11 (Time request)	5
	12 (Timed program schedule)	5
	13 (I/O status)	5
	23 (Queue schedule, wait)	3
	24 (Queue schedule, no wait)	3
	25 (Partition status)	5
	99 (Program status)	
PTOP	ACEPT	4
	FINIS	4
	GET	4
	PCLOS	4
	PCONT	4
	PNRPY	4
	POPEN	4
	PREAD	4
	PWRIT	4
	REJCT	4
RDBA	all RDBA messages	10
RFA	DAPOS/DXPO	6
	DCLOS/DXCLO	6
	DCONT	6
	DCRET/DXCRE	6
	DLOCF/DXLOC	6
	DNAME	6
	DOPEN	6
	DPOSN/DXPOS	6
	DPURG	6
	DREAD/DXREA	6
	DSTAT	6
	DWIND	6
	DWRIT/DXWRI	6

Table 7-1. DS/1000-IV Message Summary (Continued)

Service Type	Call or Command	Stream Number (octal)
Utilities	DLGNS	7
	DLGOF	7
	DLGON	7
	DMESG	5
	DMESS	7
	DNODE	5
	<DSERR>	none
	FLOAD	3
	GNODE	none
	SEGLD	none
	WHAT	3
Remote I/O Mapping	all remote messages	5
TRANSPARENT FILE ACCESS (DS TRANSPARENCY)	all transparent file access messages	14
REMAT	AT	7
	BC	5
	CL	1
	CR	6
	DE	7
	DL	1
	DU	5
	FL	6
	IO	5
	LI	5
	LO	3
	PL	3
	PU	6
	QU	3
	QW	3
	RN	6
	RW	3
	SD	7
	SL	4
	SO	4
	ST	6
	SW	7
	TE	5
	TR	6

Transparent File Access Server (TRFAS) Header

REQUEST

word						
0	stream word					14
1	transaction sequence number					
2	source node #					
3	destination node #					
4	ECOD1					
5	ECOD2					
6	reporting node #					
7	N	R	F	sub-level	qual	level
8	MA send seq #					
9	MA receive seq #					
10	MA cancel flags					
11	loop counter					
12	src sessn			dst sessn		
13	request-specific parameters					
14						
15						
16						
17						
18	session ID					

REPLY

word						
0	stream word					14
1	transaction sequence number					
2	source node #					
3	destination node #					
4	ECOD1					
5	ECOD2					
6	A	reporting node #				
7	R	R	F	sub-level	qual	level
8	MA send seq #					
9	MA receive seq #					
10	MA cancel flags					
11	loop counter					
12	src sessn			dst sessn		
13	error					
14	A-register on exec error					
15	B-register on exec error					
16	password					
17	not used					

request-specific parameters (words 13 through 17) are described on the following pages. Each request type has a specific code associated with it, as shown in the following table.

TRFAS Request Types and Codes

Request Type	Code
disk read	1
disk write	2
logon	3
logoff	4
close file	5
disc info	6
fstat	7
password	8
D.RTR request	-1

Request-Specific Parameters

Disk Read (request code 1)
and Disk Write (request code 2)

13	password	
14	request code	lu
15	length	
16	track	
17	sector	

Logon (request code 3) and
Logoff (request code 4)

13	reserved	
14	request code	0
15	session ID	
16	reserved	
17	reserved	

Close File

13	reserved	
14	5 (request code)	0
15	reserved	
16	reserved	
17	reserved	

Disc Info

13	reserved	
14	6 (request code)	0
15	lu	
16	reserved	
17	reserved	

Fstat

13	reserved	
14	7 (request code)	0
15	format code	
16	option word	
17	reserved	

Password

13	reserved	
14	8 (request code)	0
15	reserved	
16	reserved	
17	reserved	

D.RTR Function

13	31416
14	1
15	code
16	0
17	0

DLIST

REQUEST

word							
0	stream word						01
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	maximum line length						
14	0	(new request flag)					
15	name filter						
16							
17							
18	master security code						
19	crn						
20	type						

REPLY

word							
0	stream word						01
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	status						
14	actual line length						
15	(brout=coroutine address)						
16	name filter						
17							
18							
19	master security code						
20	crn						
21	type						

(continued on next page)

(DLIST REPLY, continued)

22	current disc lu
23	current disc track
24	current sector
25	displacement in buffer
26	# sectors/track
27	displacement in directory lu
28	# directory tracks

Current disc lu through # directory tracks area of reply (words 22 through 28) is stored in request message, after first time. In this manner, DLIST is able to continue listings from multiple requesters.

Status (word 13 of reply) is 0 when done.

POPEN

REQUEST

word							
0	stream word						04
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	ff	c	reserved		1 (fcode)		
14	program name						
15							
16							
17	tag [1]						
18	tag [2]						
19	tag [3]						
~	tag [4] through tag [18]						
35	tag [19]						
36	tag [20]						

REPLY

word							
0	stream word						04
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	ff	reserved			1 (fcode)		
14	slave program id						
15	slave I/O class#						
16	reserved						
17	tag [1]						
18	tag [2]						
19	tag [3]						
~	tag [4] through tag [18]						
35	tag [19]						
36	tag [20]						

Program name is the 5-character program name followed by an ASCII blank.

PREAD

REQUEST

word							
0	stream word						04
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	ff	reserved			2 (fcode)		
14	program ID						
15	I/O class #						
16	len						
17	tag [1]						
18	tag [2]						
19	tag [3]						
~	tag [4] through tag [18]						
35	tag [19]						
36	tag [20]						

REPLY

word							
0	stream word						04
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	ff	reserved			2 (fcode)		
14	slave program ID						
15	slave I/O class#						
16	reserved						
17	tag [1]						
18	tag [2]						
19	tag [3]						
~	tag [4] through tag [18]						
35	tag [19]						
36	tag [20]						

PWRIT

REQUEST

word							
0	stream word						04
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	ff	reserved			3 (fcode)		
14	program ID						
15	I/O class #						
16	len						
17	tag [1]						
18	tag [2]						
19	tag [3]						
~	tag [4] through tag [18]						
35	tag [19]						
36	tag [20]						

REPLY

word							
0	stream word						04
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	ff	reserved			3 (fcode)		
14	slave program ID						
15	slave I/O class #						
16	reserved						
17	tag [1]						
18	tag [2]						
19	tag [3]						
~	tag [4] through tag [18]						
35	tag [19]						
36	tag [20]						

PCONT

REQUEST

word							
0	stream word						04
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	ff	reserved		4 (fcode)			
14	program ID						
15	I/O class#						
16	reserved						
17	tag [1]						
18	tag [2]						
19	tag [3]						
~	tag [4] through tag [18]						
35	tag [19]						
36	tag [20]						

REPLY

word							
0	stream word						04
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	ff	reserved		4 (fcode)			
14	slave program ID						
15	slave program I/O class#						
16	reserved						
17	tag [1]						
18	tag [2]						
19	tag [3]						
~	tag [4] through tag [18]						
35	tag [19]						
36	tag [20]						

PCLOS

REQUEST

word							
0	stream word						04
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	ff	reserved			5 (fcode)		
14	program ID						
15	I/O class #						

REPLY

word							
0	040004						
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	ff	reserved			5 (fcode)		
14	slave program ID						
15	slave program I/O class #						

FINIS

REQUEST

word							
0	stream word						04
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	ff	reserved			1	5 (fcode)	
14	program ID						
15	I/O class #						

REPLY

word							
0	stream word						04
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	ff	reserved			1	5 (fcode)	
14	slave program ID						
15	slave program I/O class #						

Bit 7 in the FCODE word is set to indicate FINIS.

SLAVE OFF (REMAT)

REQUEST

word							
0	stream word						04
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	ff	reserved			6 (fcode)		
14	program name						
15							
16							

REPLY

word							
0	stream word						04
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	ff	reserved			6 (fcode)		

SLAVE LIST (REMAT)

REQUEST

word							
0	stream word						04
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	ff	reserved			7 (fcode)		

REPLY

word							
0	stream word						04
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	ff	reserved			7 (fcode)		

DEXEC 1 (Remote Read)

REQUEST

word							
0	stream word						05
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	1 (code)						
14	cnwd [1]						
15	cnwd [2]						
16	buf1						
17	prm1						
18	prm2						
19	0 (place holder)						
20	keywd						

REPLY

word							
0	stream word						05
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		

DEXEC 2 (Remote Write)

REQUEST

word							
0	stream word						05
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	2 (code)						
14	cnwd [1]						
15	cnwd [2]						
16	buf1						
17	prm1						
18	prm2						
19	0 (place holder)						
20	keywd						

REPLY

word							
0	stream word						05
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			

I/O status is returned in ECOD1 and to the called in the A-register.
Transmission log is returned in ECOD2 and to the caller in the B-Register.

DEXEC 3 (Remote I/O Control)

REQUEST

word							
0	stream word						05
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	3 (code)						
14	cnwd [1]						
15	cnwd [2]						
16	p1						
17	p3						
18	p4						
19	p2						
20	keywd						

REPLY

word							
0	stream word						05
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		

I/O status is returned in ECOD1 and to the caller in the A-Register. The contents of B-register ECOD2 are meaningless.

DEXEC 13 (Remote I/O Status)

REQUEST

word							
0	stream word						05
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	15 (code)						
14	cnwd [1]						
15	cnwd [2]						
16	0						
17	place holder						
18	0/(z-buffer length)						
19	place holder						
20	0						

REPLY

word							
0	stream word						05
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	sta1						
14	sta2						
15	sta3						
16	sta4						

A- and B-Registers (ECOD1/ECOD2) contain negative word count of z-buffer status information when the status request is executed in an RTE-L node.

DEXEC 10 (Immediate Schedule, No Wait)

REQUEST

word							
0	stream word						05
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	12 (code)						
14	program name						
15							
16							
17	prm1						
18	prm2						
19	prm3						
20	prm4						
21	prm5						
22	bufr						
23	bufl						

REPLY

word							
0	stream word						05
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			

ECOD1 contains program status, which is returned to the caller in the A-Register.

Program name is the 5-character program name followed by an ASCII blank.

DEXEC 11 (Remote Time Request)

REQUEST

word							
0	stream word						05
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	13 (code)						
14	place holders for reply						
15							
16							
17							
18							

REPLY

word							
0	stream word						05
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	time [1]						
14	time [2]						
15	time [3]						
16	time [4]						
17	time [5]						
18	year						

DEXEC 12 (Remote Timed Program Schedule)

REQUEST

word							
0	stream word						05
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	14 (code)						
14	program name						
15							
16							
17	resl						
18	mtple						
19	ofst/hrs						
20	mins						
21	secs						
22	msecs						

REPLY

word							
0	stream word						05
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			

Program name is the 5-character program followed by an ASCII blank.

DEXEC 25 (Remote Partition Status)

REQUEST

word						
0	stream word					05
1	transaction seq #					
2	source node #					
3	destination node #					
4	ECOD1					
5	ECOD2					
6	reporting node #					
7	N	R	F	sub-level	err qual	level
8	MA send seq #					
9	MA receive seq #					
10	MA cancel flags					
11	loop counter					
12	src session			dest session		
13	31 (code)					
14	part					
15	place holders for reply					
16						
17						
18						

REPLY

word						
0	stream word					05
1	transaction seq #					
2	source node #					
3	destination node #					
4	ECOD1					
5	ECOD2					
6	A	reporting node #				
7	R	R	F	sub-level	err qual	level
8	MA send seq #					
9	MA receive seq #					
10	MA cancel flags					
11	loop counter					
12	src session			dest session		
13	page					
14	pnum					
15	stat					

DEXEC 99 (Remote Program Status)

REQUEST

word							
0	stream word						05
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	143 (code)						
14	program name						
15							
16							
17	stat						

REPLY

word							
0	stream word						05
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	stat						

ECOD1 contains program which is returned to the caller in the A-Register. ECOD2 = 0, which is returned to the caller in the B-Register. If an error occurred ECOD1/A (register A) contains the ASCII characters DS, and ECOD2 (register B) contains numeric -1. The sign bit of ECOD3 = 0.

Program name is the 5-character program name followed by an ASCII blank.

DEXEC 6 (Remote Program Termination)

REQUEST

word						
0	stream word					05
1	transaction seq #					
2	source node #					
3	destination node #					
4	ECOD1					
5	ECOD2					
6	reporting node #					
7	N	R	F	sub-level	err qual	level
8	MA send seq #					
9	MA receive seq #					
10	MA cancel flags					
11	loop counter					
12	src session			dest session		
13	6 (code)					
14	program name					
15						
16						
17	numb					
18	op1					
19	op2					
20	op3					
21	op4					
22	op5					

REPLY

word						
0	stream * 4000					
1	transaction seq #					
2	source node #					
3	destination node #					
4	ECOD1					
5	ECOD2					
6	A	reporting node #				
7	R	R	F	sub-level	err qual	level
8	MA send seq #					
9	MA receive seq #					
10	MA cancel flags					
11	loop counter					
12	src session			dest session		

This request is originated on stream 5, but may be requeued onto stream 3 at the destination node if EXECM determines that the specified program is not EXECM's son.

Program name is the 5-character program name followed by an ASCII blank.

The value of op1 through op5 are meaningless when numb (word 17) is greater than 0.

DEXEC 9 (Immediate Schedule, Wait)

REQUEST

word							
0	stream word						03
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	11 (code)						
14	program name						
15							
16							
17	prm1						
18	prm2						
19	prm3						
20	prm4						
21	prm5						
22	bufr						
23	buf1 (place holder)						

REPLY

word							
0	stream word						03
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	prm1						
14	prm2						
15	prm3						
16	prm4						
17	prm5						

Program name is the 5-character program followed by an ASCII blank.

DEXEC 23 (Queue Schedule, Wait)

REQUEST

word							
0	stream word						03
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	27 (code)						
14	program name						
15							
16							
17	prm1						
18	prm2						
19	prm3						
20	prm4						
21	prm5						
22	bufr (place holder)						
23	bufl						

REPLY

word							
0	stream word						03
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	prm1						
14	prm2						
15	prm3						
16	prm4						
17	prm5						

Program name is the 5-character program name followed by an ASCII blank.

DEXEC 24 (Queue Schedule, No Wait)

REQUEST

word							
0	stream word						03
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	30 (code)						
14	program name						
15							
16							
17	prm1						
18	prm2						
19	prm3						
20	prm4						
21	prm5						
22	bufr						
23	buf1 (place holder)						

REPLY

word							
0	stream word						03
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			

Program name is the 5-character program name followed by an ASCII blank.

DPURG

REQUEST

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	000010						
14	file name						
15							
16							
17	cr[1]						
18	ID segment address						
19	secu						

REPLY

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		

Program name is the 5-character program name followed by an ASCII blank.

DOPEN

REQUEST

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	000006						
14	file name						
15							
16							
17	cr[1]						
18	ID segment address						
19	secu						
20	optn						

REPLY

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	entry number						

DWRIT

REQUEST

word						
0	stream word					06
1	transaction seq #					
2	source node #					
3	destination node #					
4	ECOD1					
5	ECOD2					
6	reporting node #					
7	N	R	F	sub-level	err qual	level
8	MA send seq #					
9	MA receive seq #					
10	MA cancel flags					
11	loop counter					
12	src session			dest session		
13	000014					
14	dcb[1]					
15	dcb[2]					
16	dcb[3]					
17	len					
18	num					

REPLY

word						
0	stream word					06
1	transaction seq #					
2	source node #					
3	destination node #					
4	ECOD1					
5	ECOD2					
6	A	reporting node #				
7	R	R	F	sub-level	err qual	level
8	MA send seq #					
9	MA receive seq #					
10	MA cancel flags					
11	loop counter					
12	src session			dest session		

DREAD

REQUEST

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	000011						
14	dcb[1]						
15	dcb[2]						
16	dcb[3]						
17	len						
18	num						

REPLY

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	len (transmission log)						

DPOSN

REQUEST

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	000007						
14	dcb[1]						
15	dcb[2]						
16	dcb[3]						
17	nur						
18	rec						

REPLY

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		

DWIND

REQUEST

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	000013						
14	dcb[1]						
15	dcb[2]						
16	dcb[3]						

REPLY

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		

DNAME

REQUEST

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	000005						
14	file name						
15							
16							
17	cr[1]						
18	id segment address						
19	secu						
20	nname						
21							
22							

REPLY

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			

DCONT

REQUEST

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	000002						
14	dcb[1]						
15	dcb[2]						
16	dcb[3]						
17	con1						
18	con2						

REPLY

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		

DLOCF

REQUEST

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	000004						
14	dcb[1]						
15	dcb[2]						
16	dcb[3]						

REPLY

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	rec						
14	rb						
15	off						
16	sec						
17	lu						
18	ty						
19	recsz						

DAPOS

REQUEST

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	000000						
14	dcb[1]						
15	dcb[2]						
16	dcb[3]						
17	rec						
18	rb						
19	off						

REPLY

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			

DSTAT

REQUEST

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	000012						
14	place holders for reply						
15							

REPLY

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			

DCLOS

REQUEST

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	000001						
14	dcb[1]						
15	dcb[2]						
16	dcb[3]						
17	trun						

REPLY

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		

DCRET

REQUEST

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	000003						
14	file name						
15							
16							
17	cr[1]						
18	ID segment address						
19	secu						
20	size[1]						
21	size[2]						
22	type						

REPLY

word							
0	stream word						06
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	entry number						

REMOTE OPERATOR REQUESTS

REQUEST

word							
0	stream word						07
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	command length (bytes)						
	command (ASCII)						

REPLY

word							
0	stream word						07
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	reply length (+words)						
	reply (ASCII)						

LOGON REQUEST

REQUEST

word							
0	stream word						07
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	src session				0		
14	(command length = 2)						
15	"X"				"X"		
16	1 (code)						
17	logon string length						
18							
	username, group name and password (up to 32 characters)						

REPLY

word							
0	stream word						07
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receove seq #						
10	MA cancel flags						
11	loop counter						
12	ECOD1						
13	src session				dest session		

LOGOF REQUEST

REQUEST

word							
0	stream word						07
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	src session			0			
14	2 (command length)						
15	"X"			"X"			
16	code=0 or -1						
17	session id						

REPLY

word							
0	stream word						07
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12							
13	src session			0			

code = -1 is used by UPLIN to indicate "no reply" to Remote Session Monitor.

NON-SESSION ACCESS REQUEST

REQUEST

word							
0	stream word						07
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session				dest session		
13	src session				0		
14	(command length = 2)						
15	"X"				"X"		
16	1 (code)						
17	password (up to 10 words)						

REPLY

word							
0	stream word						07
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12							
13	src session				376		

RDBA REQUEST

REQUEST

word							
0	stream word						12
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	reporting node #						
7	N	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	call index						
14	call mode						
15	call item or set #*						
16	search item # (DBFND)*						
17							
18	max. return RT*						
19	word size BASE parameter*						
20	filename (up to 10 words)*						
~	~						

REPLY

word							
0	stream word						12
1	transaction seq #						
2	source node #						
3	destination node #						
4	ECOD1						
5	ECOD2						
6	A	reporting node #					
7	R	R	F	sub-level	err qual	level	
8	MA send seq #						
9	MA receive seq #						
10	MA cancel flags						
11	loop counter						
12	src session			dest session			
13	RDBA call status array						
~	~						
22							
23	RDBA database # (DBOPN)						

*Note: Words 18 and beyond will be present only if the call is DBOPN, and if the call is DBOPN, words 15, 16, and 17 of the request are filled with a six character password.

RTE-MPE DS/1000-IV COMPATIBLE SERVICES MESSAGES

SECTION 8

The message formats contained in this section are those of DS/1000-IV Compatible Services messages passed between HP 1000s and HP 3000s (RTE-MPE or MPE-RTE).

Table 8-1 summarizes the messages included in this section. Message illustrations are arranged in the section in ascending numerical order according to their message class and stream type, as shown in the table.

Field labels correspond to parameter names described in the *NS-ARPA/1000 User/Programmer DS/1000 Compatible Services Reference Manual* and in the *MPE V Intrinsics Reference Manual* (for MPE file system intrinsics).

Table 8-1. HP 1000 - HP 3000 Message Summary

Message Class (decimal)	Stream Type (octal)	Meaning (Message Type, Command, or Intrinsic)
0	20 21	Initialization* Termination *
3	20 22	REMOTE command (except Class 6 commands) DSLINE
4	22 23 24 26 27	PREAD PWRITE PCONTROL ACCEPT (Reply to PREAD, PWRITE or PCONTROL) REJECT (Reply to PREAD, PWRITE or PCONTROL)
5	 20 21 23	$STDLIST to directed terminal $STDIN (READ/READX) (Read request against master terminal) FCONTROL request against master timeout (?)
6	20 21 22 23 24 25 27	Remote HELLO Remote BYE BREAK * ABORT PROGRAM * RESUME * CONTROL-Y * KILL JOB *
7	20 21 22 26 27	RFA calls to HP 3000s (FCHECK, FCLOSE, FCONTROL, FGETINFO, FLOCK, FOPEN, FPOINT, FREAD, FREADADDIR, FREADSEEK, FRELATE, FRENAME, FREADLABEL, FSETMODE, FUNLOCK, FUPDATE, FWRITEDIR, FWRITE, FWRITELABEL) POPEN or PCLOSE request DSLINE PTOP accept of POPEN PTOP reject of POPEN
8	20 21	RTE FMP RFA (RFA calls to HP 1000s) (DAPOS, DCLOS, DCONT, DCRET, DLOCF, DNAME, DOPEN, DPOSN, DPURG, DREAD, DSTAT, DWRIT) Remote EXEC calls (DEXEC I/O Control, DEXEC Write, DEXEC Execution Time, DEXEC Schedule, DEXEC Time, DEXEC I/O status, DEXEC Read.)

Message types marked by an asterisk (*) are not individually pictured later in this section because they each consist of the eight-word header only.

The first eight words (words 0 through 7) of each HP 1000 - HP 3000 message have a fixed format. This format is shown below:

Fixed Header for Request or Reply

0	length1	message class
1	reserved	
2	stream word	
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	length2	

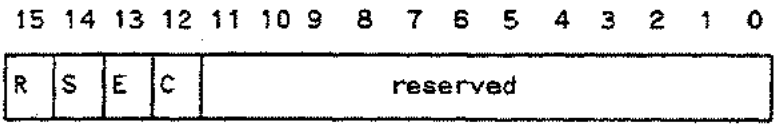
where

length1	is the length, in 16-bit words, of the header and appendage.
message class	is the message class as shown in Table 7-1.
stream word (word 2)	consists of the fields shown below:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RP	RJ	CN	BR	CM	NB	r		DS/3000 stream type							

RP	Reply bit. Reply if set, Request if 0.
RJ	Reject bit. Message rejected if set.
CN	Continue bit. Continuation record to follow, if set.
BR	
CM	Compression bit. Set if the message has been compressed.
NB	Non-PTOP break bit. Set if a break for non-PTOP activity has been detected.
r	is reserved.

DS/3000 stream type	is the stream type as shown in Table 7-1.
length2 (word 7)	is the length, in bytes, of the appendage and user data. In the diagrams later in this section, this field is described in the following format: AL + DL AL is the length of the appendage in bytes. DL is the length of the data in bytes. If the length of either the appendage or data is variable the placeholders AL and DL are used in the diagram.
from process # to process #	is the number of the process to which a particular stream type or operation belongs. The "from process #" is echoed as the "to process #" in a reply. For master requests from the HP 1000 to the HP 3000, the "from process #" is the console LU number specified by hello.
capability mask word (word 6)	is for initialization request and reply only. For initialization request and reply, this word takes the following format:



R Reply bit. Set in initialization reply if capability mask is valid for replying system.

S Set if sequence numbers are supported.

E Set if exclusive mode is supported without exclusive mode protocol.

C Set if $STDIN/$STDLIST continuation records are supported.

Each of the message format diagrams shown in the remainder of this section shows both the fixed header and additional information that is appended to the header. This additional information consists of parameters specific to the particular message followed by user data.

For REQUEST messages, parameters specific to the message consist of calling parameters. For REPLY messages, parameters specific to the message consist of return parameters. The user data appended to these parameters is data associated with a particular call--for example, data associated with an FWRITE call, or data associated with an FREAD call.

NOTE

Although initialization messages consist of only the eight word fixed header, word 3 has special meaning for initialization messages only (class 0, stream type 20), as follows: the left byte of word 3 contains the maximum buffer size, and the right byte contains the current buffer size.

All values are in octal on the following pages of this section, unless otherwise noted.

REMOTE Command: Class 3, Stream 20

REQUEST

word		
0	(21+AL)/2	3
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	AL + 0	
	remote command (ASCII string)	

REPLY

word		
0	10	3
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	0	

NOTE

AL is the length, in bytes, of the appendage. See the introduction pages at the beginning of this section for more detailed description of words 0 through 7 of each message.

PREAD: Class 4

request: stream 22
accept reply: stream 26
reject reply: stream 27

REQUEST

word		
0	41	4
1	reserved	
2	stream word	22
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	62 + 0	
0	0	
1	tcount	
2	slave program ID	
3	slave I/O class	
4	0	
5	tag	
24		

REPLY

word		
0	41	4
1	reserved	
2	stream word	26 or 27
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	62 + DL	
0	error code	
1	tcount´	
2	slave program ID	
3	slave I/O class	
4	0	
5	tag	
24		
25	data (accept only)	

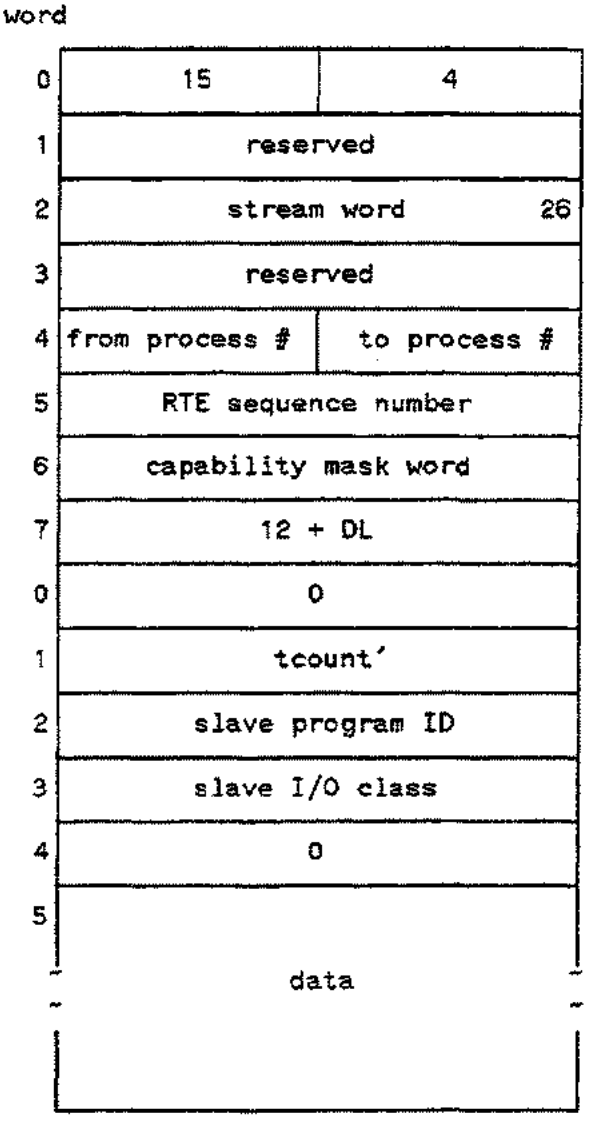
Note:

tcount specifies the requested read length.

tcount´ specifies the actual read length.

PREAD REPLY
CONTINUATION
RECORD

(ACCEPT ONLY)



NOTE

DL is the length, in bytes, of the user data. See the introductory pages at the begining of this section for more detailed description of word 0 through 7 of each message.

PWRITE: Class 4

request: stream 23
accept reply: stream 26
reject reply: stream 27

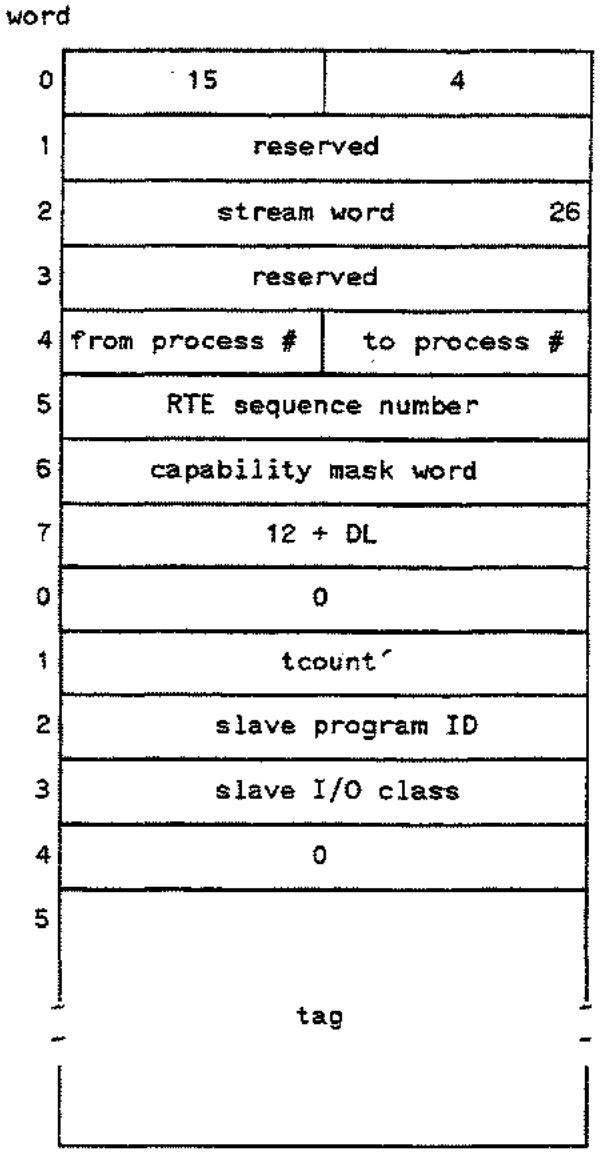
REQUEST

word		
0	41	4
1	reserved	
2	stream word	23
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	62 + DL	
0	0	
1	tcount	
2	slave program ID	
3	slave I/O class	
4	0	
5	tag	
24		
25	data (accept only)	

REPLY

word		
0	41	4
1	reserved	
2	stream word	26 or 27
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	62 + DL	
0	error code	
1	0	
2	slave program ID	
3	slave I/O class	
4	0	
5	tag	
24		

PWRITE REPLY
CONTINUATION RECORD
(ACCEPT ONLY)



PCONTROL: Class 4

request: stream 24
accept reply: stream 26
reject reply: stream 27

REQUEST

word		
0	41	4
1	reserved	
2	stream word	24
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	62 + 0	
0	0	
1	0	
2	slave program ID	
3	slave I/O class	
4	0	
5	tag	
24		

REPLY

word		
0	41	4
1	reserved	
2	stream word	26 or 27
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	62 + 0	
0	error code	
1	0	
2	slave program ID	
3	slave I/O class	
4	0	
5	tag	
24		

$STDLIST (Print): Class 5, Stream 20

REQUEST

word		
0	12	5
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	write length + 2	
0	control *	
1	blockmode indication	
	characters to be printed	

continuation request

0	10	5
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	data length	
0	characters to be printed	

REPLY

word		
0	11	5
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	2 + 0	
0	status word	

continuation reply

0	10	5
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to prcess #
5	RTE sequence number	
6	capability mask word	
7	0	

Control* (word 0 of request parameters): For more information refer to the *MPE Intrinsics Reference Manual* under carriage control.

FCONTROL for $STDIN/$STDLIST: Class 5, Stream 23

REQUEST

word		
0	13	5
1	reserved	
2	stream word	23
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	6 + 0	
0	filenum	
1	control code	
2	param	

REPLY

word		
0	12	5
1	reserved	
2	stream word	23
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	4 + 0	
0	status	
1	param	

$STDIN (READ/READX): Class 5, Stream 21

REQUEST

word		
0	13	5
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	6 + 0	
0	length	
1	unused	
2	blockmode indication	

REPLY

word		
0	12	5
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	read length+4	
0	status	
1	read length	
-	characters that were read	
-		

continuation request

0	10	5
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	0	

continuation reply

0	10	5
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	read length	
0		
-	characters that were read	
-		

REMOTE HELLO: Class 6, Stream 20

REQUEST

word		
0	(21+AL)/2	6
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	AL + 0	
	MPE HELLO command	

REPLY

word		
0	10	6
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	0	

REMOTE BYE: Class 6, Stream 21

REQUEST

word		
0	12	6
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	4 + 0	
0	"B"	"Y"
1	"E"	" "

REPLY

word		
0	10	6
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	0	

FCHECK: Class 7, Stream 20

File System Intrinsic Number 14

REQUEST

word		
0	15	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	12 + 0	
0	"R"	"F"
1	"A"	" "
2	16 (intrinsic number)	
3	filenum	
4	mask	

REPLY

word		
0	17	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	16 + 0	
0	status word	
1	errorcode	
2	tlog	
3	blknum	
4		
5	numrec	
6	mask	

FCLOSE: Class 7, Stream 20

File System Intrinsic Number 2

REQUEST

word		
0	16	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	14 + 0	
0	"R"	"F"
1	"A"	" "
2	2 (intrinsic number)	
3	filenum	
4	disposition	
5	seccode	

REPLY

word		
0	11	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	2 + 0	
0	status word	

FCONTROL: Class 7, Stream 20

File System Intrinsic Number 15

REQUEST

word		
0	16	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	14 + 0	
0	"R"	"F"
1	"A"	" "
	17 (intrinsic number)	
3	filenum	
4	control code	
5	param	

REPLY

word		
0	12	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	4 + 0	
0	status word	
1	param	

FGETINFO: Class 7, Stream 20

File Intrinsic Number 13

REQUEST

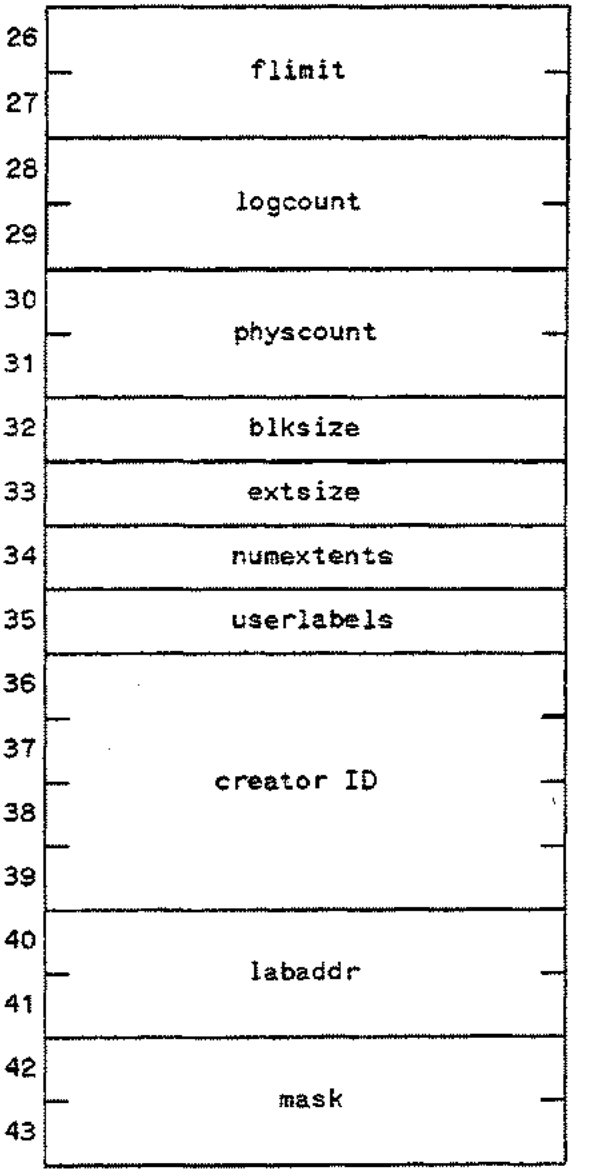
word		
0	16	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	14 + 0	
0	"R"	"F"
1	"A"	" "
2	15 (intrinsic number)	
3	filenum	
4	mask	
5		

REPLY

word		
0	64	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	130 + 0	
0	status word file name (words 1 through 14)	
15	foptions	
16	aoptions	
17	recsize	
18	devtype	
19	ldnum	
20	hdaddr	
21	filecode	
22	recpt	
23		
24	EOF	
25		

(continued on next page)

(FGETINFO REPLY, continued)



FLOCK: Class 7, Stream 20

File System Intrinsic Number 19

REQUEST

word		
0	17	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	16 + 0	
0	"R"	"F"
1	"A"	" "
2	23 (intrinsic number)	
3	filenum	
4	lockcond	
5	Q - 7	
6	Q - 6	

REPLY

word		
0	13	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	6 + 0	
0	status word	
1	Q - 7	
2	Q - 6	

FOPEN: Class 7, Stream 20

File System Intrinsic Number 1

REQUEST

word		
0	(21+AL)/2	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	AL + 0	
0	"R"	"F"
1	"A"	" "
2	1 (intrinsic number)	
3	@formaldesignator	
4	foptions	
5	aoptions	
6	recsize	
7	@device name	
8	@formsg	
9	userlabels	
10	blockfactor	
11	numbuffers	
12	filesize	
13	0	

(continued on next page)

REPLY

word		
0	12	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	4 + 0	
0	status word	
1	mask	

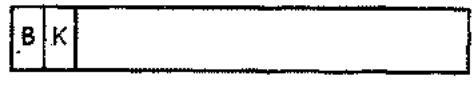
Right byte of REPLY status word contains the file number.

@ = byte pointer (relative to n) to area within the request buffer.

(FOPEN REQUEST, continued)

14	numextents
15	initialloc
16	filecode
17	mask
	formaldesignator (variable length)
	device name (variable length)
	formmsg (variable length)

mask (word 17) has the following meaningful bits:



B passes the privileged/non-privileged status of the code calling FOPEN on the local side (sign bit).

K signals that the secondary entry point of KOPEN was called to do the FOPEN.

FPOINT: Class 7, Stream 20

File System Intrinsic Number 12 (decimal)

REQUEST

word		
0	16	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	14 + 0	
0	"R"	"F"
1	"A"	" "
2	14 (intrinsic number)	
3	filenum	
4	recnum	
5		

REPLY

word		
0	11	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	2 + 0	
0	status word	

FREAD (Not Multirecord): Class 7, Stream 20

File System Intrinsic Number 3

REQUEST

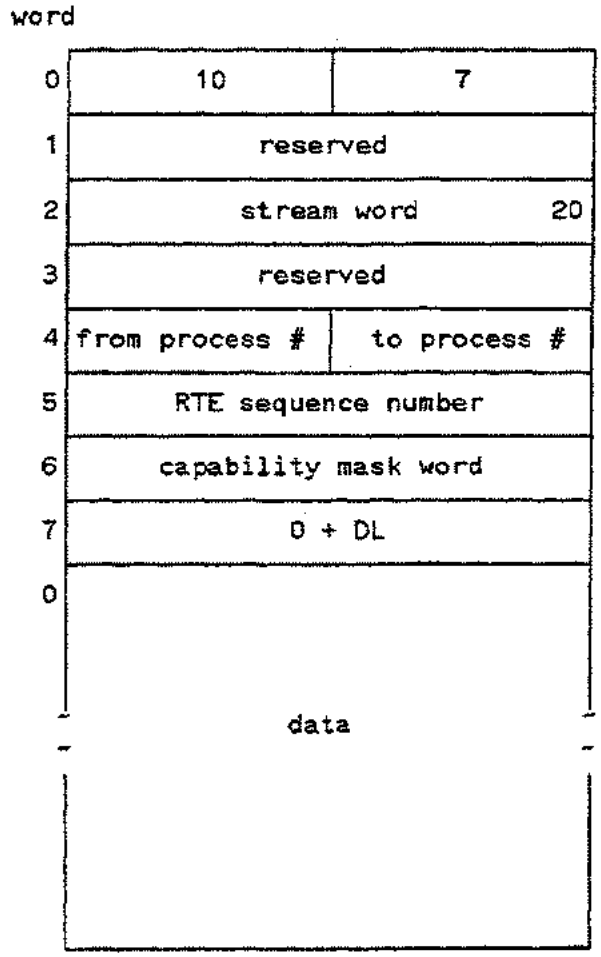
word		
0	15	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	12 + 0	
0	"R"	"F"
1	"A"	" "
2	err #	3
3	filenum	
4	tcount	

err #: Used when local system finds an error in a file. err # passed to remote system.

REPLY

word		
0	12	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	4 + DL	
0	status word	
1	tcount'	
2	data	

FREAD REPLY
CONTINUATION
RECORD



FREADDIR (Not Multirecord): Class 7, Stream 20

File System Intrinsic Number 4

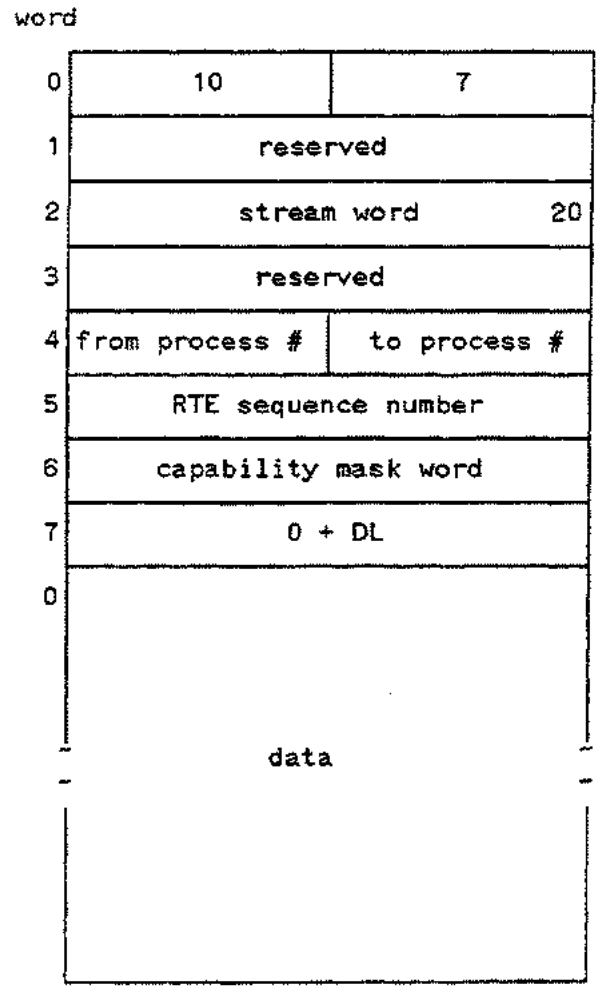
REQUEST

word		
0	17	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	16 + 0	
0	"R"	"F"
1	"A"	" "
2	err #	4
3	filenum	
4	tcount	
5	recnum	
6		

REPLY

word		
0	12	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	4 + DL	
0	status word	
1	tcount	
2	data	

FREADDIR REPLY
CONTINUATION
RECORD



FREADSEEK: Class 7, Stream 20

File System Intrinsic Number 5

REQUEST

word		
0	16	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	14 + 0	
0	"R"	"F"
1	"A"	" "
2	5 (intrinsic number)	
3	filenum	
4	recnum	
5		

REPLY

word		
0	11	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	2 + 0	
0	status word	

FRELATE: Class 7, Stream 20

File System Intrinsic Number 18

REQUEST

word		
0	15	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	10 + 0	
0	"R"	"F"
2	"A"	" "
3	22 (intrinsic number)	
4	infilenum	
5	listfilenum	

REPLY

word		
0	12	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	4 + 0	
0	status word	
1	reply	

FRENAME: Class 7, Stream 20

File System Intrinsic Number 17

REQUEST

word		
0	(AL+21)/2	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	name length+8	
0	"R"	"F"
1	"A"	" "
2	21 (intrinsic number)	
3	filenum	
4	new file name (up to 28 characters) (28 = decimal)	

REPLY

word		
0	11	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	2 + 0	
0	status word	

FREADLABEL: Class 7, Stream 20

File System Intrinsic Number 8

REQUEST

word		
0	17	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	16 + 0	
0	"R"	"F"
1	"A"	" "
2	10 (intrinsic number)	
3	filenum	
4	tcount	
5	labelid	
6	mask	

Label can be up to 128 (decimal) words.

REPLY

word		
0	(21+AL)/2	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	(label length*2)+2	
0	status word	
1	label	

FSETMODE: Class 7, Stream 20

File System Intrinsic Number 16

REQUEST

word		
0	15	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	12 + 0	
0	"R"	"F"
1	"A"	" "
2	20 (intrinsic number)	
3	filenum	
4	modeflags	

REPLY

word		
0	11	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	2 + 0	
0	status word	

FSPACE: Class 7, Stream 20

File System Intrinsic Number 11

REQUEST

word		
0	15	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	14 + 0	
0	"R"	"F"
1	"A"	" "
2	13 (intrinsic number)	
3	filenum	
4	displacement	

REPLY

word		
0	11	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	2 + 0	
0	status word	

FUNLOCK: Class 7, Stream 20

File System Intrinsic Number 20

REQUEST

word		
0	16	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	14 + 0	
0	"R"	"F"
1	"A"	" "
2	24 (intrinsic number)	
3	filenum	
4	Q - 7	
5	Q - 6	

REPLY

word		
0	11	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	2 + 0	
0	status word	

FUPDATE (Not Multirecord): Class 7, Stream 20

File System Intrinsic Number 10

REQUEST

word		
0	15	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	12 + DL	
0	"R"	"F"
1	"A"	" "
2	12 (intrinsic number)	
3	filenum	
4	tcount	
5	data	

REPLY

word		
0	11	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	2 + 0	
0	status word	

FWRITE (Not Multirecord): Class 7, Stream 20

File System Intrinsic Number 6

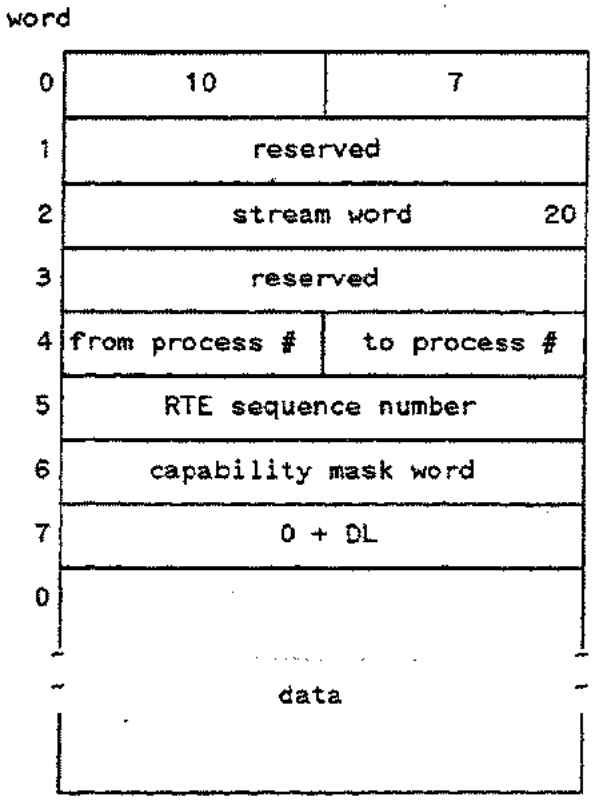
REQUEST

word		
0	16	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	14 + DL	
0	"R"	"F"
1	"A"	" "
2	err #	6
3	filenum	
4	tcount	
5	control	
6	data	

REPLY

word		
0	11	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	2 + 0	
0	status word	

FWRITE REQUEST
CONTINUATION
RECORD



FWRITEDIR (Not Multirecord): Class 7, Stream 20

File System Intrinsic Number 7

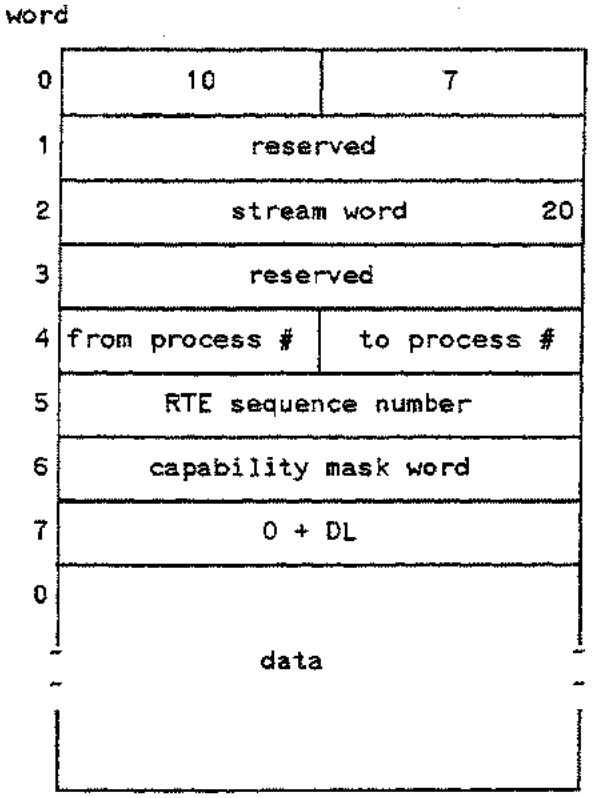
REQUEST

word		
0	17	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	16 + DL	
0	"R"	"F"
1	"A"	" "
2	err #	7
3	filenum	
4	tcount	
5	recnum	
6		
7	data	
-	-	-
-	-	-

REPLY

word		
0	11	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	2 + 0	
0	status word	

FWRITEDIR REQUEST
CONTINUATION
RECORD



FWRITELABEL: Class 7, Stream 20

File System Intrinsic Number 9

REQUEST

word		
0	(21+AL)/2	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	label length +16	
0	"R"	"F"
1	"A"	" "
2	11 (intrinsic number)	
3	filenum	
4	tcount	
5	labelid	
6	mask	
7	label	

REPLY

word		
0	11	7
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	2 + 0	
0	status word	

Label can be up to 128 (decimal) words.

PCLOSE: Class 7, Stream 21

File System Intrinsic Number 22

REQUEST

word		
0	16	7
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	14 + 0	
0	"R"	"F"
1	"A"	" "
2	26 (intrinsic number)	
3	slave program ID	
4	slave I/O class	
5	0	

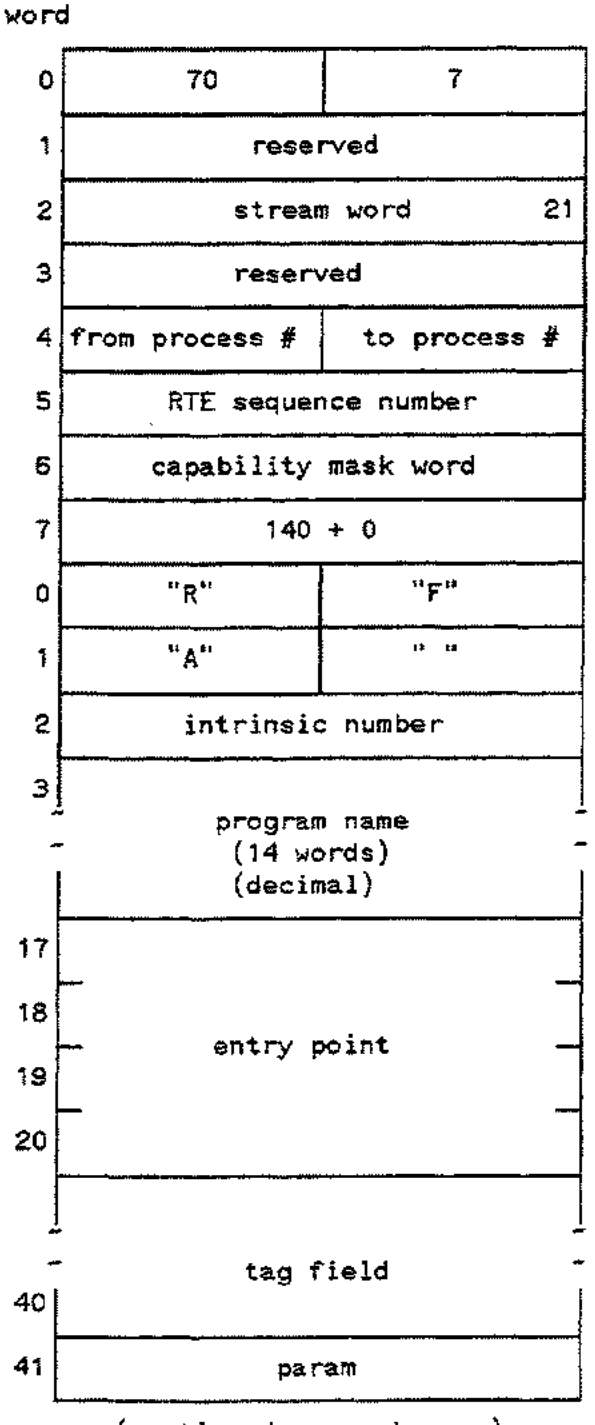
REPLY

word		
0	10	7
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	0 + 0	

POPEN: Class 7

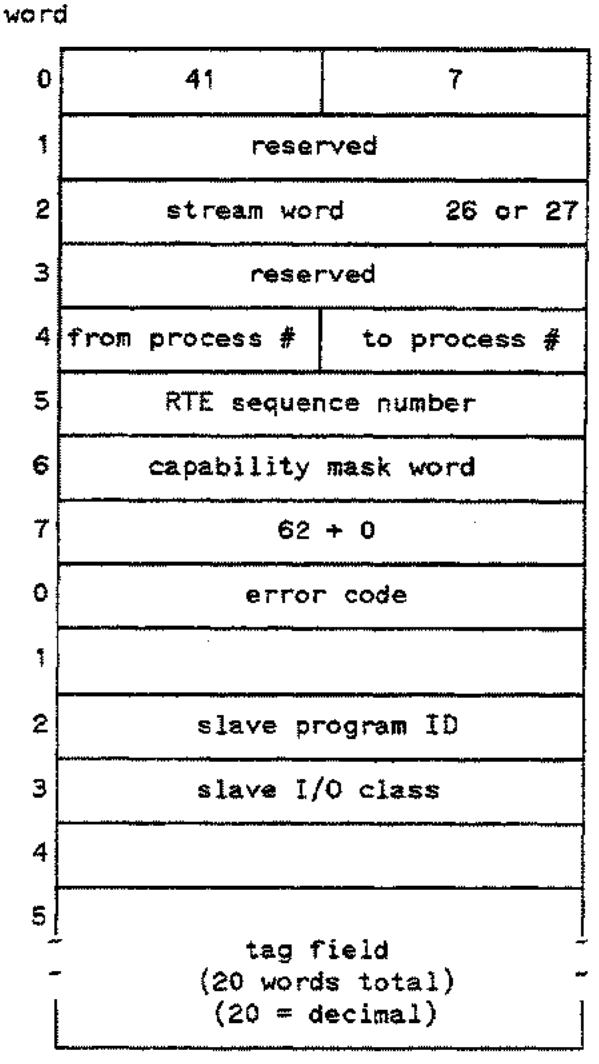
request: stream 24
accept reply: stream 26
reject reply: stream 27
File System Intrinsic number 21

REQUEST



(continued on next page)

REPLY



mask indicates which parameters were provided (bit = 0 means use default)

- 9 - program name
- 8 - entry point
- 6 - param
- 5 - flags
- 4 - stacksize
- 3 - DL size
- 2 - maxdata

(POPEN REQUEST, continued)

42	flags
43	stacksize
44	DL size
45	maxdata
46	bufsize
47	mask

DAPOS: Class 8, Stream 20

Intrinsic Number 161

REQUEST

word		
0	22	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	24 + 0	
0	241 (intrinsic number)	
1	0	
2	0	
3	0	
4	dcb	
5		
6		
7	rec	
8	rb	
9	off	

REPLY

word		
0	13	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	6 + 0	
0	A-register	
1	0	
2	err	

DLCOS: Class 8, Stream 20

Intrinsic Number 157

REQUEST

word		
0	20	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	20 + 0	
0	235 (intrinsic number)	
1	0	
2	0	
3	0	
4	dcb	
5		
6		
7	trun	

REPLY

word		
0	13	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	6 + 0	
0	A-register	
1	0	
2	err	

DCONT: Class 8, Stream 20

Intrinsic Number 159

REQUEST

word		
0	21	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	22 + 0	
0	237 (intrinsic number)	
1	0	
2	0	
3	0	
4	dcb	
5		
6		
7	con1	
8	con2	

REPLY

word		
0	13	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	6 + 0	
0	A-register	
1	0	
2	err	

DCRET: Class 8, Stream 20

Intrinsic Number 150

REQUEST

word		
0	24	8
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	28 + 0	
0	226 (intrinsic number)	
1	0	
2	0	
3	0	
4	name	
5		
6		
7	size (1)	
8	size (2)	
9	type	
10	secu	
11	cr	

REPLY

word		
0	13	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	6 + 0	
0	A-register	
1	RFAM entry number	
2	err	

DLOCF: Class 8, Stream 20

Intrinsic Number 160

REQUEST

word		
0	17	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	16 + 0	
0	240 (intrinsic number)	
1	0	
2	0	
3	0	
4	dcb	
5		
6		

REPLY

word		
0	22	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	24 + 0	
0	A-register	
1	0	
2	err	
3	rec	
4	rb	
5	off	
6	sec	
7	lu	
8	ty	
9	recsz	

DNAME: Class 8, Stream 20

Intrinsic Number 158

REQUEST

word		
0	24	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	28 + 0	
0	236 (intrinsic number)	
1	0	
2	0	
3	0	
4	name	
5		
6		
7	nname	
8		
9		
10	secu	
11	cr	

REPLY

word		
0	13	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	6 + 0	
0	A-register	
1	0	
2	err	

DOPEN: Class 8, Stream 20

Intrinsic Number 152

REQUEST

word		
0	22	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	24 + 0	
0	230 (intrinsic number)	
1	0	
2	0	
3	0	
4	name	
5		
6		
7	optn	
8	secu	
9	cr	

REPLY

word		
0	13	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	6 + 0	
0	A-register	
1	RFAM entry number	
2	err	

DPOSN: Class 8, Stream 20

Intrinsic Number 155

REQUEST

word		
0	21	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	22 + 0	
0	233 (intrinsic number)	
1	0	
2	0	
3	0	
4	dcb	
5		
6		
7	nur	
8	rec	

REPLY

word		
0	13	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	6 + 0	
0	A-register	
1	0	
2	err	

DPURG: Class 8, Stream 20

Intrinsic Number 151

REQUEST

word		
0	21	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	22 + 0	
0	227 (intrinsic number)	
1	0	
2	0	
3	0	
4	name	
5		
6		
7	secu	
8	cr	

REPLY

word		
0	13	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	6 + 0	
0	A-register	
1	0	
2	err	

DREAD: Class 8, Stream 20

Intrinsic number 154

REQUEST

word		
0	22	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	24 + 0	
0	232 (intrinsic number)	
1	0	
2	user data length (in words)	
3	0	
4	dcb	
5		
6		
7	len	
8	0	
9	num	

REPLY

word		
0	10	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	0 + DL	
0	data	

DREAD REPLY
LAST
RECORD

word		
0	14	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	10 + 0	
0	A-registry	
1	0	
2	err	
3	rlen	

DSTAT: Class 8, Stream 20

Intrinsic Number 162

REQUEST

word		
0	14	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	10 + 0	
0	242 (intrinsic number)	
1	0	
2	124 (decimal)	
3	0	

REPLY

word		
0	12	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	0 + 370	
0	status array (124, decimal, words) (sent as data)	

LAST RECORD (REPLY)

0	12	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	4 + 0	
0	A-register	
1	0	

DWRIT: Class 8, Stream 20

Intrinsic Number 153

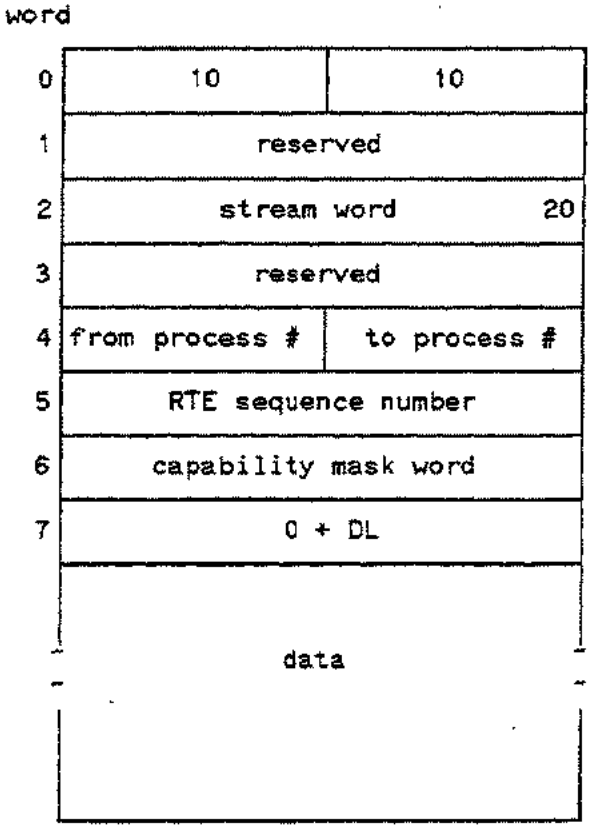
REQUEST

word		
0	21	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	22 + DL	
0	231 (intrinsic number)	
1	0	
2	user data length (in words)	
3	0	
4	dcb	
5		
6		
7	len	
8	num	
9	data	
-		

REPLY

word		
0	13	10
1	reserved	
2	stream word	20
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	6 + 0	
0	A-register	
1	0	
2	err	

DWRIT REQUEST
CONTINUATION
RECORD



DEXEC WRITE: Class 8, Stream 21

Intrinsic Number 163, code = 2

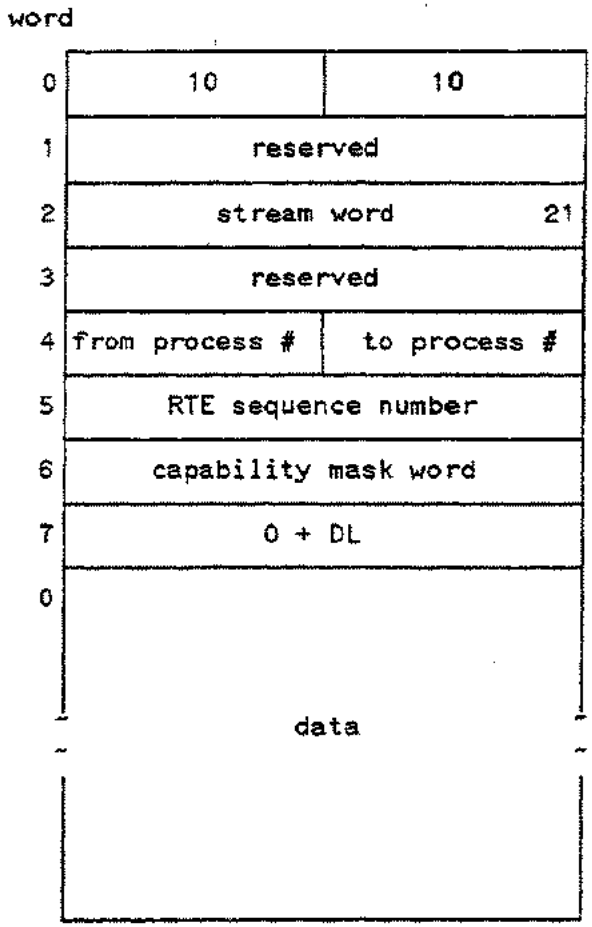
FIRST REQUEST

word		
0	21	10
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	22 + DL	
0	243 (intrinsic number)	
1	0	
2	user data length (words)	
3	0	
4	2 (code)	
5	cnwd	
6	buf1	
7	prm1	
8	prm2	
9	data	

REPLY

word		
0	12	10
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	4 + 0	
0	A-register (EQT 5)	
1	B-register (XMSN)	

DEXEC WRITE REQUEST
ADDITIONAL
DATA



DEXEC EXECUTION TIME: Class 8, Stream 20

Intrinsic Number 163, code = 12

REQUEST

word		
0	26	10
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	34 + 0	
0	243 (intrinsic number)	
1	0	
2	0	
3	0	
4	14 (code)	
5	program name	
6		
7		
8	res1	
9	mtple	
10	ofst	
11	mins	
12	secs	
13	msecs	

Not used if word 10 is negative.

REPLY

word		
0	12	10
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	4 + 0	
0	A-register	
1	B-register	

DEXEC I/O CONTROL: Class 8, Stream 20

Intrinsic Number 163, code = 3

REQUEST

word		
0	17	10
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	16 + 0	
0	243 (intrinsic number)	
1	0	
2	0	
3	0	
4	3 (code)	
5	cnwd	
6	prm1	

REPLY

word		
0	12	10
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	4 + 0	
0	A-register	
1	B-register	

DEXEC I/O STATUS: Class 8, Stream 20

Intrinsic Number 163, code = 13

REQUEST

word		
0	16	10
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	14 + 0	
0	243 (intrinsic number)	
1	0	
2	0	
3	0	
4	15 (code)	
5	cnwd	

REPLY

word		
0	15	10
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	12 + 0	
0	A-register	
1	B-register	
2	stat 1 (EQT 5)	
3	stat 2 (EQT 4)	
4	stat 3 (subchannel)	

DEXEC READ: Class 8, Stream 21

Intrinsic Number 163, code = 1

REQUEST

word		
0	21	10
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	22 + 0	
0	243 (intrinsic number)	
1	0	
2	number of data words	
3	0	
4	1 (code)	
5	cnwd	
6	bufl	
7	prm1	
8	prm2	

FIRST RECORD (REPLY)

word		
0	10	10
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	0 + data length	
0	data	

LAST RECORD (REPLY)

0	12	10
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	4 + 0	
0	A-register (EQT 5)	
1	B-register (XMSN)	

DEXEC SCHEDULE: Class 8, Stream 21

Intrinsic Number 163, code = 10

REQUEST

word		
0	25	10
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	32 + 0	
0	243 (intrinsic number)	
1	0	
2	0	
3	0	
4	12 (code)	
5	program name	
6		
7		
8	prm 1	
9	prm 2	
10	prm 3	
11	prm 4	
12	prm 5	

REPLY

word		
0	12	10
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	4 + 0	
0	A-register	
1	B-register	

DEXEC TIME: Class 8, Stream 21

Intrinsic Number 163, code = 11

REQUEST

word		
0	15	10
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	12 + 0	
0	243 (intrinsic number)	
1	0	
2	0	
3	0	
4	13 (code)	

REPLY

word		
0	17	10
1	reserved	
2	stream word	21
3	reserved	
4	from process #	to process #
5	RTE sequence number	
6	capability mask word	
7	16 + 0	
0	A-register	
1	B-register	
2	time [1]	
3	time [2]	
4	time [3]	
5	time [4]	
6	time [5]	

WELL-KNOWN ADDRESSES AND PORTS

APPENDIX A

This appendix contains lists of well-known addresses and ports. It contains the following:

- Well-Known TCP SAPs.
- Well-Known UDP Ports.
- Some Known Ethernet Type Fields.
- Some Known MAC Address Vendor Codes.
- Some Known Ethernet Multicast and Broadcast Addresses.

Hewlett-Packard makes no claims for the current validity of the information in this appendix. This information is provided only for the convenience of the user.

WELL-KNOWN TCP SAPS

Table A-1. Well-Known TCP SAPs

Service	Octal Value	Hex Value	Decimal Value
FTP	25	15	21
TELNET	27	17	23
SMTP	31	19	25
Nameserver	65	35	53
RPC Port Map	157	6F	111
Name Server	211	89	137
Datagram	212	8A	138
Session	213	8B	139
Remote Exec	1000	200	512
rlogin	1001	201	513
remshell	1002	202	514
remote print	1003	203	515
rlb	2354	4EC	1260
NFT	3000	600	1536
VT	3001	601	1537
Reverse VT	3002	602	1538
PTOP	3004	604	1540
RPM	3006	606	1542
Ninstall	4071	839	2105
RFA (MPE)	5000	A00	2560
RDBA (MPE)	5001	A01	2561
RFA (UX)	11100	1240	4672

WELL-KNOWN UDP PORTS

Table A-2. Well-Known UDP Ports

Service	Octal Value	Hex Value
Nameserver	65	35
RPC Port Map	157	6F
Name Server	211	89
Datagram	212	8A
Session	213	8B
rwho, ruptime	1001	201
syslog	1002	202
route (RIP)	1010	208
NFS daemon	4001	801

ETHERNET TYPE FIELD

The 13th and 14th octets of an Ethernet packet consist of the "Type" field. (If the value of this field is less than 1500, or 5DC hex, this is an IEEE 802 packet and this field is interpreted as the "length" field.)

These Ethernet type field values are managed by XEROX. Some assignments are public, others private. Current information includes: Xerox Public Ethernet Packet Type documentation; IEEE802.3 Std; NIC RFC960; and contributions from network managers and vendors.

Table A-3. Ethernet Type Fields

Hex Value	Meaning
0000-05DC	IEEE802.3 Length Field (0 - 1500.)
0101-01FF	Experimental (for development, conflicts with IEEE802.3 Length)
0200	Xerox PUP (conflicts with IEEE802.3 Length Field range) (see 0A00)
0201	Xerox PUP Address Translation (conflicts ...) (see 0A01)
0600	Xerox XNS IDP
0800	DOD Internet Protocol (IP) (See notes 1 & 2 below.)
0801	X.75 Internet
0802	NBS Internet
0803	ECMA Internet
0804	CHAOSnet
0805	X.25 Level 3
0806	Address Resolution Protocol (ARP) (for IP & CHAOS) (See Note 1.)
0807	XNS Compatibility
081C	Symbolics Private
0888-088A	Xyplex
0900	Ungermann-Bass network debugger
0A00	Xerox IEEE802.3 PUP
0A01	Xerox IEEE802.3 PUP Address Translation
0BAD	Banyan Systems
1000	Berkeley Trailer negotiation
1001-100F	Berkeley Trailer encapsulation for IP
1600	BBN Simnet Private
4242	PCS Basic Block Protocol
5208	BBN Simnet Private (See note 3.)
6000	DEC unassigned
6001	DEC Maintenance Operation Protocol (MOP) Dump/Load Assistance
6002	DEC Maintenance Operation Protocol (MOP) Remote Console
6003	DECNET Phase IV
6004	DEC Local Area Transport (LAT)
6005	DEC diagnostic protocol
6006	DEC customer protocol
6007	DEC DECNet SCA
6008	DEC AMBER
6009	DEC MUMPS
6010-6014	3Com Corp.

Table A-3. Ethernet Type Fields (continued)

Hex Value	Meaning
7000	Ungermann-Bass download
7001	Ungermann-Bass NIU
7002	Ungermann-Bass diagnostic/loopback
7020-7029	LRT (England)
7030	Proteon
8003	Cronus VLN
8004	Cronus Direct
8005	HP Probe protocol
8006	Nestar
8008	AT&T
8010	Excelan
8013	Silicon Graphics diagnostic (obsolete)
8014	Silicon Graphics network games (obsolete)
8015	Silicon Graphics reserved (obsolete)
8016	Silicon Graphics XNS NameServer, bounce server (obsolete)
8019	HP-Apollo DOMAIN
802E	Tymshare
802F	Tigan, Inc.
8035	Reverse Address Resolution Protocol (RARP)
8036	Aeonic Systems
8038	DEC LanBridge Management
8039	DEC unassigned
803A	DEC unassigned
803B	DEC unassigned
803C	DEC unassigned
803D	DEC Ethernet Encryption Protocol
803E	DEC unassigned
803F	DEC LAN Traffic Monitor Protocol
8040	DEC unassigned
8041	DEC unassigned
8042	DEC unassigned
8044	Planning Research Corp.
8046	AT&T
8047	AT&T
8049	ExperData (France)
805B	Versatile Message Translation Protocol RFC-1045 (Stanford)
805C	Stanford V Kernel, production
805D	Evans & Sutherland
8060	Little Machines
8062	Counterpoint Computers
8065	University of Massachusetts, Amherst
8066	University of Massachusetts, Amherst
8067	Veeco Integrated Automation
8068	General Dynamics
8069	AT&T
806A	Autophon (Switzerland)
806C	ComDesign
806D	Compugraphic Corp.

Table A-3. Ethernet Type Fields (continued)

Hex Value	Meaning
806E-8077	Landmark Graphics Corp.
807A	Matra (France)
807B	Dansk Data Elektronic A/S (Denmark)
807C	Merit Internodal
807D	VitaLink Communications
807E	VitaLink Communications
807F	VitaLink Communications
8080	VitaLink Communications bridge
8081	Counterpoint Computers
8082	Counterpoint Computers
8083	Counterpoint Computers
8088	Xyplex
8089	Xyplex
808A	Xyplex
809B	EtherTalk (AppleTalk over Ethernet)
809C	Datability
809D	Datability
809E	Datability
809F	Spider Systems, Ltd. (England)
80A3	Nixdorf Computer (West Germany)
80A4-80B3	Siemens Gammasonics Inc.
80C0	Digital Communications Assoc.
80C1	Digital Communications Assoc.
80C2	Digital Communications Assoc.
80C3	Digital Communications Assoc.
80C6	Pacer Software
80C7	Applitek Corp.
80C8-80CC	Integraph Corp.
80CD	Harris Corp.
80CE	Harris Corp.
80CF-80D2	Taylor Inst.
80D3	Rosemount Corp.
80D4	Rosemount Corp.
80D5	IBM SNA Services Over Ethernet
80DD	Varian Assoc.
80DE	TRFS (Integrated Solutions Transparent Remote File System)
80DF	Integrated Solutions
80E0	Allen-Bradley
80E3	Allen-Bradley
80E4-80F0	Datability
80F2	Retix
80F3	Kinetics, AppleTalk ARP (AARP)
80F4	Kinetics
80F5	Kinetics
80F7	HP-Apollo Computer
80FF-8103	Wellfleet Communications
8107	Symbolics Private
8108	Symbolics Private

Table A-3. Ethernet Type Fields (continued)

Hex Value	Meaning
8109	Symbolics Private
8130	Waterloo Microsystems
8131	VG Laboratory Systems
8137	Novell (old)
8138	Novell
8139-813D	KTI
9000	Loopback (Configuration Test Protocol)/ DEC MOP LAN Loopback
9001	Bridge Communications XNS Systems Management
9002	Bridge Communications TCP/IP Systems Management
9003	Bridge Communications
FF00	BBN VITAL-LanBridge cache wakeups %

Note 1: These protocols use Ethernet broadcast, where multicast would be preferable.
Note 2: BBN Butterfly Gateways also use 0800 for non-IP, with IP version field = 3.
Note 3: BBN Private Protocols, not registered

MAC ADDRESS VENDOR CODES

Table A-4. MAC Address Vendor Codes

Hex Value	Address
00000C	Cisco
000020	DIAB (Data Intdustrier AB)
000022	Visual Technology
00002A	TRW
00005A	S & Koch
000065	Network General
000093	Proteon
00009F	Ameristar Technology
0000A9	Network Systems
0000AA	Xerox, Xerox machines
0000B3	CIMLinc
0000C0	Western Digital
0000DD	Gould
000102	BBN, BBN internal usage (not registered)
001700	Kabel
00DD00	Ungermann-Bass
00DD01	Ungermann-Bass
020701	Interlan, UNIBUS or QBUS machines, HP-Apollo
020406	BBN, BBN internal usage (not registered)
02608C	3Com: IBM PC; Imagen; Valid
02CF1F	CMC, Masscomp, Silicon Graphics
080002	Bridge
080003	ACC (Advanced Computer Communications)
080005	Symbolics, LISP machines
080008	BBN
080009	Hewlett-Packard
08000A	Nestar Systems
08000B	Unisys
080010	AT+T
080014	Excelan, BBN Butterfly, Masscomp, Silicon Graphics
080017	NSC
08001A	Data General
08001B	Data General
08001E	HP-Apollo
080020	Sun, Sun machines
080022	NBI
080025	CDC
080028	TI, Explorer
08002B	DEC, UNIBUS or QBUS machines, VAXen, LANBridges (DEUNA, DEQNA, DELUA)
080036	Intergraph, CAE stations
080039	Spider Systems
080045	Xylogics
080047	Sequent

Table A-4. MAC Address Vendor Codes (continued)

Hex Value	Address
080049	Univation
08004C	Encore
08004E	BICC
08005A	IBM
080067	Comdesign
080068	Ridge
080069	Silicon Graphics
08006E	Excelan
080075	DDE (Danish Data Elektronik A/S)
08007C	Vitalink, TransLAN III
080080	XIOS
080089	Kinetics, AppleTalk-Ethernet interface
08008B	Pyramid
08008D	XyVision, XyVision machines
AA0003	DEC, Global physical address for some DEC machines
AA0004	DEC, Local logical address for systems running DECNET

ETHERNET MULTICAST AND BROADCAST ADDRESSES

Table A-5. Ethernet Multicast Addresses

Ethernet Multi-cast Address	Type Field	Usage
09-00-02-04-00-01	8080	Vitalink printer
09-00-02-04-00-02	8080	Vitalink management
09-00-09-00-00-01	8005	HP Probe
09-00-09-00-00-01	802.2LLC	HP Probe
09-00-09-00-00-04	8005	HP DTC
09-00-1E-00-00-00	8019	HP-Apollo DOMAIN
09-00-2B-00-00-03	8038	DEC Lanbridge Traffic Monitor (LTM)
09-00-2B-00-00-0F	6004	DEC Local Area Transport (LAT)
09-00-2B-01-00-00	8038	DEC LanBridge Copy packets
09-00-2B-01-00-01	8038	DEC LanBridge Hello packets 1 packet per second, sent by the designated LanBridge
09-00-4E-00-00-02	8137	Novell IPX
09-00-7C-02-00-05	8080	Vitalink diagnostics
09-00-7C-05-00-01	8080	Vitalink gateway?
0D-1E-15-BA-DD-06	?	HP
AB-00-00-01-00-00	6001	DEC Maintenance Operation Protocol (MOP) Dump/Load Assistance
AB-00-00-02-00-00	6002	DEC Maintenance Operation Protocol (MOP) Remote Console, 1 System ID packet every 8-10 minutes, by every DEC LanBridge DEC DEUNA interface DEC DELUA interface DEC DEQNA interface (in a certain mode)
AB-00-00-03-00-00	6003	DECNET Phase IV end node Hello packets 1 packet every 15 seconds, sent by each DECNET host.
AB-00-00-04-00-00	6003	DECNET Phase IV Router Hello packets 1 packet every 15 seconds, sent by DECNET
AB-00-00-05-00-00 through AB-00-03-FF-FF-FF	????	Reserved DEC
AB-00-03-00-00-00	6004	DEC Local Area Transport (LAT) - old
AB-00-04-00-00-00 through AB-00-04-00-FF-FF	????	Reserved DEC customer private use
AB-00-04-01-xx-yy	6007	DEC Local Area VAX Cluster groups
CF-00-00-00-00-00	9000	Ethernet Configuration Test protocol (Loopback)

Table A-6. Ethernet Broadcast Addresses

Ethernet Broadcast Address	Type Field	Usage
FF-FF-FF-FF-FF-FF	0600	XNS packets, Hello or gateway search? 6 packets every 15 seconds, per XNS station
FF-FF-FF-FF-FF-FF	0800	IP (e.g. RWHOD via UDP) as needed
FF-FF-FF-FF-FF-FF	0804	CHAOS
FF-FF-FF-FF-FF-FF	0806	ARP (for IP and CHAOS) as needed
FF-FF-FF-FF-FF-FF	0BAD	Banyan
FF-FF-FF-FF-FF-FF	1600	VALID packets, Hello or gateway search 1 packets/30 seconds, per VALID station
FF-FF-FF-FF-FF-FF	8035	Reverse ARP
FF-FF-FF-FF-FF-FF	807C	Merit Internodal (INP)
FF-FF-FF-FF-FF-FF	809B	EtherTalk

INDEX

A

C

D

E

F

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

SPECIAL CHARACTERS

HEWLETT® PACKARD

Reorder No. or Manual Part No.

91790-61024
E1292

Printed on Recycled Paper

Printed in USA

91790-61024