HEWLETT PACKARD

NS-ARPA/1000 Maintenance and Principles of Operation

Manual

Software Technology Division
11000 Wolfe Road
Cupertino, CA 95014-9804

Manual Part No. 91790-90031
E1193

Printed in U.S.A. November 1993
Sixth Edition

NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THE MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARs 252.227.7013.

Printing History

The Printing History below identifies the edition of this manual and any updates that are included. Periodically, update packages are distributed which contain replacement pages to be merged into the manual, including an updated copy of this printing history page. Also, the update may contain write-in instructions.

Each reprinting of this manual will incorporate all past updates; however, no new information will be added. Thus, the reprinted copy will be identical in content to prior printings of the same edition with its user-inserted update information. New editions of this manual will contain new information, as well as all updates.

To determine what manual edition and update is compatible with your current software revision code, refer to the Manual Numbering File or the Computer User's Documentation Index. (The Manual Numbering File is included with your software. It consists of an "M" followed by a five digit product number.)

First Edition	Feb 1986	Rev. 2608
Update 1	May 1986	Rev. 2626
Update 2	Oct 1986	Rev. 4010/4.1
Update 3	Aug 1987	Rev. 5000/5.0
Second Edition	Oct 1989	Rev. 5016/5.16
Third Edition	May 1990	Rev. 5020/5.2
Fourth Edition	Aug 1991	Rev. 5240/5.24
Fifth Edition	Dec 1992	Rev. 6000/6.0
Sixth Edition	Nov 1993	Rev. 6100/6.1

Preface

Hewlett-Packard Network Services for the HP 1000 (NS-ARPA/1000) provides the networking software that allows HP computer systems to communicate with each other.

Audience

NS-ARPA/1000 Maintenance and Principles of Operation is intended for the Network Manager, whose responsibilities include network configuration, maintenance, and troubleshooting.

Contents

This manual describes the NS-ARPA/1000 maintenance utilities, troubleshooting techniques, and the internal operation of NS-ARPA/1000.

The Network Manager should use this manual in conjunction with *NS-ARPA/1000 Generation and Initialization*, part number 91790-90030. *NS-ARPA/1000 Generation and Initialization* describes NS-ARPA/1000 and how to configure, generate, and initialize the HP 1000 nodes in an NS-ARPA network.

Assumptions

This manual assumes that the Network Manager is familiar with NS-ARPA/1000 and its capabilities, and has read *NS-ARPA/1000 Generation and Initialization* and *NS-ARPA/1000 User/Programmer Reference Manual*, part number 91790-90020. The Network Manager should also be familiar with the RTE-A operating system.

In addition, the Network Manager should be familiar the operating systems of any other machines in the network and any link subsystems used, such as LAN/1000 or X.25/1000.

Organization

Section 1	**Troubleshooting Guidelines**—gives troubleshooting guidelines and describes how to use maintenance utilities for troubleshooting.
Section 2	**NS-ARPA Information Utility**—describes NSINF, the NS-ARPA information utility. NSINF prints the status and parameter values of NS-ARPA/1000 components.
Section 3	**Level 3 Loop Back Diagnostic, PING**—describes PING, the level 3 (IP level) loop back diagnostic.
Section 4	**NS-ARPA Event Logging**—describes the utilities used to record event logging messages. These utilities are EVMON and BREVL.
Section 5	**NS-ARPA Message Tracing**—describes the utilities used to trace NS-ARPA messages and DS/1000-IV Compatible Services (RTE-RTE) messages. These utilities are NSTRC, BRTRC, and FMTRC.
Section 6	**Nodal Registry List Utility (NRLIST)**—describes the Nodal Registry List utility, NRLIST. NRLIST lists the contents of the Nodal Registry.
Section 7	**DS/1000-IV (RTE-MPE) Message Tracing Utilities**—describes LOG3K and TRC3K, the utilities used to trace and format DS/1000-IV Compatible Services (RTE-MPE) messages.
Section 8	**Modification Utility (DSMOD)**—describes the DS/1000-IV Compatible Services parameter modification utility, DSMOD.
Section 9	**Principles of Operation**—describes the internal operation of NS-ARPA/1000.

Caution

If you have purchased the 12079A LAN/1000 Direct Driver Access (DDA) product, please note that HP does not support NS-ARPA/1000 under the following condition:

- NS-ARPA/1000 used with an altered LAN/1000 interface driver.

With RTE-A and NS-ARPA/1000 software revision codes of 5.0 or later, the 12079A DDA product is supported when NS-ARPA/1000 is also installed. There are restrictions applied to user-written LAN programs. Refer to the *LAN/1000 Link Direct Driver Access Manual* for more information.

If you use an Original Equipment Manufacturer's product that incorporates the 12079A LAN/1000 DDA product, HP may require that the OEM product be certified in order to provide support for NS-ARPA/1000. The purpose of such certification is to ensure that the OEM product does not interfere with the NS-ARPA/1000 product. The certification is billable at time and material rates. For information on HP's certification program, contact your HP representative.

HP does not claim compatibility with non-HP implementations of the Internet Protocol (IP) or the Transmission Control Protocol (TCP).

Guide to NS-ARPA/1000 Manuals

The following are brief descriptions of the manuals included with the NS-ARPA/1000 product.

91790-90020 NS-ARPA/1000 User/Programmer Reference Manual

Describes the user-level services provided by NS-ARPA/1000. The NS services are network file transfer (NFT), network interprocess communication (NetIPC), and remote program management (RPM). The ARPA services are TELNET and FTP. Because these are interactive and programmatic services, this manual is intended for interactive users as well as programmers. It should also be read by Network Managers before designing an NS-ARPA/1000 network so that they will have a clear understanding of the full implications of various NS-ARPA/1000 functions and features.

91790-90030 NS-ARPA/1000 Generation and Initialization Manual

Describes the tasks required to install, generate and initialize NS-ARPA/1000. This manual is intended for the Network Manager. Before reading this manual, the Network Manager should read the *NS-ARPA/1000 User/Programmer Reference Manual* to gain an understanding of the NS-ARPA/1000 user-level services. The Network Manager should also be familiar with the RTE-A operating system and system generation procedure.

91790-90031 NS-ARPA/1000 Maintenance and Principles of Operation Manual

Describes the NS-ARPA/1000 network maintenance utilities, troubleshooting techniques, and the internal operation of NS-ARPA/1000. The Network Manager should use this manual in conjunction with the *NS-ARPA/1000 Generation and Initialization Manual*. This manual may also be used by advanced users to troubleshoot their applications.

91790-90040 NS-ARPA/1000 Quick Reference Guide

Lists and briefly describes the interactive and programmatic services described in the *NS-ARPA/1000 User/Programmer Reference Manual* and the *NS-ARPA/1000 DS/1000-IV Compatible Services Reference Manual*. The purpose of this guide is to provide a quick reference for users who are already familiar with the concepts and syntax presented in those two manuals. The *NS-ARPA/1000 Quick Reference Guide* also contains abbreviated syntax for certain programs and utilities described in the *NS-ARPA/1000 Generation and Initialization Manual* and the *NS-ARPA/1000 Maintenance and Principles of Operation Manual*. For your convenience, the *NS-ARPA/1000 Quick Reference Guide* also contains a master index of NS-ARPA/1000 manuals. This is a combined index from the NS-ARPA/1000 manuals to help you find information that may be in more than one manual.

91790-90045 NS-ARPA/1000 Error Message and Recovery Manual

Lists and explains, in tabular form, all of the error codes and messages that can be generated by NS-ARPA/1000. This manual should be consulted by programmers and users who will be writing or maintaining programs for NS-ARPA/1000 systems. Because it contains error messages generated by the NS-ARPA/1000 initialization program NSINIT and other network management programs, it should be consulted by Network Managers.

91790-90050 NS-ARPA/1000 DS/1000-IV Compatible Services Reference Manual

Describes the user-level services provided by the DS/1000-IV backward compatible services. These services are Remote File Access (RFA), DEXEC, REMAT, RMOTE, program-to-program communication (PTOP), utility subroutines, remote I/O mapping, remote system download to memory-based DS/1000-IV nodes only, and remote virtual control panel.

91790-90060 NS-ARPA/1000 BSD IPC Manual

Describes the 4.3 Berkeley Software Distribution Interprocess Communication (BSD IPC) facility on the HP 1000. BSD IPC is a set of programming development tools originally developed by the University of California at Berkeley (UCB). BSD IPC on the HP 1000 offers a programmatic interface for multi-vendor connectivity to other systems with BSD IPC 4.3.

5958-8523 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.

5958-8563 NS Cross-System NFT Reference Manual

Provides cross-system NFT information. It is a generic manual that is a secondary reference source for programmers and operators who will be using NFT on NS-ARPA/1000, NS3000/V, NS3000/XL, NS/9000, NS for the DEC VAX* computer, and PC (PC NFT on HP OfficeShare Network). Information provided in this manual includes file name and login syntax at all of the systems on which NS NFT is implemented, a brief description of the file systems used by each of these computers, and end-to-end mapping information for each supported source/target configuration.

*DEC and VAX are U.S. registered trademarks of Digital Equipment Corporation.

Conventions Used in this Manual

NOTATION	DESCRIPTION
`nonitalics`	Words in syntax statements that are not in italics must be entered exactly as shown. Punctuation characters other than brackets, braces, and ellipses must also be entered exactly as shown. For example: `EXIT;`
`italics`	Words in syntax statements that are in italics denote a parameter that must be replaced by a user-supplied variable. For example: `CLOSE` *`filename`*
`[ ]`	An element inside brackets in a syntax statement is optional. Several elements stacked inside brackets means the user may select any one or none of these elements. For example: `[A B]` User *may* select A or B or neither.
`{ }`	When several elements are stacked within braces in a syntax statement, the user must select one of those elements. For example: `{A B C}` User *must* select A or B or C.
`. . .`	A horizontal ellipsis in a syntax statement indicates that a previous element may be repeated. For example: `[,`*`itemname`*`]...;` In addition, vertical and horizontal ellipses may be used in examples to indicate that portions of the example have been omitted.
,	A shaded delimiter preceding a parameter in a syntax statement indicates that the delimiter *must* be supplied whenever (a) that parameter is included or (b) that parameter is omitted and any *other* parameter that follows is included. For example: *`itema`*`[,`*`itemb`*`][,`*`itemc`*`]` means that the following are allowed: *`itema`* *`itema,itemb`* *`itema,itemb,itemc`* *`itema,,itemc`*

Δ

When necessary for clarity, the symbol Δ may be used in a syntax statement to indicate a required blank or an exact number of blanks. For example:

```
SET[(modifier)]Δ(variable);
```

underlining

When necessary for clarity in an example, user input may be underlined. For example:

```
NEW NAME? ALPHA
```

Brackets, braces or ellipses appearing in syntax or format statements that must be entered as shown will be underlined. For example:

```
LET var[[subscript]] = value
```

Output and input/output parameters are underlined. A notation in the description of each parameter distinguishes input/output from output parameters. For example:

```
CREATE (parm1,parm2,flags,error)
```

The symbol  may be used to indicate a key on the terminal's keyboard. For example, RETURN indicates the carriage return key.

CONTROL *char*

Control characters are indicated by CONTROL followed by the character. For example, CONTROL Y means the user presses the control key and the character Y simultaneously.

Table of Contents

Chapter 1
Troubleshooting Guidelines

Overview 1-1
Summary of Checklists 1-3
    Checklist for Configuration or Initialization Problems 1-3
    Checklist for Other NS-ARPA Problems 1-3
Configuration or Initialization Problems 1-4
    Initialization Error Messages 1-4
    Hardware and Generation Verification 1-5
    Proper Software in Place 1-5
    Local Address Consistency 1-6
    Configured Table Sizes 1-7
    RTE-A Resources 1-7
    Mixing Software Revisions 1-8
    Cross-Network Address Consistency 1-8
Other NS-ARPA Problems 1-9
    General Guidelines 1-9
    NetIPC or BSD IPC Level Problems 1-10
    NetIPC/TCP or BSD IPC Problems 1-11
    DS/1000-IV Compatible Services Problems 1-11
    NFT Problems 1-12
    FTP Problems 1-12
    TELNET Problems 1-12
Link Troubleshooting 1-13
    Isolating Hardware Failures 1-13
    LAN Links 1-14
        Troubleshooting with LAN/1000 Node Manager 1-14
        Troubleshooting with NSINF 1-15
        Troubleshooting with NRLIST 1-15
    HDLC Links 1-16
        HDLC Card Loop-Back Hoods 1-16
    Bisync Links 1-17
        DSTEST 1-18

Chapter 2
NS Information Utility, NSINF

Overview 2-1
NSINF Runstring 2-2
    Command Summary 2-3
?—NSINF Main Menu 2-4
A—Local Name and Addresses 2-5
C—Configured Resources 2-7
I—List all NS-ARPA/1000 LUs 2-10
L—Information on an NS-ARPA LU 2-11
M—Message Accounting 2-16
N—Nodal Routing Vector 2-18
P—Individual Program Information 2-19
R—Rerouting Information 2-22

S—Remote Session 2-23
T—List Tables 2-24
Master List Entries 2-25
Slave List Entries 2-25
Process Number List Entries 2-26
U—NS-ARPA Utility Status and Statistics 2-27
V—DS/1000-IV Values 2-28
W—Information on TELNET Sessions 2-30

Chapter 3
Level 3 Loop Back Diagnostic, PING

PING 3-1

Chapter 4
NS-ARPA Event Logging

Overview 4-1
Using the Event Logging Utilities 4-2
EVMON 4-2
BREVL 4-4
LOGCHG 4-5
EVMON Output 4-7
EVMON Header Fields 4-7
Log Record Header Fields 4-7
Log Message Field 4-9

Chapter 5
NS-ARPA Message Tracing

Overview 5-1
Using Message Tracing 5-2
NSTRC 5-3
BRTRC 5-5
FMTRC 5-6
VMA Size 5-7
Error Handling 5-7
Dialogue Syntax 5-7
Answer Files 5-7
FMTRC Dialogue 5-8
Duplicate Formatted File 5-12
Dialogue Examples 5-13
Interactive 5-13
Answer File 5-14
Trace File Formats 5-15
Trace Record Formats 5-16
NICE Formatted Records 5-17
Socket Trace Records 5-19
Socket Trace Record Fields 5-19
DS/1000-IV Compatible Services 5-20
LAN Trace Records 5-21
LAN Trace Record Fields 5-21
LAN Header 5-22
Router/1000 Records 5-24

Router/1000 Trace Record Fields 5-24
Router/1000 Header 5-25
HDLC Link Up Messages 5-26

Chapter 6
Nodal Registry List Utility, NRLIST

Overview 6-1
Help Menu 6-1
Output Files 6-2
Output Formats 6-2
Interactive Display 6-2
Banner 6-2
Dump Mode 6-3
Raw Mode 6-3

Chapter 7
DS/1000-IV (RTE-MPE) Message Tracing Utilities

Overview 7-1
LOG3K 7-2
LOG3K Operation 7-2
LOG3K Command Summary 7-3
Using a Magnetic Tape 7-3
?? 7-4
EN, EX, /E, NO 7-5
LU 7-6
TY 7-7
UP 7-8
LOG3K Example 7-9
TRC3K 7-10
TRC3K Operation 7-10
TRC3K Command Summary 7-11
?? 7-12
EXIT 7-13
FORMAT 7-14
LIST 7-15
PRINT 7-16
SET 7-17
TRC3K Example 7-18
Message Classes and Stream Types 7-20

Chapter 8
Modification Utility, DSMOD

Overview 8-1
DSMOD 8-1
DSMOD Command Summary 8-2
Command Files 8-3
??—Command List 8-4
/A 8-5
CN 8-6
DI 8-7
/E 8-8

/I 8-9
/L 8-10
/N 8-11
/S 8-13
/T 8-14
/T Example 8-14
Network Timeouts 8-15

Chapter 9
Principles of Operation

Overview 9-1
NS-ARPA/1000 Path Flow 9-1
NS-ARPA Protocol Modules 9-1
Logical Data Flow 9-3
Path Records 9-4
Paths 9-6
Nodal Path Reports and Connect Site Path Reports 9-7
VC Connection Establishment 9-8
Probe 9-10
Physical Data Flow 9-12
Messages 9-12
Outbound Messages 9-12
Inbound Messages 9-13
Message Size 9-13
DS/1000-IV Compatible Services Internals 9-14
Data Flow through the DS/1000-IV Software 9-14
UPLIN and the Transaction Control Block (TCB) 9-18
Class I/O Rethreads 9-19
Failures on HDLC Links with Dynamic Rerouting 9-19
Failures on Non-Rerouting HDLC Links 9-20
HDLC Message Processing 9-20
I/O Completion Processing 9-21
Slave-Side Processing 9-22
X.25 Links 9-23
#MAST 9-23
GRPM 9-25
Incoming Requests 9-25
Outgoing Line Completions 9-26
Dynamic Rerouting 9-26
Cost Matrix 9-26
Link Vector 9-27
Nodal Routing Vector (NRV) 9-27
Dynamic Rerouting Processing 9-27
Link Failure Procedure 9-27
Up Link Procedure 9-28
Network Update Message Procedure 9-28
The #SEND Process 9-28
Compatibility 9-28
Message Accounting 9-30
Lost Message Retransmission 9-31
Message Accounting Terminology 9-31
Message Accounting Initial Connection 9-32
Message Accounting Sequence Number Assignment 9-32
Message Accounting Receive Processing 9-33

Message Accounting Message Acknowledgement 9-33
Message Accounting Message Timeout 9-34
Message Accounting Message Retransmission 9-35
Message Accounting Disconnect 9-35
Message Accounting GRPM Interface 9-36
QUEUE 9-36
#SLAV 9-37
#GET 9-38
UPLIN 9-39
Accessing Nodes with Session Monitor 9-40
REMAT Session Module (#RMSM) 9-40
Distributed LOGON/LOGOFF (DLGON) 9-40
Network Account Table (#NAT) 9-41
Master Program Session Module (#MSSM) 9-41
Router/1000 Message Header 9-41
Program-to-Program Communication (PTOPM) 9-42
POPEN Processing 9-42
PREAD, PWRIT, and PCONT Processing 9-42
PNRPY Processing 9-43
PCLOS Processing 9-43
GET Processing 9-43
ACEPT Processing 9-43
REJCT Processing 9-43
FINIS Processing 9-44
Slave List Processing 9-44
Slave Off Processing 9-44
Remote File Access Monitor (RFAM) 9-44
DEXEC 9-45
Driver ID*66 9-45
HDLC Link Connect Processing 9-45
HDLC Retry Processing 9-46
HDLC Link Down Processing 9-47
Read and Write Processing 9-47
DSLIN Processing 9-47
DS/1000-IV Compatible Services Communication (RTE-MPE) 9-48
DS/3000 Messages 9-49
DS/3000 Bisync Protocol 9-50
ID*66, CXL66 and PSI Links 9-51
ID*66 I/O Requests for Bisync 9-52
QUEX and QUEZ for the PSI Link 9-52
X.25 Links 9-53
RTE to MPE Master Side Communication 9-54
HELLO, BYE, and Master Subroutines 9-54
D3KMS Subroutine 9-56
RTE to MPE Slave Side Communication 9-58
Request and Reply Converters 9-59
CNSLM 9-60
HP 1000 to HP 3000 Clean-Up 9-60
Memory Manager 9-61
Overview 9-61
Global Area 9-61
Tables Area 9-62
Buffer Area 9-62
Network File Transfer 9-62
Modules 9-62

NFT Data Structures in DSAM 9-63
General Flow 9-63
Message Flow Summary 9-67
DSCOPY 9-69
NFTMN 9-69
PRODC 9-70
Main Control Loop 9-70
Sending File Data 9-70
Directories 9-71
PRDC1 9-71
Error Handling 9-72
Termination Handling 9-72
CONSM 9-72
Session Management 9-73
Session Termination 9-73
Error Handling 9-73
Abort Handling 9-73

List of Illustrations

Figure 1-1 Summary of Troubleshooting Utilities and Diagnostics 1-2
Figure 4-1 EVMON Output 4-7
Figure 5-1 Example Trace File Header 5-15
Figure 5-2 Example TCP Trace Record in Nice Format 5-17
Figure 5-3 Example Probe Trace Record in Nice Format 5-17
Figure 5-4 Example NFT Trace Record in Nice Format 5-18
Figure 5-5 Example ARP Request Trace Record in Nice Format 5-18
Figure 5-6 Socket Trace Record Format 5-19
Figure 5-7 LAN Trace Record Format 5-21
Figure 5-8 Router/1000 Trace Record Format 5-24
Figure 9-1 INPRO and OUTPRO Protocol Modules 9-3
Figure 9-2 Conceptual Message Flow 9-4
Figure 9-3 Example of Address Information in Path Records and Message Headers 9-6
Figure 9-4 Example of Sockets, Path Records and Paths in a Node 9-7
Figure 9-5 Sending Node Data Flow, HDLC Link 9-15
Figure 9-6 Sending Node Data Flow, 802 Link 9-15
Figure 9-7 Receiving Node Data Flow, HDLC Link 9-16
Figure 9-8 Receiving Node Data Flow, 802 Link 9-16
Figure 9-9 Rerouting Topology Considerations 9-29
Figure 9-10 Rerouting Topology Considerations 9-29
Figure 9-11 Rerouting Topology Considerations 9-30
Figure 9-12 Message Accounting Receive Processing Window 9-33
Figure 9-13 Message Accounting Acknowledgement Window 9-34
Figure 9-14 Message Accounting Incrementation 9-34
Figure 9-15 Overview of NS-ARPA/1000 to NS/3000 Communications 9-48
Figure 9-16 Continuation Data Buffer 9-50
Figure 9-17 DS/3000 Bisync Exchange 9-51
Figure 9-18 Communications Access Software for PSI 9-51
Figure 9-19 DS/1000-IV to DS/3000 Master Side Modules 9-56
Figure 9-20 RMOTE and D3KMS Operation 9-58
Figure 9-21 DS/1000-IV to DS/3000 Slave Side Modules 9-59
Figure 9-22 NFT Connection Establishment 9-66

Tables

Table 2-1	NSINF Commands	2-3
Table 4-1	Log Mask Event Classes	4-3
Table 7-1	LOG3K Commands	7-3
Table 7-2	TRC3K Commands	7-11
Table 7-3	Message Classes and Stream Types	7-20
Table 8-1	DSMOD Commands	8-2
Table 9-1	Path Record Allocation	9-5
Table 9-2	T1 Default Values	9-46
Table 9-3	PSI Communication Interfaces	9-48
Table 9-4	CNSLM Processing	9-60
Table 9-5	NFT Messages	9-68

1

Troubleshooting Guidelines

Overview

This section contains troubleshooting guidelines or checklists. Use this section as a guide to using the troubleshooting utilities and diagnostics described in the other sections of this manual. These utilities and diagnostics are as follows:

- *NSINF*—network information utility that provides information regarding the network and connections. Use to obtain an overview of network activity.
- *PING*—diagnostic that verifies the physical connection in a level 3 (IP level) loop back. Use as a quick check to other nodes.
- *EVMON*, *BREVL*, *LOGCHG*—event logging utilities which record networking events occurring at the local node.
- *NSTRC*, *BRTRC*, *FMTRC*—three parts of the message tracing utility which record networking messages between two nodes. Use to see if and what messages are sent between two nodes.
- *NRLIST*—nodal registry list utility which lists NPRs, Nodal Path Reports. Nodal Path Reports contain IP addresses, LAN station addresses, and other protocol information.
- *LOG3K* and *TRC3K*—DS/1000-IV (RTE-MPE) message tracing utilities which record networking messages over bisync and X.25 links.
- *DSMOD*—DS/1000-IV modification utility to alter DS/1000-IV initialization and/or timing parameters.

The LAN node management software is documented in the *HP 12076A LAN/1000 Link Node Manager's Manual*, part number 12076-90002. The node management software helps troubleshoot from the LAN driver level of the local node to another node.

When using the troubleshooting checklists in this section, keep in mind the scope of the problem. When you note what does not work, you should also document what *does* work. The comparison of what does and what does not work can often help you pinpoint a problem.

You should think about the following questions when isolating problems.

- How much of the network is affected?

- Does the problem affect one node's ability to connect to one particular node or all nodes?
- Is the problem with NS Common Services, ARPA Services, or with DS/1000-IV Compatible Services?
- Which applications work and which do not?

Note If your network includes connection to other types of computer systems, refer to the documentation for those systems to help you with cross-system troubleshooting.

You must keep up-to-date copies of the system generation and NSINIT answer files, installation, BOOT, and WELCOME files. Should you have any problems with your system, HP will require these files.

This section contains the following subsections:

- Summary of Checklists
- Configuration or Initialization Problems
- Other NS-ARPA Problems
- Link Troubleshooting

Figure 1-1 shows the extent of each troubleshooting utility or diagnostic.

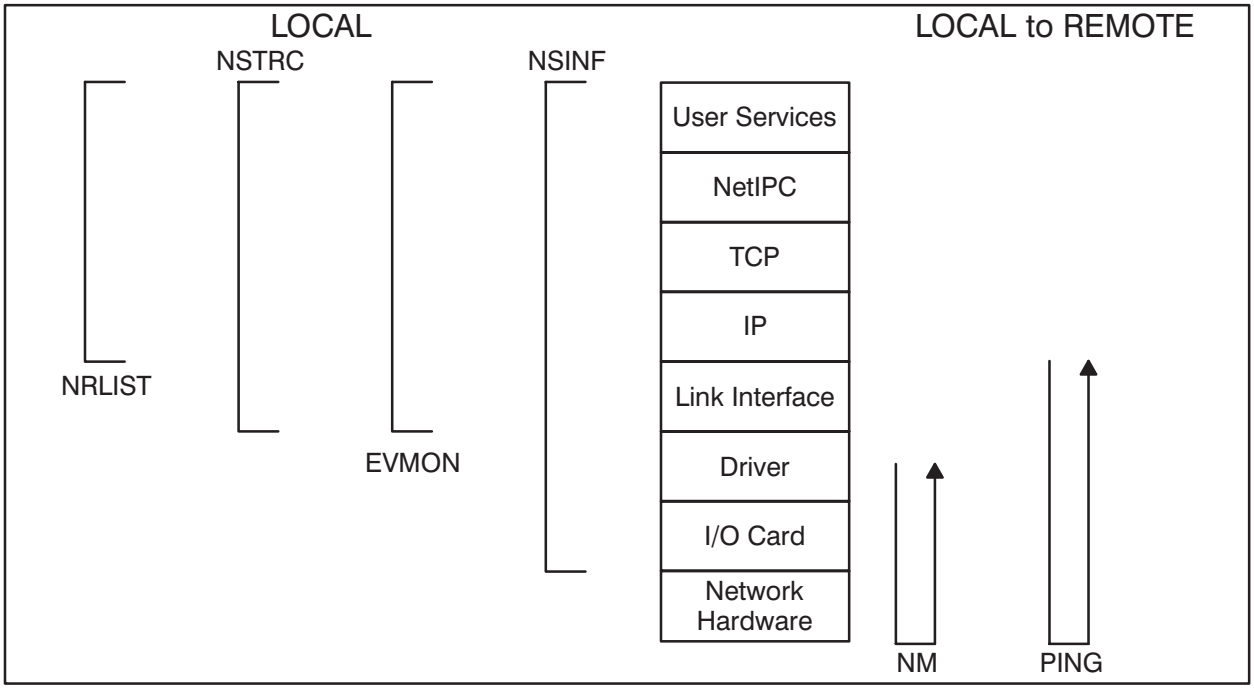


Figure 1-1. Summary of Troubleshooting Utilities and Diagnostics

Summary of Checklists

This subsection contains two checklists: one for configuration or initialization problems, and one for other problems. If you exhaust one checklist, go through the other checklist. In many cases, deciding where a problem lies can be difficult.

The actions you should take for the checklist items are described in the following subsections.

Checklist for Configuration or Initialization Problems

A. Error messages from initialization

B. Hardware/Generation verification:

> LUs, Select Codes, cards, card self test, cables, data link layer software (X.25, LAN/1000). Refer to "Link Troubleshooting" later in this section.

C. Proper software installed

D. Local Addresses consistent between NSINIT, NRINIT, LAN/1000

E. Appropriate table sizes configured

F. RTE-A resources are sufficient

G. No mixing of software versions (RTE, DS/1000-IV, NS-ARPA/1000)

H. Cross Network Addresses check out (draw reverse map)

Checklist for Other NS-ARPA Problems

A. General Guidelines

B. NetIPC or BSD IPC level problems

C. NetIPC/TCP or BSD IPC/TCP problems

D. DS/1000-IV Compatible Services problems

E. NFT problems

F. FTP problems

G. TELNET problems

Configuration or Initialization Problems

The items in this checklist start from the hardware level and progress up. Any subsection can be used in isolation, but the checklist will be most effective if you read it in its entirety, at least initially.

Initialization Error Messages

Look at the error messages that NSINIT and NRINIT printed. Verify that these programs completed successfully on every node in the network. If they did not run successfully on a given node, that node will be unable to communicate with any other node. Use the *NS-ARPA/1000 Error Message and Recovery Manual* to isolate the initialization problems indicated by the error messages.

Also look for error messages on the system console. Some NS-ARPA programs print error messages on the console if they encounter errors when they begin.

If initialization did not complete successfully, it can be difficult to determine the exact line in the answer file that caused NSINIT to fail. Keep in mind that NSINIT prompts are dependent on previous responses. Changing one response may cause different questions to be asked later on in the initialization dialogue.

One common change that alters the questions asked is changing the response to the following prompt: `"Do you want any DS/1000-IV Compatible Services?"`. If the node has DS/1000-IV Compatible Services, NSINIT asks additional questions to build the tables and start the monitors for these services.

If you change an answer file incorrectly, it may be difficult to determine the exact prompt and response that caused the error. The simplest approach is to create a new output file to ensure that the responses in the file will match the NSINIT dialogue. To do this, perform the following steps:

1. Edit the input file, delete all the lines in the file that contain only comments. Most of these lines are NSINIT prompts. Save the changed file under a different name (for example, `CHGNSIN`).

2. Run NSINIT with `CHGNSIN` as the input file, and another file, such as `CHGNSOUT` as the output file. NSINIT will still terminate with input errors, but you now have a record of where the program failed.

3. Edit `CHGNSOUT` and look at the last lines in the file. The file will contain the NSINIT prompts, along with the responses NSINIT read from `CHGNSIN`. Find the first place where the prompt and response do not match. This is the problem.

4. Make the necessary corrections in the original input file, and rerun NSINIT.

Hardware and Generation Verification

1. Verify that the LUs you initialized contain the cards you said they did, that the select code is correct, and that the card is plugged into the backplane.

2. Verify that all cables are connected.

3. Verify the hardware is up, and that the data link layer connection is all right. Check the card self test if one exists. Refer to the link interface card manuals for this information.

   If you have a LAN link: Refer to the LAN/1000 manual set for the verification procedure. If you are changing any station addresses, make sure that all station address configurations match. Refer to the subsection "LAN Links" later in this section for more troubleshooting guidelines.

   If you have an X.25 link: Refer to the X.25 manual set for the verification procedure. Also make sure that you initialized the X.25 subsystem (via XINIT) before initializing NS-ARPA/1000. Note that the file `/NS1000/CMD/INSTALL_NS1000.CMD` has X.25 variables which must be set properly for HP 1000-to-HP 3000 X.25 connections. Refer to this file and the *NS-ARPA/1000 Generation and Initialization Manual*.

   If you have an HDLC link: Refer to the subsection "HDLC Links" for more troubleshooting guidelines.

   If you have a Bisync link: Refer to the subsection "Bisync Links" for more troubleshooting guidelines.

4. Verify that the RTE system generation is correct; refer to the appropriate RTE manuals. In particular, make sure the drivers and the DVT extensions are correct.

Proper Software in Place

Using the tables in the section "Internal Resources" in *NS-ARPA/1000 Generation and Initialization Manual*, make a list of all the NS-ARPA software modules that should be present in a given node after startup. Enter the RTE `WH,PA` or `WH,AL` command to see which programs are actually present. All programs should be waiting, either on a Resource Number (a socket), or on a class number, except those noted in the paragraphs below. If a particular program is NOT present in the system after initialization, look for error messages or event log messages that NSINIT or NRINIT may have generated. Look up any messages in the *NS-ARPA/1000 Error Message and Recovery Manual*.

The following programs will be dormant most of the time:

QUEUE (if there are HDLC, Bisync, or X.25 links)
#SEND (if there are Dynamic Rerouting links)

The following programs will be in the time list most of the time:

UPLIN
MATIC

Remember that the following programs must be present in *every* NS-ARPA system:

INPRO
OUTPRO
UPLIN

Refer to the section "Internal Resources" in the *NS-ARPA/1000 Generation and Initialization Manual* for more information.

If `/SYSTEM/NSERRS.MSG` is not present or is not current, FTP or TELNET may print only error numbers instead of ASCII error messages.

Local Address Consistency

Local addresses are entered at initialization time through NSINIT and NRINIT (if required). If the addresses do not match, the node will be unable to communicate with other nodes under some circumstances. You should verify that all information about the node is the same. You can check the addresses configured via NSINIT with the NSINF `A` command (the list local name and addresses command). You can check the NRINIT configuration through NRLIST. Refer to the NSINF and NRLIST sections in this manual for more information.

1. Verification of NSINIT and NRINIT information for: local node name, local IP address(es), LAN station address(es), and subnet mask.

   a. Local node name. The local node name is the first question after the menu in the NSINIT dialogue. This name must also be in the Nodal Registry. Verify that the two names match.

   b. Local IP address. The local IP address is entered in the DCN section of the NSINIT dialogue. This address must match the address in the Nodal Registry for the local node.

   c. LAN Station Address. The LAN station address(es) can also be entered via NSINIT and NRINIT. If you entered the LAN station address via NRINIT, it must match the one used in NSINIT.

      Also note that if you do not configure the default (card) station address for NSINIT and NRINIT, the alternate address is only changed in RAM. You may want to configure the alternate address in NOVRAM via the LAN/1000 Node Manager program.

      The LAN station address is optional. If it is not specified, the default (card) address will be used. Refer to the LAN/1000 manuals for more information.

   d. Subnet mask. The subnet mask is entered in the DCN section of the NSINIT dialogue. Check the subnet mask with the NSINF `A` command.

2. Specific information about the exact configuration of the node. You should verify this information via NSINF by examining NS-ARPA tables once initialization is complete. The configuration information should enable you to draw an accurate map of the node, complete with LI types and LU numbers, network boundaries and all node addresses and names. You can then compare this map with the one you drew when you planned the network. The maps should be the same.

The information you need is

the local address information and the link LU information for the LAN Link (use the NSINF `A` command)

the Router/1000 Nodal Routing Vector information (use the NSINF `N` command)

the IP Gateway Table (use the NSINF `A` command)

Router/1000 rerouting LUs (use the NSINF `R` command)

The map you draw will show how a single node views all the nodes in the internet and which links are used to get there.

Configured Table Sizes

Specifying values for table sizes smaller than the defaults has implications on how many NetIPC or BSD IPC connections can simultaneously exist. If you use the NSINIT minimum values, you can make one NetIPC or BSD IPC connection between two nodes. It also assumes no NFT or EVMON. These values should be considered bare minimums. HP does NOT recommend configuring the minimum values unless the customer is sophisticated.

In particular, the following tables are particularly sensitive to table sizes less than the defaults:

NS-ARPA programs (user records)

NS-ARPA sockets (sockets)

Connect-Site Path reports

The NSINF `C` command (list configured resources command) reports how many of the configured resources are in use at any given time.

RTE-A Resources

Make sure you have enough RTE system resources for your NS-ARPA needs even if you use the defaults. The following list shows the resources most likely to cause configuration problems. Refer to the "Internal Resources" section in the *NS-ARPA/1000 Generation and Initialization Manual* for a complete discussion of this topic.

RTE system resources most likely to cause configuration problems are as follows:

- ID Segments
- Resource Numbers
- Class Numbers
- size of MMINIT's SHEMA (`,,,DS`, which determines DSAM size)
- System Memory Block (SMB)

NS-ARPA will usually print an error message indicating what resources were unavailable. You should still evaluate the resources used, as occasionally the error indication may be masked by other symptoms, but will still exist.

You can use the NSINF `C` and `V` (list DS/1000-IV values) commands to print the class numbers and resource numbers in use by NS-ARPA at any given time. NSINIT prints the required SMB size and DSAM table size at the end of its initialization dialogue. For ID Segment, Class Number, and Resource Number requirements, refer to the section "Internal Resources" in *NS-ARPA/1000 Generation and Initialization Manual*.

Mixing Software Revisions

Mixing versions of NS-ARPA/1000 can have drastic ramifications for the reliability of the product. Do not mix software revisions.

Furthermore, do not attempt to mix DS/1000-IV modules with NS-ARPA/1000. Some of these modules have two or three versions, one for NS-ARPA/1000, one for ARPA/1000 (98170A product), and one for DS/1000-IV. If the wrong version is present in your system, the results will be unpredictable.

The symptoms may be as straightforward as a particular network service not working properly, or as drastic as the inability for any node to talk to any other node.

The revision levels of RTE-A and NS-ARPA/1000 must be up-to-date and supported revisions. Refer to the "NS-ARPA/1000 Cover Letter," part number 91790-91001, for a matrix of supported RTE and NS-ARPA/1000 revisions. Be sure to re-link the RTE modules TRFAS and DSRTR each time you install a new version of NS-ARPA/1000. Also re-link RDBAM (for Remote Image/1000) if it is used on your system.

Cross-Network Address Consistency

After all other configuration checklists have been used, the cross-network configuration check should be performed. The information here is also useful to doublecheck the internet configuration. In many cases, drawing the complete map of the internet from the NS-ARPA tables will catch simple mistakes in the configuration.

The time spent in verifying that the network is properly configured when it is first initialized could save the trouble of debugging through many software levels once an application is running.

1. Take the individual nodal pictures drawn for step D, and put them all together. Does the network map match the one you drew before configuring the internet? Make any necessary corrections, and repeat the process until the pictures are the same.

2. Verify that the information in the NRINIT file is correct. For example, suppose you have a network with nodes A, B, and C. Start at A, compare the addresses that it has for B in the Nodal Registry with the local addresses for B. Make sure they match. Verify the addresses A has for node C are also correct. Do this for every node.

   It may be easier to install the same Nodal Registry initialization file on every node in the network. That way, you need only verify the information at one node, since you know the

same information is present at every node. However, make sure that the line corresponding to the local node is commented out in the NRINIT file.

3. Duplicate Addresses: Check for two nodes on a given network having the same addresses or names. Unpredictable results can occur under these circumstances. The information to check is: Router/1000 address, IP address, local node name, and LAN station address.

4. GT: Use the NSINF `A` command to check the Gateway Tables (GTs) of all the nodes in the internet for consistency and completeness.

5. NRV Addresses: Any node added to a LAN must be added to the NRV (Nodal Routing Vector) of all the nodes in the internet with which the new node is to communicate, including DS/1000-IV nodes.

   Check the IP addresses specified in the NRV correspond to the node's RTR link. Check the rules for NRV entries in the "Network Configuration Planning" section of *NS-ARPA/1000 Generation and Initialization Manual*.

Other NS-ARPA Problems

This information is organized according to the utilities used to diagnose the problem. The first section discusses common user level errors.

When the list has been exhausted you should attempt to go through the configuration information. Any information needed in both lists is documented in the configuration category, but is also referenced here.

This list can be used in sections, but initially you should read the entire list.

General Guidelines

Some general guidelines about capturing information during a failure are listed below:

1. The window for examining path information is small. Protocols below NetIPC often are told to clean up path information well before an error indication is handed to the user. This means running NSINF when the problem seems to occur can be deceptive.

2. You can use NSTRC to capture information about a problem to be evaluated later. The information can then be formatted on a socket or message basis. Collecting the trace information will change the timing of NS-ARPA/1000, which means you can alter timing-critical problems by turning NSTRC on and off.

3. Information in the EVMON log file is often very detailed, but it can be an important debugging tool. First start with the mnemonic of the protocol reporting the problem. Once you know the protocol, you can use the information in the *NS-ARPA/1000 Error Message and Recovery Manual* to narrow down the specific source of the problem within the protocol.

NetIPC or BSD IPC Level Problems

Evaluate the error messages and log file entries produced by NetIPC or BSD IPC. Use the meanings and actions described in the *NS-ARPA/1000 Error Message and Recovery Manual*.

1. Changes in NetIPC software:

   a. As of Software Revision Code 5.0 or later, the NetIPC errors 64 and 65 have switched meanings. If you checked for NetIPC error 65 previously, you must change your code to check for NetIPC error 64 (and vice versa).

   b. As of Software Revision Code 5.0 or later, the DATA_WAIT flag of the `IPCRecv` call has been implemented. If you were waiting for a specified amount of data in `IPCRecv`, then you must now initialize the DATA_WAIT flag.

2. Common coding errors:

   a. Using a node or socket name that is not in a configuration for the node on which it is being run. Make sure the names that a program uses are configured in the node.

   b. Misunderstanding the use of parameters in a NetIPC or BSD IPC call. Carefully evaluate the parameters for the NetIPC or BSD IPC calls. Remember to look at the return parameters in addition to the error codes.

3. If you are using DEXEC to schedule the remote process, check the following items:

   a. You configured DS/1000-IV Compatible Services.

   b. The program to be scheduled has been loaded as a system utility and restored (`RP`ed).

   c. In the DEXEC call, all the letters in the program name are upper-case characters.

4. BSD IPC level problems:

   a. Watch out for coding errors in Pascal and FORTRAN. Pascal and FORTRAN users must make proper use of the `AddressOf` and `ByteAdrOf` routines when passing addresses as parameters to the BSD IPC routines. The type of parameter must also match the type expected by the routines.

   b. Resolve references to `errno` and `errno2` at compile time. Pascal users must search the `errnodec.rel` file in the `/NS1000/REL` directory to resolve references to `errno` and `errno2` at compile time.

   c. Resolve references to `errno` and `errno2` at link time. The NS-ARPA libraries, `BSD_CDS.LIB` and `BIGNS_CDS.LIB` must be searched before the C library.

NetIPC/TCP or BSD IPC Problems

It is often difficult to pinpoint a problem between TCP and NetIPC or TCP and BSD IPC. Various approaches are listed here. You can use them one at a time or in combination with each other. Refer to the appropriate section for more information about each utility.

1. NSTRC: The socket level formatting of a trace will be of the most use in diagnosing user level problems. Refer to the NSTRC and FMTRC discussion for more information.

   You should determine the Global Socket Descriptor (GSD) for each socket that a process owns before tracing at the socket level. Use the NSINF `P` command to get the GSD for a particular process.

2. You can use the NSINF `P` command to be sure NS-ARPA/1000 creates the proper resources (call socket, VC sockets) for the program.

3. The EVMON log file contains information about errors encountered. The file will help pinpoint the specific NS-ARPA module that encountered the problem.

4. Timeouts. Timeouts have two causes:

   a. Insufficient time for a message to be received by the other side. In this case, increasing the timeout value at the NetIPC level should correct the problem.

      Note that the timeout error reported by NetIPC is an interface timer between TCP and NetIPC. If TCP times out the error returned will be `retry count exhausted` (NetIPC error 90) or `aborted locally` (NetIPC error 34). A NetIPC error 34 may also indicate that the remote NetIPC process cannot get the resources (DSAM, resource number, socket record) it requires.

      TCP is an end-to-end protocol. An error like the one described above means the message did not get to the other side. The problem could be either on the local or remote side, or at an intermediate node.

   b. Configuration Problem. In this case, increasing the timeout value at NetIPC will make no difference; the message will still fail. Return to the configuration checklist to pursue the configuration problem.

DS/1000-IV Compatible Services Problems

If NS Common Services or ARPA Services work but DS/1000-IV Compatible Services do not, you may want to check the following items:

1. Software versions. Some modules have two versions, one for NS-ARPA/1000 and one for DS/1000-IV. Check that the NS-ARPA/1000 versions are installed rather than the DS/1000-IV versions. If you are using Transparent File Access, make sure that you re-link TRFAS and DSRTR each time you install a new version of NS-ARPA/1000.

2. If you are using DS/1000-IV Compatible Services (RTE-RTE) over LAN links, make sure that IFPM is present in the system.

NFT Problems

If NFT is not working, you may want to check the following NFT modules:

1. DSCOPY. DSCOPY must be installed in `/PROGRAMS`.
2. NFTMN. Before initializing the node, you must restore (`RP`) NFTMN.
3. DSCOPY.HLP. The DSCOPY help file `DSCOPY.HLP` must be installed in `/SYSTEM`.

In addition, you should check Nodal Registry entries (via NRLIST), and IP address and node name consistency throughout the internet.

Refer to the *NS-ARPA/1000 Error Message and Recovery Manual* for the meaning of any NFT error messages or event log messages.

FTP Problems

If FTP is not working, you may want to check the following FTP modules:

1. FTPSV. FTPSV must be installed in `/PROGRAMS`.
2. INETD. Before initializing the node, you must restore (`RP`) INETD. The configuration files used by INETD, `/ETC/INETD.CONF` and `/ETC/SERVICES`, must contain entries for FTP.
3. FTP.HLP. The FTP help file `FTP.HLP` must be installed in `/SYSTEM`.
4. `/SCRATCH` directory. FTP uses the `/SCRATCH` directory for its temporary files. Make sure the directory exists and that it is readable and writeable by all.

In addition, you should check Nodal Registry entries (via NRLIST), and IP address and node name consistency throughout the internet.

Refer to the *NS-ARPA/1000 Error Message and Recovery Manual* for the meaning of any FTP error messages or event log messages.

TELNET Problems

If TELNET is not working, you may want to check the following TELNET modules:

1. TNSRV. TNSRV must be installed in `/PROGRAMS`.
2. INETD. Before initializing the node, you must restore (`RP`) INETD. The configuration files used by INETD, `/ETC/INETD.CONF` and `/ETC/SERVICES`, must contain entries for TELNET.
3. TELNET.HLP. The TELNET help file `TELNET.HLP` must be installed in `/SYSTEM`.

The RTE-A modules, `%ALARM` and `%SIGNL` must be generated into your RTE-A system.

In addition, you should check Nodal Registry entries (via NRLIST), and IP address and node name consistency throughout the internet.

Refer to the *NS-ARPA/1000 Error Message and Recovery Manual* for the meaning of any TELNET error messages or event log messages.

Make sure the TELNET pseudo terminal LUs are generated and initialized correctly. Refer to the *NS-ARPA/1000 Generation and Initialization Manual*.

Link Troubleshooting

This section gives guidelines for troubleshooting LAN/1000, HDLC, and Bisync links.

Isolating Hardware Failures

There are three ways to isolate hardware failures:

- substitute spare pieces of equipment
- self-test
- the use of diagnostics.

Diagnostics for each piece of equipment are described in the hardware documentation covering the equipment.

In a store-and-forward environment, a failure at one node may cause error messages to appear at other nodes that are not directly connected to the node which failed. When troubleshooting, you must first determine which node has failed. The applications software should print the node number that reported the error (certain errors are reported by one node due to conditions at another), as well as the error code. Error codes are described in the *NS-ARPA/1000 Error Message and Recovery Manual*.

If hardware failure is a possible cause, use alternate equipment, if available, to isolate the failure as quickly as possible. The hardware diagnostics can be run quickly and are easier to perform than looking for bugs in software. However, some diagnostics are not designed to function in the RTE operating system environment and must be run with the computer in a stand-alone mode (real-time activities must cease). If alternate nodes exist, a node may be taken off-line to run diagnostic checks. If not, there may be planned shutdown periods which can be utilized.

You can usually isolate the problem to the HP 1000-to-HP 1000 link level by using REMAT (if DS/1000-IV Compatible Services are installed). Send the REMAT `TE` command from each of the nodes you suspect to each of their neighbors. If communications function properly to or from any node, the chances are good that the computer and disk (if it exists) are operational. You may then troubleshoot the interfaces, cables, and modems (if phone lines are used). To use the `TE` procedure, at least one side of every link needs an operator (but not necessarily both sides). However, if you notice that you can't transmit or receive messages from another node, it is not always possible to determine whether the computer or the link is at fault. The error message will help resolve the situation.

Intermittent failures can be detected using the same procedures above, but use the `ST` command instead of the `TE` command using a fairly large file. Sending a large file should exercise the

communications line and interface cards enough that an intermittent failure can be identified. Loose interface card hoods or damaged cables can be a source of intermittent failures. Try wiggling them a little during the test.

The NSINF `L` command will show the line error statistics which are useful in determining if the line has deteriorated noticeably. Refer to the "NSINF" section for these statistics.

LAN Links

This subsection gives guidelines for troubleshooting LAN/1000 links used with NS-ARPA/1000. Before troubleshooting LAN/1000 with NS-ARPA/1000, you should verify that the LAN/1000 subsystem works in isolation. Refer to the following LAN/1000 manuals for information on LAN/1000 troubleshooting.

LAN Cable and Accessories Installation Manual
HP 12076A LAN/1000 Link Installation Manual, part number 12076-90001
HP 12076A LAN/1000 Link Node Manager's Manual, part number 12076-90002

Troubleshooting with LAN/1000 Node Manager

Task	LAN/1000 Node Manager Command
Verify communication with card.	`TC[,`*`ADR`*`][,`*`LU#`*`][,`*`Rep`*`]`
Verify communication with link.	`EL[,`*`ADR`*`][,`*`LU#`*`][,`*`Rep`*`]`
Read card statistics. They should all be 0 at this point, but will be needed later.	`RS[,`*`ADR`*`][,`*`LU#`*`]`
Verify communication with remote card.	`XID[,`*`ADR`*`],00[,`*`LU#`*`][,`*`Rep`*`]`
Read card statistics if `XID` failed, to determine where packet was lost.	`RS[,`*`ADR`*`][,`*`LU#`*`]`
Verify communication with remote Node Manager.	`XID[,`*`ADR`*`],F8[,`*`LU#`*`][,`*`Rep`*`]`
Read card statistics.	`RS[,`*`ADR`*`][,`*`LU#`*`]`

If attempts to verify communication with the remote card or remote Node Manager fail, the statistics from the card will help to pinpoint the problem area. Refer to *HP 12076A LAN/1000 Link Node Manager's Manual* for information on interpreting these statistics. You should also check the following item:

- LAN card LU assignments in the system generation. The transmit (user) LU should be the lower LU, an even number.

If all above tests pass, then there may be a problem in the interface between LAN/1000 and NS-ARPA/1000, and you should check the following items using the Node Manager `RC` (Read Link Configuration) command:

Task	LAN/1000 Node Manager Command
Verify INPRO's class number in the driver table. It should be at DSAP 6 and 252 for an 802 LI.	`RC[,`*`ADR`*`],8[,,`*`LU#`*`]`
Verify the presence of multicast addresses for an 802 LI.	`RC[,`*`ADR`*`],2[,,`*`LU#`*`]`
Verify that the card is enabled to accept multicast packets and the station address is correct. The `receive packet filter` must be 4 or greater for 802-only LI. For Ethernet-only LI, the value should be 2, 3, 6, or 7. For LAN (both Ethernet and 802) LI, the value should be 6 or greater.	`RC[,`*`ADR`*`],A[,,`*`LU#`*`]`

Troubleshooting with NSINF

Task	NSINF Command
Verify the station address is the same as that on each card.	`A`
Verify that the multicast addresses are the same.	`A`
Verify that the LU is up.	`L`
Verify that the NS-ARPA/1000 software also thinks the LU is up.	`A`

Troubleshooting with NRLIST

You should run NRLIST and verify that the station address of the remote node matches the address reported by LAN/1000 Node Manager and NSINF at that node.

HDLC Links

The HDLC cards have built-in self-test firmware that executes whenever the card is reset (e.g., power-up, preset, CLC 0, etc.). During a preset or CLC 0, all four LEDs on the front edge of the card go on. During the self-test you will see LED 0 off and all other LEDs on. Upon completion of the self test, if no errors occurred, the lights will all go off.

The NS-ARPA software will enable this link, signalling it to connect with the card on the other end of the link. If the card can connect, LED 0 will light, indicating the link is initialized and functioning. This means the NS-ARPA driver was able to communicate with the card and the card has communicated with the card at the other end of the link. You must also check the lighting sequence on the other interface to see that two-way communication has been established. This light may also be checked with the NSINF `L` command. Look for a `1` in the `Connect Req.` and `Connected` fields.

The NSINF `L` command should also be used to examine the firmware Revision Code to determine that both cards have the same firmware. The baud rate and internal/external clock switch settings are also displayed and must be the same at both ends of the link in order to function properly.

This command also prints card statistics which are useful in evaluating the quality and predicting failures of a link.

HDLC Card Loop-Back Hoods

All HDLC cards are supplied with a loop-back verifier hood. Since the cards are full-duplex, the installation of the hood allows the output port of the card to communicate directly with the input port of the same card.

If, after installation of the firmware, self-test is not successful (lights on the board do not go out), performing the loop-back tests will be of no value. The purpose of the loop-back test is to verify that the line drivers, receivers, and backplane are functioning properly. It is also used to determine if the NS-ARPA/1000 software and RTE operating system have been properly generated.

To utilize the loop-back feature, install the proper hood on the I/O card which you wish to test. (There are different hoods for the modem and direct-connect boards.) The power must be turned off before connecting the hood. The hood is installed so that the words on the hood match the directions of the components. Apply power. The board should, again, successfully complete the self-test sequence.

Initialize NS-ARPA by running NSINIT; LED 0 should light. If not, check your generation and your NSINIT initialization file; you may be enabling the wrong LU. The card has not been told to connect by the driver or download firmware. Check the following items:

- Did you specify the correct LUs during initialization?
- Is the card select code indicated correctly in the generation?
- Are the card select code switches set correctly?

Once you have determined that a faulty link exists, the following possibilities may also exist:

- One or both interface cards may be bad.

- The cable may have an open- or short-circuit.
- Modem may have failed (phone line connections only).
- Line may be extremely noisy, or has drifted considerably from nominal parameters.

The failure could exist at either side. If spares exist, the fastest way to identify the problem is to begin substituting cables, interfaces, and modems, one piece at a time. If your RTE was generated with LU, DVT, and IFT entries for a spare interface board, then you may be able to switch over the LU assignment of the bad link to the spare entry, move the cable to the spare interface, and enable the link using DSMOD. Limitations on this are:

- The spare interface board must be the same type as the suspect one (i.e., you cannot substitute a modem interface for a hardwire interface, or vice versa, unless substitution is made on both sides).
- Interfaces on both sides of the link must be configured identically (speed, frame size, etc.). If not, you must remove the I/O board to set the switches properly. Be sure to remove power from the I/O interface first. You must shut down the computer to change the switches, since you must turn off the power to the I/O interfaces before removing any of them. Upon restoration of power, a power-fail restart will be made automatically (HP computers that support NS-ARPA/1000 require battery backup for this) if the power-fail software has been set up properly. UPLIN will automatically re-enable the line within 5 seconds. (See the interface hardware manual for a description of the switch settings.) The HDLC and modem BISYNC boards allow the speed settings, etc., to be checked on-line; if incorrect you must shut down.
- The spare communications interface must have the same subchannel and the same line timeout as the one it is replacing.

If substitution solves the problem, proceed according to the applicable service procedures depending on whether your HP warranty is still in effect, a maintenance contract is in effect, or you are using your own in-house repair facility.

Bisync Links

When troubleshooting an HP 1000 to HP 3000 Bisync link, check the following items:

- the PSI Bisync card.
- the LU, IFT, DVT and SC mappings.
- the physical connection between the HP 1000 and the HP 3000 (the cable or modem).
- software compatibility between the HP 1000 and the HP 3000 (code versions, speed, communication buffer size). Both systems must have the most recent versions of the software. Refer to the NS/3000 or DS/3000 manual set for more information on checking this information on the HP 3000.
- all the necessary DS/1000-IV Compatible Services monitors are running.
- the HP 3000 is ready for communication.

The first step in troubleshooting a Bisync link is to test the PSI card. Run the card through the card self test and diagnostic hood self-tests. (Refer to the appropriate firmware manual for instructions.)

Next, run NSINF and issue the `L` command. If you get the message `DRIVER REPORTS ERROR`, either the system cannot communicate with the driver and/or the driver cannot communicate with the card. Check the IFT, DVT, LU, SC mappings. Check that you enabled the correct LU at network initialization time. Test the card again.

When you can issue the NSINF `L` command without getting the `DRIVER REPORTS ERROR` message, note the value for `SPEED` to compare later with the HP 3000.

Next, run DSLIN:

- If you run DSLIN in primary mode and get the message `PRIMARY CONNECT TIMED OUT`, DSLIN was unable to communicate with the PSI card. Check the interface mappings again.
- If you run DSLIN and get the message `LINE IS UP WITH BUFFER SIZE` *nnn*, the PSI link should be available for 1000 to 3000 communication. You may want to check the buffer size with NSINF.
- If you run DSLIN and get the message `LINE IS UP BUT HP3000 NOT REPLYING`, check the cable or modems. Be sure that the `DSCONTROL` command was issued at the HP 3000. See if QUEX and QUEZ are active (state 3). If you ran DSLIN in secondary mode, you must also check that the `DSLINE` command is issued at the HP 3000.
- If problems persist when you run DSLIN, test the low-level (Bisync) link. To do this, issue the NSINF `L` command. Look at the PSI card's `Connected` and `Connect Req.` fields. If there is a zero in both fields, the system and the card may not be communicating. Using NSINF, check the counts in the card's TTD, NAK, and WACK fields. If the counts are incremented when you run DSLIN, the card and the software are communicating. However, the Bisync protocol is in error recovery mode. To clear the error, reset both cards. (On the HP 1000, run DSMOD and use the `/L` command. This closes the line, then re-enables the card.) Run the cards through self-tests.
- If there is a zero in the `Connected` field and a one in the `Connect Req.` field, create a file to use as a log file. Run LOG3K and set up the log file. Run DSLIN in primary mode. Run LOG3K and close the log file. Run TRC3K. Check that an initialization request was sent by the HP 1000. If not, check your card again, including its TTD, NAK, and WACK counts. If an initialization request was sent, see if it was received by the HP 3000. If so, troubleshoot the HP 3000. If it was not received by the HP 3000, troubleshoot the HP 1000 also. Also see if QUEX and QUEZ are active (state 3).
- If there is a one (1) in the `Connected` and `Connect Req.` fields, your Bisync link should be available for RTE to MPE communication. If not, set up a log file as above and trace the messages sent.

DSTEST

A procedure for testing the HP 3000-side communications hardware, called DSTEST, is described in the HP 3000 documentation set. On the HP 1000 side, the slave program DSTES must be restored using the `RP` command in the system session.

2

NS Information Utility, NSINF

Overview

This section explains how to use the NS-ARPA/1000 network information utility, NSINF. NSINF prints the following information about the local node:

- configured resources
- the local node's name, address and Gateway Table (GT)
- DS/1000-IV Compatible Services tables
- Message Accounting statistics
- Nodal Routing Vector (NRV) entries
- rerouting (Router/1000) links
- DS/1000-IV Compatible Services values, including resource numbers used, assigned class numbers, timeout values, and the LU table for links to HP 3000s that support DS/1000-IV Compatible Services
- NS-ARPA/1000 interface card LUs
- NS-ARPA message tracing and event logging utilities
- the socket and protocol records used by a program
- open socket addresses, connection states, and owners
- TELNET virtual terminal sessions

NSINF Runstring

```
[RU,]NSINF
```

NSINF is an interactive program. When you run NSINF, it will prompt you for a command as follows:

```
NSInf>
```

Enter `?` and NSINF will print its main menu (described later in subsection "?—Main Menu") with a list of valid commands. Table 2-1 summarizes the NSINF commands.

To exit NSINF, enter `E` in response to the `NSInf>` prompt.

If NSINF has more than a screenful of information to display, it will print one screen and then ask you if you want to print more information as follows:

```
More...
```

The following replies are allowed:

- Enter RETURN, a space, or a plus (+) to display the next screen of information.
- Enter `A` or `Q` to abort the display.
- Enter `Z` to suspend NSINF. Entering any character will then invoke the RTE system prompt `CM>`. Enter `GO` at the RTE system prompt to continue the display.
- All other characters will have no affect on continuing the display.

Command Summary

Table 2-1 summarizes the NSINF commands.

Table 2-1. NSINF Commands

Command	Description
?	Prints the NSINF main menu.
A	Prints the local node's name, addresses and Gateway Table (GT) as configured via NSINIT. Use to check the gateway or routing configuration.
C	Prints the configured resources for the NS Common Services and ARPA Services. Use to check if there are enough resources available.
I	Prints the transmit LUs of the NS-ARPA/1000 link interface cards generated in the system. Use to verify the card configuration.
L	Reads the NS-ARPA link interface card statistics and IFT extent word information. NSINF will prompt you for a link interface card LU. Use to verify the card configuration.
M	Prints Message Accounting (MA) information for each node in the NRV with MA enabled.
N	Prints the Nodal Routing Vector (NRV).
P	Prints socket record and protocol path record information for the sockets owned by a program. NSINF will prompt you for a program name. Use to troubleshoot a user program.
R	Prints information about the local node's Router/1000 links.
S	Prints information about the open VC and CALL sockets.
T	Prints the number entries in the Master Transaction Control Block (TCB) list, Slave TCB list and Process Number Lists (PNL).
U	Prints information about the NS-ARPA message tracing utility (NSTRC) and the NS-ARPA event logging monitor (EVMON). Use to check the status of tracing or logging.
V	Prints information about NS-ARPA/1000 resource numbers, information about DS/1000-IV Compatible Services system resources and configuration values, and LU table and links used for DS/1000-IV Compatible Services (RTE-MPE).
W	Prints information about TELNET virtual terminal sessions, both initiated at and remotely connected to the local node. Use to troubleshoot TELNET.
Display Commands	When more than one screenful of information needs to be displayed, NSINF prompts you to continue or to end the display. Enter one of the following commands: + - displays the next screen of information RETURN - displays the next screen of information space - displays the next screen of information A - aborts the display Q - aborts the display Z - suspends NSINF display

?—NSINF Main Menu

NSINF prints the following menu when you enter `?` in response to its prompt, or any invalid command.

```
    CI> nsinf

    NSInf > ?

                    NSInf Main Menu ('e' to exit)

Configuration Information
    C   Configured Resources
    A   Local Name and Addresses

DS/1000-IV Information
    T   List Tables
    M   Message Accounting
    N   Nodal Routing Vector
    R   Rerouting information
    V   DS/1000-IV Values

Status and Statistics
    I   List all NS-ARPA/1000 LUs
    L   Detailed information on an NS-ARPA LU
    U   NS-ARPA Utility Status and Statistics
    P   Individual Program Information
    S   Sockets
    W   Information on TELNET sessions
```

A—Local Name and Addresses

The Local Name and Addresses command prints the local node's name, addresses, and Gateway Table (GT) information derived from the NSINIT configuration. Use this command to check the gateway or routing configuration.

Example

```
                          LOCAL NAME AND ADDRESSES

 Local Name: CRICKET.OFFLINE.HP            Router Address 572

IP address        LU Status Type  Station address    Multicast addresses
---------------  --- ------ ----  -----------------  -------------------
192.006.001.002  134   UP    LAN  08-00-09-00-02-7C  09-00-09-00-00-01
                                                     09-00-09-00-00-02
192.006.250.002   36   UP    RTR

                                GATEWAY TABLE

Destination net       Gateway       Down PID  Seg Size  Hops  Subnetwork Mask
---------------  ----------------   --------  --------  ----  ---------------
192.006.001.000      local net      IEEE-802    1490       0  255.255.255.000
192.006.250.000      local net         RTR      8000       0  255.255.255.000
192.006.251.000   192.006.250.003      RTR      8000     100  000.000.000.000
```

Local Name and Address Field Definitions

Local Name—the local node's fully-qualified name. *Router Address*—the local node's Router/1000 address. Used for DS/1000-IV services and routing over Router/1000 links.

IP address—the local node's IP address or addresses.

LU—the LU of the link associated with the IP address. If this value is `-99`, NSINF was not able to communicate with the LU. If this occurs, you may want to check the system generation configuration, the NSINIT configuration, and the link interface card's select code.

Status—the status of the link LU. Possible states are `UP` and `DOWN`.

Type—the Link Interface type of the link LU. There are four possible types: `RTR` (Router/1000), `802` (IEEE 802.3), `ETHER` (Ethernet), `LAN` (both IEEE 802.3 and Ethernet).

Station address—the LAN station address associated with the link LU.

Multicast addresses—the Probe multicast addresses associated with the link LU. The first address is the Target Address; the second address is the Proxy Address, as defined in the *NS-ARPA/1000 Generation and Initialization Manual*.

Gateway Table Field Definitions

The Gateway Table display contains IP address and routing information that is derived from the NSINIT Directly Connected Network (DCN), and Gateway Table (GT) entries.

Destination net—the IP address of the network, with the node portion of the address set to zero. There should be an entry for each network in the internet that the local node has configuration information for, including the network or networks to which the local node belongs.

Gateway—the IP address of the configured gateway (GT) to use to reach Destination net. This should be a gateway on the local node's directly connected network (DCN). If the local node belongs to Destination net, this field will be local net.

Down PID—the protocol ID (PID) of the LI used to communicate with the Gateway. Possible values: RTR (Router/1000), IEEE-802 (IEEE 802.3), ETHER (Ethernet), and LAN (both IEEE 802.3 and Ethernet).

Segment Size—the network segment size, in bytes, for the DCN link.

Hops—the configured maximum number of gateways that IP will route through before reaching Destination net. (IP uses this in the Time-to-Live field of the IP header.) If the destination node is a member of the local net as shown in the Gateway column, then the Hops field is set to zero (0).

Subnetwork Mask—the subnetwork or subnet mask specifies the portion of the node address used to identify the subnet number. The remaining part of the node address is used to identify the node on that particular subnetwork. In the example the local network has the subnet mask of a Class C IP address, 255.255.255.000, which means there is no subnetting. Therefore, the first three octets specify the network number and the last octet specifies the node number.

C—Configured Resources

The C command displays the configured resources for NS Common Services and ARPA Services. Use this command to check if there are enough resources available.

Example

```
                        Resource Usage

                  Max Configured          Currently Active

NS Programs             14                       2
Sockets                 38                      10
Memory Buffers         110                      15
Name Records            19                       0
IP Path Records         40                       4
ARM CB Records          10                       0
Probe PCB Records       10                       0
ARP CB Records           7                       0

TELNET User Programs    24                       1
TELNET Server Programs   4                       1

   INPRO class   24
   OUTPRO class  23
   IFPM class    28
   IFP class     27

TCP Segment Size  : 4096 bytes
NFT Buffer Size   : 2048 bytes
NFT Checksum      : OFF
```

Field Definitions

NS Programs-Max Configured—the maximum number of NS-ARPA programs, as configured via NSINIT. This includes four NS-ARPA subsystem programs if you configured NFT; it includes three NS-ARPA subsystem programs if you did not configure NFT.

NS Programs-Currently Active—the number of concurrently active NS-ARPA programs, including NS-ARPA subsystem programs (such as INPRO, OUTPRO, EVMON, and NFTMN), at this time.

Sockets-Max Configured—the maximum number of NetIPC sockets, including root, call, and VC sockets, and BSD IPC sockets as configured via NSINIT. This includes 15 for NS-ARPA subsystem programs if you configured NFT; it includes six for NS-ARPA subsystem programs if you did not configure NFT.

Sockets-Currently Active—the number of concurrently active NetIPC Sockets, including NS-ARPA subsystem sockets, at this time.

Memory Buffers-Max Configured—the total number of dynamic memory buffers (Mbufs) available in DSAM. Each Mbuf is one page (2048 bytes). This number is the number of pages in DSAM left after the fixed tables have been configured.

Memory Buffers-Currently Active—the number of dynamic memory buffers currently in use.

Name Records-Max Configured—the maximum number of NetIPC name records, as configured via NSINIT. NetIPC creates a name record when a user names a socket with an `IPCName` call or gives a socket away with an `IPCGive` call.

Name Records-Currently Active—the number of concurrently active NetIPC name records at this time.

IP Path Records-Max Configured—the maximum number of concurrently active IP path records, as configured via NSINIT. There must be one IP path record for each unique triplet of source node, destination node, and upper-level protocol (TCP, PXP, or IFP). IP path records are not used for DS/1000-IV Compatible Service transactions over Router/1000, Bisync or X.25 links.

IP Path Records-Currently Active—the number of currently active IP Path Records.

ARM CB Records-Max Configured—the maximum number of concurrently active ARM CB records (for HP internal use only).

ARM CB Records-Currently Active—the number of concurrently active ARM CB records (for HP internal use only).

Probe PCB Records-Max Configured—the maximum number of concurrently active Probe Protocol Blocks (PCBs), as configured via NSINIT. Each active Probe query requires a Probe PCB. This field only shows up for nodes with 802 or LAN LIs, not for Ethernet-only LIs.

Probe PCB Records-Currently Active—the number of concurrently active Probe PCB records.

ARP CB Records-Max Configured—the maximum number of concurrently active ARP CB records (for HP internal use only). This field only shows up for nodes with LAN or Ethernet-only LIs, not for 802-only LIs.

ARP CB Records-Currently Active—the number of concurrently active ARP CB records (for HP internal use only). This field only shows up for nodes with LAN or Ethernet-only LIs, not for 802-only LIs.

TELNET User Programs-Max Configured—the maximum number of TELNET user programs configured for this node. The TELNET user program runs on the client (local) node and communicates with the remote server. The maximum number of TELNET user programs is 24 even if more terminal LUs are generated into the system. If there are no terminal LUs (not pseudo terminals) generated into the system, then the maximum number of TELNET user programs is 4.

TELNET User Programs-Currently Active—the number of TELNET user programs (virtual terminal sessions) currently active.

TELNET Server Programs-Max Configured—the maximum number of TELNET server programs configured for this node. The TELNET server program runs on the remote node and handles the virtual terminal session. The maximum number of TELNET server programs is 24 even if more pseudo terminal LUs are generated into the system. If there are no pseudo terminal LUs generated into the system, then the maximum number of TELNET server programs is 4.

TELNET Server Programs-Currently Active—the number of TELNET server programs currently active.

Monitor class *classNumber*—a list of the monitors required for NS Common Services, ARPA Services, and for DS/1000-IV Compatible Services over non-Router/1000 links (if configured at your node), and the class numbers. For IFP, this is the class number owned by INPRO for moving inbound messages from GRPM to INPRO.

Refer to the T command for the remaining DS/1000-IV monitors.

All nodes should have INPRO and OUTPRO. If the local node supports DS/1000-IV Compatible Services over non-Router/1000 links, IFPM and IFP are also required.

TCP Segment Size—the TCP initial segment size (bytes), as configured via NSINIT.

NFT Buffer Size—the default NFT buffer size (bytes), as configured via NSINIT.

NFT Checksum—ON indicates that the Layer 4 protocol checksumming is enabled for NFT; OFF indicates that it is not enabled.

I—List all NS-ARPA/1000 LUs

The `I` command lists the transmit LUs of the NS-ARPA/1000 link interface cards generated in the system. Use this command to verify the card configuration.

Example

```
NS/1000 LU Summary:

lu     device type      select code      availability

32          66b              25b             up
34          66b              26b             up
36          66b              30b             up
40          66b              32b             up
42          66b              33b             up
90          66b              31b             up
132         67b              35b             down
134         67b              34b             up
```

Field Definitions

lu—the link interface card's transmit LU.

device type—the link interface card's device type. `66` indicates a PSI link; `67` indicates LAN link.

select code—the link interface card's select code.

availability—the state of the LU, as recorded in word 6 of the DVT. Possible values are:

- `up`—the card is available
- `down`—the card is unavailable
- `busy`—the card is processing a request
- `down` and `busy`—the card is down, but it has a request pending.

L—Information on an NS-ARPA LU

The `L` command reads NS-ARPA link interface card statistics and IFT extent word information. Use this command to verify the card configuration. If you enter this command, NSINF will prompt you for a link interface card LU:

```
NSINF: Enter LU (cr for all LUs) >
```

Enter the transmit LU of a link interface card or press RETURN for the statistics of all the NS-ARPA/1000 interface cards on the local node.

The output for HDLC, Bisync, and LAN links is shown in the following three examples. Refer to the HDLC, Bisync, or LAN card documentation while reviewing these examples.

HDLC Example

```
*********************************************************************

lu     device type     select code     availability

32         66b             25b           up

HDLC Board, Firmware Revision:2519, Speed:230 K BPS, 1024 byte frame size
FCL disabled, Diagnostic Hood not sensed

           xxx  Good I-Frames Recvd         xxx   RR Frames Received
             0  RNR Frames Received           0   reject Frames Rcvd
             0  RCV Proc Overruns             0   CRC Errors
             0  Abort Seq. Received           0   Receiver Overruns
             0  Rx Buffer Overflows           0   Frames W/Bad Addr
             0  Cmdr Frames Rcvd              7   Unack Fr Window Size
            10  N2 Retry Count               15   T1 T.O. In 0.01 Sec

Flag bits from IFT extent (word 11)

  0 Read Aborted    0 Write Aborted   0 Rd Rq Pending    0 Wt Rq Pending
  0 Bkpl Locked RP  0 Bkpl Locked WP  0 Short TO active  0 Med. TO Active
  0 Long TO Active  1 Connected       1 Start of Msg.    0 Non-DS Mode
  1 Connect Req.    0 Severe Error    1 P-F Reconnect    0 RFP Wait
```

NSINF reads the following information from the PSI card which has the HDLC firmware and resets the values after it reads them. For more information on HDLC protocol, refer to the HDLC firmware installation manual.

GOOD I-FRAMES RCVD—Number of Information frames received by the HDLC card. Information frames carry data.

RR FRAMES RECEIVED—the remote card sends Receiver Ready (RR) frames to indicate that it is ready to receive an information frame or to acknowledge information frames. The remote card can also send an RR frame with the poll bit set to one to solicit a response

RNR FRAMES RECEIVED—the remote card sends Receiver Not Ready (RNR) frames when it is busy or unable to receive frames (for example, if the board runs out of buffers).

REJECT FRAMES RECEIVED—the remote card sends Reject (REJ) frames to request retransmission of information frames starting from the frame number in the N(R) (Receive Sequence Number) field.

RCV PROC OVERRUNS—Number of times the card ran out of buffer space because the host system could not read information off the card. Check that QUEUE is running, and the amount of SAM in your system.

CRC ERRORS—the Cyclic Redundancy Check (CRC) checks for transmission errors for each frame. A high CRC error rate may indicate a noisy link.

ABORT SEQ. RECEIVED—the remote card sends an abort sequence (seven or more consecutive ones) to prematurely terminate the transmission of a frame.

RECEIVER OVERRUNS—Number of times the card could not read data from the line into a receive buffer.

RX BUFFER OVERFLOWS—Number of times the card received a frame larger than the expected frame size. Check what size frames the remote card is sending.

FRAMES W/BAD ADDR—If the first byte in the address field is not one or three, the card ignores the frame and increments this counter.

CMDR FRAMES RCVD—the remote card sends Command Reject (CMDR) frames if it receives a frame without a CRC error, but it is unable to process, for one of the following reasons: the frame was invalid; the information frame exceeds the available buffer size; or the frame has an invalid N(R) field.

UNACK FR WINDOW SIZE—the unacknowledged frame window size is the maximum number of messages that can be outstanding before the card requires an acknowledgement. This number is set to seven.

N2 RETRY COUNT—the number of times the card will try to retransmit after a T1 timeout. Refer to the section “Principles of Operation” in this manual for more information.

T1 T.O. in .01 SEC—the maximum period of time that the sending card will wait for an acknowledgement before retransmitting a frame. This value is determined by the line speed and the selected frame size. Refer to the section “Principles of Operation” in this manual for more information.

Bisync Example

```
*********************************************************************
lu     device type     select code     availability

43         66b             26b           up

Card Parameters and Statistics for LU 43

BSC Board, Firmware Revision:2328, Speed:57.6K BPS, Internal Clock
FCL disabled, Diagnostic Hood not sensed

         11272 GOOD BLOCKS SENT           11270 GOOD BLOCKS RCVD
             0  BAD BLOCKS RECEIVED           0  NAKS RECEIVED
             0  WACKS SENT                    0  WACKS RECEIVED
             0  TTDS SENT                     0  TTDS RECEIVED
             0  RESPONSE ERRORS               0  3 SECOND TIMEOUTS
             0  LINE ERRORS                   0  AUTOMATIC RECONNECTS
          2048 BLOCKSIZE (BYTES)              8 RETRY LIMIT
           255 CONNECT TIMER                2490 TRACE SIZE (BYTES)

CARD CONNECTED AS SECONDARY

Flag bits from IFT extent (word 11)

  0 Read Aborted    0 Write Aborted   0 Rd Rq Pending    0 Wt Rq Pending
  0 Bkpl Locked RP  0 Bkpl Locked WP  0 Short TO active  0 Med. TO Active
  0 Long TO Active  1 Connected       1 Start of Msg.    0 Non-DS Mode
  1 Connect Req.    0 Severe Error    0 P-F Reconnect    0 RFP Wait
```

All counters on the Bisync card are maintained as 16-bit unsigned integers. If they are not reset, they will roll over at 65,535. Statistics are reset each time the connection is established unless the connection is automatically reenabled. The following is an explanation of each of the parameters or statistics reported:

GOOD BLOCKS SENT—Number of data blocks that the Bisync card has sent which were acknowledged as good by the remote HP 3000. A data block is the basic transmission unit at the Bisync level. One user message may be sent as several Bisync data blocks. Each data block sent must be acknowledged by the receiver.

BAD BLOCKS RECEIVED—Number of bad blocks received from the remote HP 3000. (See GOOD BLOCKS SENT for an explanation of data blocks.)

WACKS SENT—Number of Wait Positive Acknowledgments sent. A WACK is sent when a good data block has been received but the receiver is not yet ready to receive another. Upon receipt of a WACK, the sender will normally send an ENQ until the receiver indicates it is ready to receive by responding with an ACK 0. A WACK may be sent if the receiver does not have an available buffer.

TTDS SENT—Number of Temporary Text Delays sent to the remote. A TTD is transmitted by the sending station when it wants to retain control of the line but is not yet ready to send. This may occur if the send buffers are not yet full.

RESPONSE ERRORS—This counter is incremented each time the local station receives an unexpected response (i.e., a protocol error). This may be the result of a transmission error.

LINE ERRORS—Number of times an error has been detected with the line (such as loss of data set ready). This may be an indication of faulty lines or connections.

BLOCK SIZE (BYTES)—the maximum number of bytes that the card can send across the line as a single block.

CONNECT TIMER—Number of seconds that the receiver will wait for an initial connection.

RETRY LIMIT—Number of times a transmission will be retried if errors are detected. Default is 8.

TRACE SIZE (BYTES)—Size in bytes of the buffer area on the Bisync card which is reserved for state change logging.

AUTOMATIC RECONNECTS—the automatic enable flag indicates to the firmware that the card should be re-enabled in secondary mode after a disconnect has occurred. This flag is always set.

GOOD BLOCKS RCVD—the number of good data blocks received from the remote HP 3000. (See GOOD BLOCKS SENT for an explanation of data blocks.)

NAKS RECEIVED—Number of Negative Acknowledgements received. A NAK may be sent by the receiving station if it receives a bad data frame. A high number of NAKs received or sent may indicate poor line quality or bad connections. Upon receipt of a NAK, the sending station should retransmit the previous data block.

WACKS RECEIVED—Number of Wait Positive Acknowledgements sent. (See WACKS SENT for an explanation of WACKs.)

TTDS RECEIVED—Number of Temporary Text Delays received from the remote station. (See TTDS SENT for an explanation of TTDs.)

3 SECOND TIMEOUTS—the amount of time the receiver will wait for any response before requesting a retry.

CARD CONNECTED AS PRIMARY—indicates that the card was connected in primary mode.

CARD CONNECTED AS SECONDARY—indicates that the card was connected in secondary mode.

LAN Example

```
*********************************************************************
lu     device type     select code     availability

134        67b             34b           up

LAN Card, Firmware Revision:2544

         34578  Good Bytes Sent             18616  Good Bytes Rcvd
           312  Good Packets Sent             245  Good Packets Rcvd
             0  Errors on Xmit                  0  Errors on Recv
             0  Babble Errors                   0  Heart Beat Errors
             0  Missed Packets                  0  Memory Errors
             0  Framing Errors                  0  Discarded Packets
             0  CRC Errors                      0  Length Errors
             0  Multiple Retries                0  No Retries
             0  Transmit Deferral               0  Underflows
             0  Late Collisions                 0  Loss of Carrier
             0  Retry Exceeded                  0  TDR Information

  0 Last Write Failed   0 Link Write Pending    0 Interrupt Pending
  0 Wait For TX Buffer  0 RCV Packet Available  0 No Receive Buffers
  0 MAU Power Failed    0 Lost NOVRAM data      0 NOVRAM Bank 2 Enable
  1 Multicast Mode      0 Broadcast Mode        0 Promiscuous Mode
```

For the meanings of the LAN card statistics, refer to *HP 12076A LAN/1000 Link Node Manager's Manual*.

M—Message Accounting

The M command prints Message Accounting (MA) information for each node entered in the NRV with MA enabled.

Example

```
Message Accounting Information

Node State  # Unack  # Linedowns  Timeout  # Cancellations
   1   None       0           0        3               0
   3   Up         0           0        3               0
   4   Up         0           0        3               0
   5   Up         0           0        3               0
   6   Pend       0           0        3               0
   7   Up         0           0        3               0
  12   Up         0           0        3               0
  13   Down       0           0        3               0
  14   Up         0           0        3               0
 561   Up         0           0        3               0
 571   Up         0           0        3               0
```

Field Definitions

Node—the Router/1000 nodes entered in the local node's NRV with MA enabled.

State—the state of communications between the local node and the node identified in the Node column. Possible values:

None—indicates that the remote node doesn't have MA initialized.

Up—indicates that communication is up between the nodes (normal state).

Pend—indicates that the Message Accounting software is attempting to establish communications with the remote node.

Down—indicates that communication between the two MA nodes has failed (no path to the node exists, or no path has been used since NS-ARPA initialization).

 # Unack—the number of unacknowledged messages that have been sent from the local node to other MA nodes from which no acknowledgement has yet been received. This field can contain entries from 0 to 15. When 15 unacknowledged messages are pending, any attempt to send additional messages will result in a DS08 (node busy or resource unavailable) error message returned to the caller.

 # Linedowns—indicates how many times since NS-ARPA initialization that all paths to the destination node have gone down. This value can indicate the link quality, that is, if the linedown count is small, the link quality must be reasonably good. Note that MA will automatically reinitialize a link that has been down but comes back to service.

Timeout—indicates how long, in seconds, MA will wait for an acknowledgement for a particular message.

 # Cancellations—the number of currently unacknowledged CANCEL messages. If a message is not acknowledged within the the Timeout value, the MA software sends a CANCEL message to the remote MA software before retransmitting it. If MA does not receive an acknowledgment to the CANCEL message within the Timeout value, it will retransmit the CANCEL message. MA will continue retransmitting the CANCEL message until it receives an acknowledgment or it exceeds the MA retry limit. If it exceeds the retry limit, MA will generate a DS05(3) error message and set the communications state to the remote node to Down.

When Message Accounting is not present in the system, NSINF prints the following message:

```
MESSAGE ACCOUNTING INFORMATION

          NO ENTRIES
```

N—Nodal Routing Vector

The N command prints the Nodal Routing Vector (NRV).

Example

```
NRV Specifications:

 Local node #:    16 , No. of Nodes =   10

      Node        IP ADDRESS      LU          T/O(Sec)     Type     Level
         1     192.006.001.001    90               0        66         1
         2*    192.006.001.001    90               0        66         1
         4     192.006.001.004    90               0        66         1
         5     192.006.001.005     0,(NR)          0                   1
        12     192.006.001.005     0,(NR)          0                   1
        14     192.006.001.014     0,(NR)          0                   1
        16*    192.006.250.016     0               0                   1
        19     192.006.250.019     0,(NR)          0                   1
```

Field Definitions

Local node #—the Router/1000 address of the local node.

No. of Nodes—the number of nodes in the NRV, including the local node.

Node—Router/1000 address of the node. An asterisk after the address indicates a directly-connected node.

IP Address—IP address of the node. If the remote node and the local node belong to the same Router/1000 network, this field will be `000.000.000.000`. If the remote node is a DS/1000-IV (91750) node, this field will be the IP address of the 91750 node's Guardian node, unless it belongs to the same Router/1000 network as the local node.

LU—the transmit LU of the Router/1000 link used to reach the node. If the link is not a Router/1000 link, the LU is 0 and marked `(NR)`.

T/O(Sec)—the transaction timeout in seconds. If 0, Router uses the Master timeout value.

Type—the link device type.

Level—the software upgrade level of the remote node. A `0` level indicates DS/1000 (91740). A `1` level indicates DS/1000-IV (91750) or NS-ARPA/1000 (91790).

P—Individual Program Information

The `P` command prints socket record and protocol path record information for the sockets owned by a program. Use this command to help troubleshoot a user program. If you enter this command, NSINF will prompt you for a program name:

```
ProgName:
```

Enter the name of a currently scheduled program.

Example

```
 Sockets owned by program myipc

GSD: 17   UrecID: 14   Kind:  ROOT     STATE: ROOT_CLEAR

GSD: 18   UrecID: 14   Kind:  CALL     STATE: CALL_BOUND
  down ref: 6
TCP     source port:  16386

GSD: 19   UrecID: 14   Kind:  VC        STATE: VC_OPEN_CONFIRMING
  down ref: 11
TCP     source port:  16393                 dest port: 16385
         rcv ulp ct:      3                snd ulp ct:     2
          snd dn ct:      2                 snd dn ct:     2
  down ref: 8
IP      source addr:  192.006.001.001       dest addr: 192.006.001.002
         rcv ulp ct:      2                snd ulp ct:     2
          snd dn ct:      2                 snd dn ct:     2
  down ref: 257
802     source addr:  08-00-09-00-02-4F      dest addr: 08-00-09-00-02-39
         rcv ulp ct:     71                snd ulp ct:    69
 lu: 132
```

For each NetIPC socket owned by the program, NSINF prints a socket record, with all the protocol path records associated with the socket printed below the socket. NSINF then prints the next socket record, with all its associated protocol path records, until it has printed the socket records and associated protocol path records for all the NetIPC sockets owned by the program.

For more information, refer to the subsection “NS-ARPA/1000 Path Flow” in the section “Principles of Operation.”

Field Definitions

GSD—the socket’s Global Socket Descriptor, which is the numeric identifier for the socket that is unique on this node.

UrecID—the socket's User Record ID, or the NetIPC identifier for the program that owns the socket.

Kind—the type of socket. Possible values are ROOT, CALL, or VC (Virtual Circuit). NetIPC allocates a Root socket for every program that makes a NetIPC call. For more information, refer to the subsection "NS-ARPA/1000 Path Flow" in the section "Principles of Operation."

State—the state of the socket.

down ref—the reference to the lower-layer (TCP or PXP) path record below this socket.

TCP source port—the source TCP port. If a TCP path record has a source TCP port but not a destination TCP port, it means that either: the TCP path record is associated with a NetIPC Call socket; or that the TCP path record is associated with a VC Socket, but the TCP connection has not been established yet.

TCP dest port—the destination TCP port (the TCP port on the remote node).

TCP rcv ulp ct—the number of event messages received from the upper-layer protocol, NetIPC.

TCP snd ulp ct—the number of event messages sent to the upper-layer protocol, NetIPC.

TCP snd dn ct—the number of event messages sent to the lower-layer protocol, IP.

TCP rcv dn ct—the number of event messages received from the lower-layer protocol, IP.

TCP down ref—the reference to the IP path record below this path record.

PXP source port—the source PXP port. If a PXP path record only has a source PXP port, it means that the PXP path record is associated with a service port.

PXP dest port—the destination PXP port (the PXP port on the remote node).

PXP rcv ulp ct—the number of event messages received from the upper-layer protocol, NetIPC.

PXP snd ulp ct—the number of event messages sent to the upper-layer protocol, NetIPC.

PXP snd dn ct—the number of event messages sent to the lower-layer protocol, IP.

PXP rcv dn ct—the number of event messages received from the lower-layer protocol, IP.

PXP down ref—the reference to the IP path record below this path record.

IP source addr—the IP source address. Should be one of the local node's IP addresses.

IP dest addr—the destination node's IP address.

IP rcv ulp ct—the number of event messages received from the upper-layer protocol, such as TCP or PXP.

IP snd ulp ct—the number of event messages sent to the upper-layer protocol, such as TCP or PXP.

IP snd dn ct—the number of event messages sent to the lower-layer protocol, such as LAN or RTR.

IP rcv dn ct—the number of event messages received from the lower-layer protocol, such as LAN or RTR.

IP down ref—the reference to the Link Interface path record below this record.

LAN source addr—the IEEE 802.3 or Ethernet source address of the local node.

LAN dest addr—the IEEE 802.3 or Ethernet address of the Appropriate Next Hop.

LAN rcv ulp ct—the number of event messages received from the upper-layer protocol, IP, or Probe.

LAN snd ulp ct—the number of event messages sent to the upper-layer protocol, IP, or Probe.

LAN lu—the LU used to reach `LAN dest addr`.

RTR dest addr—the Router/1000 address of the Appropriate Next Hop.

RTR lu—the LU used to reach `RTR dest addr`.

R—Rerouting Information

The `R` command prints information about the local node's Router/1000 links.

Example

```
Rerouting specifications:

                              Up/Down
                LU     Cost   Counter    Status
                36        1         0       up
                88        1         0       up
```

Field Definitions

`LU`—the link interface card's transmit LU.

`Cost`—the link's cost value. (The Router/1000 software uses this to determine the best route between nodes, where the link with the lowest cost is chosen.)

`Up/Down Counter`—the current count of link failures and re-enables.

`Status`—the link status. Possible values: `Up` and `Down`.

If there are no Router/1000 links generated in the local system, NSINF will print the following message:

```
Rerouting specifications:

  No rerouting links at this node
```

S—Sockets

The `S` command prints information about the open VC and CALL sockets.

Field Definitions

`Type`—the type of socket. Possible values are CALL or VC (Virtual Circuit). Root sockets are not displayed.

`Local-port`—the local TCP port associated with the socket.

`Foreign-Address`—the IP address of the remote socket.

`Foreign-Port`—the TCP port on the remote node. This field will be blank for a call socket, or if the VC socket has not established a connection.

`State`—the TCP state of the socket. Possible values are `CLOSED`, `LISTEN`, `SYN_SENT`, `SYN_RECEIVED`, `ESTABLISHED`, `CLOSE_WAIT`, `FIN_WAIT_1`, `CLOSING`, `LAST_ACK`, `FIN_WAIT_2`, `TIME_WAIT`.

`Owner/Session`—the program which owns the socket and its session number.

Example

```
NSInf > s
Type  Local-port  Foreign-Address  Foreign-Port   State        Owner/Session

CALL       21                                   LISTEN        INETD
CALL       23                                   LISTEN        INETD
CALL       25                                   LISTEN        INETD
CALL    25500                                   LISTEN        INETD
CALL      514                                   LISTEN        INETD
CALL      515                                   LISTEN        INETD
CALL     1542                                   LISTEN        RPMMN
CALL     1536                                   LISTEN        NFTMN
CALL       53                                   LISTEN        NAMED
VC         23      192.006.070.008   1226       ESTABLISHED   TNS.B
VC         21      192.006.070.007   3263       ESTABLISHED   FTPSV/120
VC         21      192.006.070.005   3795       ESTABLISHED   FTP.A/123
CALL       20                                   CLOSED        FTPSV/120
VC         23      192.006.070.007   1060       ESTABLISHED   TNS.A
CALL       20                                   CLOSED        FTP.A/123
VC         23      192.006.070.005   3237       ESTABLISHED   TNSRV
VC         23      192.006.070.004   3031       ESTABLISHED   TNS.C
```

T—List Tables

The `T` command prints the number entries in the Master Transaction Control Block (TCB) list, Slave TCB list and Process Number Lists (PNL). If a list has entries, NSINF prints the address of the first TCB in the list. The first word in the entry is the address of the next entry or else zero to indicate the end of the list.

PNL entries, Master TCBs, and Slave TCBs are allocated out of the Transaction Control Block Pool.

Example

```
DS/1000 Lists:

      1 Entries in Master Request List
       Prog  Class  T/O CTR
       NSI.A    27      0
        Active Slave Monitors             1st TCB
        stream  class  monitor  entries  address
           1      40    DLIST      0          0
           3      39    EXECW      0          0
           4      38    PTOPM      0          0
           5      37    EXECM      0          0
           6      36    RFAM       0          0
           7      35    OPERM      0          0
          12      34    TRFAS      0          0
      0 Entries in Slave Lists

      2 Entries in Process Number List, Starting at 63000
       Prog LogLU
      *RMO20  90
     B TST21  21

     20 Entries in Null list, starting at 64670
```

Master List Entries

The Master List contains an entry for each master request currently outstanding. The words in the TCB have these meanings:

```
WORD 1 - Address of next entry in list (0 indicates end)
WORD 2 - Bit 15:      Temporary bit used by UPLIN
         Bit 14:      Set for requests to HP 3000
         Bit 13:      MA acknowledgement
         Bits 8-12:   Reserved for future use
         Bits 0- 7:   Timeout counter. Timeout occurs when the
                      counter reaches octal 377.

WORD 3 - Local Sequence Number
WORD 4 - Bit 15:      When set, indicates long master timeout
                      (about twenty minutes)
   Bits 0-14:  Master Class Number

WORD 5 - Bit 15:      When set, indicates an entry UPLIN will
                      release during its next execution.
         Bits 0-14:   Master program's ID Segment address
WORD 6 - MA Sequence Number (HP 1000) or negative LU (HP 3000)
```

Slave List Entries

The Slave Lists contain an entry for each outstanding slave request. There is a Slave List for each enabled slave monitor. Entries in these lists represent each remote request waiting for service at the time the lists are printed. The words in the TCB have these meanings:

```
WORD 1 - Address of next entry in list (0 indicates end)
WORD 2 - Same as master list entry
WORD 3 - Local Sequence Number (Assigned when request arrives)
WORD 4 - Origin Sequence Number (Assigned at the origin node)
WORD 5 - Origin Nodal Address (HP 1000) or -LU (HP 3000)
WORD 6 - Reserved
```

Process Number List Entries

An entry is made in this list each time an HP 1000 user issues a successful HELLO to an HP 3000 or logs onto a remote HP 1000. The entry is deleted after a BYE (HP 3000) or a DLGOF (HP 1000) or when the Master Program terminates. The words in the PNL have these meanings:

```
WORD 1 - Address of next entry in list (0 indicates end)
WORD 2 - Bit 14: when set, indicates an HP 3000 session
WORD 3 - Remote node number (HP 1000) or -LU (HP 3000)
WORD 4 - Logging device LU number
WORD 5 - Bit 15:      When set, indicates an entry UPLIN will
                      release at its next execution.
         Bits 0-14:   Program's ID Segment address
WORD 6 - Remote Session ID or HP 3000 Process Number
```

Any entries which are not in one of these three active lists are not currently being used and are in the Null List.

The asterisks in the Process List and Master Request List indicate the programs that are using an HP 3000 link. A `B` in the Process List and Master Request List indicates that the associated entry will be deleted by UPLIN. Slave monitor stream lists are not printed if the associated slave monitor has not been enabled by NSINIT.

U—NS-ARPA Utility Status and Statistics

The U command prints information about the NS-ARPA message tracing utility (NSTRC) and the NS-ARPA event logging monitor (EVMON). Use the information to check EVMON statistics and the event classes being logged.

Example

```
                  NS Utility Status and Statistics

                       Event Logging Summary

Current Log Mask:      161b
Log file name:        /SYSTEM/NS_EVENT.LOG

                      * Resource Limit       0
                      * Disaster             0
                      * Error                7
                        Warning              0
                        Event Message        0

Message Tracing                             Trace level: Network

                       NSTRC class number 43
```

Field Definitions

Log file name—the name of the current log file.

Current Log Mask—the octal value of the current log mask. For the meaning of the bits in the log mask, refer to the section, "NS-ARPA Event Logging," in this manual.

Resource Limit—the number of Resource Limit log records posted since NS-ARPA/1000 was last initialized. An asterisk indicates that this log class is enabled.

Disaster—the number of Disaster log records posted since NS-ARPA/1000 was last initialized. An asterisk indicates that this log class is enabled.

Error—the number of Error log records posted since NS-ARPA/1000 was last initialized. An asterisk indicates that this log class is enabled.

Warning—the number of Warning log records posted since NS-ARPA/1000 was last initialized. An asterisk indicates that this log class is enabled.

Event Messages—the number of Event Message log records posted since NS-ARPA/1000 was last initialized. An asterisk indicates that this log class is enabled.

Trace level—the NSTRC trace level. Possible values are Network, Socket, and Both (both Network and Socket).

NSTRC class number—NSTRC's class number, used by NSTRC to read NS-ARPA messages from SAM.

V—DS/1000-IV Values

The V command prints information about NS-ARPA/1000 resource numbers, information about DS/1000-IV Compatible Services system resources and configuration values, and the LU table of links used for DS/1000-IV Compatible Services (RTE-MPE).

Example

```
DS/1000 Values:

 Resource numbers:      Owner     Locker
   Initialization  2   (Global)   (None)
   TCB Access      3   (Global)   (None)
   MA Table Access 5   (Global)   (None)
   QUEX            7   (Global)   (Global)
 Classes assigned to programs:
   26  EXECM
   27  EXECM
   31  M. A.
   30  QCLM
   41  GRPM

 Timeout Values (sec):
   Master Timeout          45
   Slave Timeout           30
   Remote Busy Retries      3
   Remote Quiet Wait        0
   Max Retry Delay       2.00
 Progl Message LU           0
 Last APLDR Load-Node    None
 Upgrade Level              1
 Upgrade SubLevel           1
    128 RFA Files May Be Open

HP 3000 LU Table
  LU  Buffer Size  CR Flag
  42      1024         1     (X.25)
```

If a portion of DS/1000-IV Compatible Services or Transport (HP 3000 links, HP 1000 links, or Message Accounting) is not enabled, NSINF will not print the information associated with that item.

Resource numbers—the resource numbers used by NS-ARPA subsystem programs. The Owner program should be (Global) because all NS-ARPA/1000 RNs are allocated globally. Under normal conditions, the resource numbers are not locked (Locker is (None)) for Initialization, TCB Access and MA Table Access. If QUEZ is present in the system, its resource number is globally locked under normal conditions.

Classes assigned to programs—are the class numbers assigned to the DS/1000-IV Compatible Services monitors. For the class numbers of the monitors required for NS Common Services, ARPA Services, and for DS/1000-IV Compatible Services over non-Router/1000 links, refer to the `C` command.

Timeout Values—the timeout values used for DS/1000-IV Compatible Services and Transport. These values are described in the section "DSMOD," under the `/T` command.

Progl Message LU—the LU used by PROGL to display download initiation and error messages.

Last APLDR Load-Node—the Router/1000 address of the last node from which APLDR loaded a program.

Upgrade Level—the local node's DS/1000-IV Compatible Services message format upgrade level number. If the local node sends a message to a node with a lower upgrade level it will convert the message format before transmitting outbound messages to that node, and will convert inbound messages from that node. An upgrade level of 0 indicates a DS/1000 (91740) node; an upgrade level of 1 indicates a DS/1000-IV (91750) or NS-ARPA/1000 (91790) node.

Upgrade SubLevel—the local node's DS/1000-IV Compatible Services message format sub-level number. Sub-level numbers indicate minor protocol changes that do not require message converters.

nnn `RFA files May Be Open`—the maximum concurrent number of files at the local node that may be open via RFA. This value is determined by RFAM's program size.

HP 3000 LU—the transmit LU of the PSI Bisync or X.25 link used to communicate with the HP 3000 for DS/1000-IV Compatible Services.

Buffer Size—the communication buffer size, in bytes. If the link is an X.25 HP 3000 LU, it is the library size generated in the system (4096). There is no communication buffer size for X.25.

CR Flag—Continuation record flag. `1` indicates that the HP 3000 accepts continuation records for $STDIN and $STDLIST; `0` indicates that it does not.

(X.25)—indicates that the LU is an X.25 link.

W—Information on TELNET Sessions

The `W` command prints information about TELNET virtual terminal sessions, both initiated at and remotely connected to the local node. Use this command to help troubleshoot TELNET.

Example

```
                                Resource Usage

   TELNET          Max Configured          Currently Active

User Programs            24                       1

Server Programs           4                       2

Bad commands received       0
Bad options received        0

Pseudo LU           Status           Server Name     Bad Cmd/Options
   30         allocated by INETD       TNSRV               0
   31                free                                  0
   95               in use             TNS.A               0
   96                free                                  0
```

Field Definitions

`TELNET User Programs-Max Configured`—the maximum number of TELNET user programs configured for this node. The TELNET user program runs on the client (local) node and communicates with the remote server. The maximum number of TELNET user programs is 24 even if more terminal LUs (not pseudo terminals) are generated into the system. If there are no terminal LUs generated into the system, then the maximum number of TELNET user programs is 4.

`TELNET User Programs-Currently Active`—the number of TELNET user programs (virtual terminal sessions) currently active.

`TELNET Server Programs-Max Configured`—the maximum number of TELNET server programs configured for this node. The TELNET server program runs on the remote node and handles the virtual terminal session. The maximum number of TELNET server programs is 24 even if more pseudo terminal LUs are generated into the system. If there are no pseudo terminal LUs generated into the system, then the maximum number of TELNET server programs is 4.

`TELNET Server Programs-Currently Active`—the number of TELNET server programs currently active.

`Bad commands received`—the number of bad TELNET protocol commands received so far on the local node. If the value is not zero, TELNET may be having problems.

Bad options received—the number of bad TELNET protocol options received so far on the local node. If the value is not zero, TELNET may be having problems.

Pseudo LU—the pseudo terminal LUs defined at system generation. Check the values here with the ones in your system generation. Note that the number of pseudo terminal LUs is equal to the number of TELNET server programs.

Status—there are four different kinds of status:

- *free*—the pseudo terminal LU is not being used by any TELNET server.
- *in use*—the pseudo terminal LU is being used by a TELNET server.
- *allocated by INETD*—the pseudo terminal LU has already been allocated by INETD for a TELNET server, but that server has not started to execute yet.
- *dirty, waiting to be cleaned up*—the TELNET server has terminated, but the pseudo terminal LU is waiting to be deallocated and resources returned to the system.

Server Name—the name of the TELNET server which has allocated the corresponding pseudo terminal LU.

Bad Cmd/Options—the total number of bad TELNET protocol commands or options received on that pseudo terminal LU. If the value is not zero, TELNET may be having problems.

3

Level 3 Loop Back Diagnostic, PING

PING

PING is a troubleshooting diagnostic that performs a level 3 (IP level) loop back. PING sends an Internet Control Message Protocol (ICMP) echo packets to a remote node. PING sends one packet per second to the remote host. Each packet that is echoed back is reported on the screen, including its round-trip time. PING continuously sends packets to the remote host until the break flag is set using the RTE BR command. When PING terminates, it reports statistics concerning the number of packets sent and lost as well as a summary of round-trip time.

Syntax

```
[RU,]PING remote_host [packet_size] [packet_number]
```

Parameters

remote_host

The *remote_host* parameter can be the node name or the IP address of the remote host. All symbolic names specified for a host are looked up in the Nodal Registry only. If it is an IP address, it must be in the following format, where *n* is an integer between 0 and 255, inclusive:

```
n.n.n.n
```

Refer to the "IP Address" subsection in the "NS-ARPA/1000 Configuration" section of the *NS-ARPA/1000 Generation and Initialization Manual* for more information on IP addresses.

packet_size

The size of the packets transmitted, in bytes. The minimum value allowed is 8 bytes and the maximum allowed is 1477 bytes. Also, if the packet size is less than 16 bytes, there is not enough room for timing information. In this case, the round-trip time for each packet and the summary round-trip time will not be displayed.

Range: 8 to 1477 bytes.
Default: 64 bytes.

packet_number The number of packets PING will transmit before terminating.

Range: 1 to 32767 packets.
Default: If no *packet_number* parameter is specified, PING will continue to send packets until the BREAK flag is set by using the RTE BR (break) command.

Discussion

PING verifies the physical connection to a remote node and reports the round-trip communications time between the local and remote nodes. PING uses the Internet Control Message Protocol (ICMP) echo facility to send ICMP packets to the Network Layer (OSI layer 3, also known as IP layer) of the remote node. The remote node must support receiving and responding to ICMP echo packets. A packet is sent to the remote node every second. As each echo response is received from the remote host, the round-trip time is reported.

Although the local and remote nodes must both be capable of ICMP, you do not need to understand this protocol to execute PING.

PING should be initiated

- to do a preliminary connectivity check when setting up new nodes
- when difficulties arise in connecting to a particular node or when response from a node seems unusually slow
- to check the reliability of a route through a gateway

If PING is initiated and the remote node does not respond to the outgoing packets, no round-trip information is reported. An error message may or may not be displayed depending on the nature of the problem. When halted, PING statistics typically indicate a 100% packet loss.

Note Only one PING program can run at any one time. If more PING programs are scheduled, they will be queued and run one at a time.

To halt PING, hit the space bar to obtain the CM> prompt. Then enter the RTE BR command, and PING stops executing.

Example

Listed below is a normal PING output when ZING is the remote host:

```
CI> PING ZING 100
PING  ZING:  100 byte packets
 100 bytes from ZING  icmp_seq =    1 time =   70 ms.
 100 bytes from ZING  icmp_seq =    2 time =   30 ms.
 100 bytes from ZING  icmp_seq =    3 time =   30 ms.
 100 bytes from ZING  icmp_seq =    4 time =   30 ms.
 100 bytes from ZING  icmp_seq =    5 time =   30 ms.
CM> BR <return>
 100 bytes from ZING  icmp_seq =    6 time =   30 ms.
******************>ZING  PING  Statistics<******************
    6 packets transmitted,     6 packets received,  0.00% packet loss.
Round-trip (ms) min/avg/max =    30/   37/   70
```

4

NS-ARPA Event Logging

Overview

This section describes how to use the NS-ARPA event logging utilities.

Event logging is the recording of network events at your local node. Network events can be normal events, network errors, warnings, disasters, or other events that may affect the integrity of your network.

In NS-ARPA you enable event logging by scheduling EVMON. EVMON can also be scheduled at network initialization time by NSINIT. Refer to the *NS-ARPA/1000 Generation and Initialization Manual*. NS-ARPA protocol handlers and services send records for NS-ARPA events, called *log records*, to EVMON. EVMON then writes the log records to a file or device. A log record consists of two parts: a log record header and the log message. Some NS-ARPA software modules have specific codes in the log records. The contents of log records are explained in "EVMON Output," later in this section and in the *NS-ARPA/1000 Error Message and Recovery Manual*.

The NS-ARPA logging utilities categorize events into seven *event classes*, which are described in the following pages. You can specify the event classes for which you want EVMON to write the log records.

There are three NS-ARPA event logging utilities described in this section. These logging utilities enable and terminate event logging and change the event classes:

- EVMON, the event monitor. EVMON receives log records from NS-ARPA protocol handlers and services, and writes the records to a log file or device.
- BREVL, which terminates EVMON, allowing you to purge the log file.
- LOGCHG, which allows you to change the event classes for EVMON to log while EVMON is running.

Using the Event Logging Utilities

In most cases, you would use the NS-ARPA event logging utilities in the following manner:

1. Start event logging with EVMON, or via NSINIT if you are initializing the node. Specify a file for the log device.
2. At regular intervals (once a day, for example), examine the log file for unusual events. EVMON opens log files with shared read access, so you can read or copy the log file without interrupting EVMON. To rename or purge the current log file, however, you must run BREVL to halt logging.
3. Resume event logging with EVMON.

In addition, you can use the NSINF `U` command to check EVMON statistics and the event classes that EVMON is logging.

EVMON

Enable event logging.

Runstring

```
[XQ,]EVMON[,logDevice][,logMask[B]]
```

Parameters

`logDevice`

The device (file or device LU) to which EVMON will log the events.

If *`logDevice`* is an existing file, EVMON will append to it. EVMON opens the log file with shared read access and writes logging records to a file until the LU that the file resides on is full, or until you halt EVMON (via BREVL), or until you shut down NS-ARPA/1000.

Default: The file `/SYSTEM/NS_EVENT.LOG`.

`logMask``[B]`

The octal representation of the bits set in the following mask. The *`logMask`* must be octal; as an option, you may append a `B` to it.

Log Mask:

The bits set select the event classes as described in Table 4-1.

Default: the current value of *`logMask`* which is specified at initialization in the NSINIT dialogue. NSINIT initially sets this value to 161 octal to log logging statistics, severe errors, disasters, and resource depletion.

Table 4-1. Log Mask Event Classes

Bit	Position	Logging Event Class
L	0	Logging Statistics (Class 0). *Must be set*. Logging start and stop times.
P	1	Protocol-specific information (Class 1). *HP recommends that you do not select this class unless your HP representative specifically asks that you do so*. This logging event class generates a multitude of log file entries.
M	2	Event messages (Class 2). *HP recommends that you do not select this class unless your HP representative specifically asks that you do so*. This logging event class generates a multitude of log file entries.
W	3	Warnings (Class 3), which indicate abnormal events, but do not necessarily indicate subsystem problems.
E	4	Severe errors (Class 4), which indicate that NS-ARPA/1000 is not performing as it should, but the subsystem was able to recover. *HP recommends that you select this logging class*. NSINIT selects this logging class automatically.
D	5	Disasters (Class 5), which indicate that the NS-ARPA/1000 software detected a severe and irrecoverable problem. *HP recommends that you select this logging class*. NSINIT selects this logging class automatically.
R	6	Resource limit (Class 6), which indicates that a user-configurable resource has been depleted. The network manager can use this information to adjust the node's configuration. *HP recommends that you select this logging class*. NSINIT selects this logging class automatically.
-	7	Reserved. *Must be zero*.

Discussion

If EVMON is unable to write a logging message to *logDevice*, EVMON will attempt to write an error message to the terminal from which EVMON was scheduled. The *logMask*, which defines the event classes, can be changed by LOGCHG described later in this subsection. EVMON is terminated with BREVL described in the next subsection.

EVMON error messages are listed in the *NS-ARPA/1000 Error Message and Recovery Manual*.

Example

```
CI>XQ,EVMON,MYLOG.LOG,161
```

Log logging statistics, severe errors, disasters, and resource depletion (the recommended log mask) to MYLOG.LOG in the current working directory.

BREVL

BREVL terminates EVMON and halts logging.

Runstring

```
[RU,]BREVL
```

Discussion

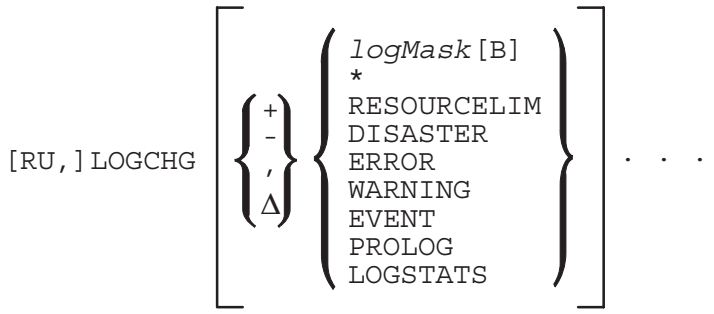
To rename or purge the current log file, you must terminate EVMON, which halts event logging. To do this, run BREVL. BREVL sets EVMON's break flag and BREVL then sends a termination record to EVMON. The termination record is necessary because EVMON is normally suspended, waiting for a log record to arrive and will not terminate until it receives another record. Before terminating, EVMON releases its NetIPC socket and writes a termination message to the log file or device and prints the following message at EVMON's scheduling terminal:

```
BREVL: Event Logging Terminated.
```

LOGCHG

LOGCHG allows you to change the log mask, *logMask* while EVMON is running.

Runstring



Parameters

`+`
`-`
`,`
Δ

LOGCHG evaluates the runstring parameters as an expression and sets the log mask (*logMask*) to the new value. The parameters are delimited by a plus, minus, comma, or a blank space.

LOGCHG uses two operators when evaluating the expression: + and -. The + represents logical addition (bit inclusion); the - represents logical subtraction (bit exclusion). When two terms have no operator between them, the + operator is used. For example, `LOGCHG DISASTER LOGSTAT` is the same as `LOGCHG DISASTER+LOGSTAT`.

When the expression begins with a + (or -), the value of the expression is added to (or subtracted from) the current log mask. For example, `LOGCHG +WARNING` is the same as `LOGCHG *+WARNING`; either one will set the `WARNING` bit of the log mask. The asterisk (*) in `*+WARNING` is the current value of *logMask*.

logmask`[B]`

The octal representation of the bits set in the following mask. The *logMask* must be octal; as an option, you may append a `B` to it.

The bits set select the event classes as described in Table 4-1.

The current value of *logMask*.

The log mask can also be specified using bit names. LOGCHG recognizes the following bit names which represent one bit (one event class) in the log mask:

RESOURCELIM	The bit name for the resource limit event class (Class 6); same as bit 6.
DISASTER	The bit name for the disasters event class (Class 5); same as bit 5.
ERROR	The bit name for the severe errors event class (Class 4); same as bit 4.
WARNING	The bit name for the warnings event class (Class 3); same as bit 3.
EVENT	The bit name for the event messages event class (Class 2); same as bit 2.
PROLOG	The bit name for the protocol-specific information event class (Class 1); same as bit 1.
LOGSTAT	The bit name for the logging statistics event class (Class 0); same as bit 0. You must always log this event class.

Discussion

You can abbreviate all bit names to as few as two characters. For example, the following abbreviations are all valid entries for EVENT:

```
EV
EVE
EVEN
```

However, all characters that you specify must be valid. For example, EVX is not valid.

You can examine the value of *logMask* using the NSINF U command.

If no parameters are specified, the log mask is unchanged.

LOGCHG error messages are listed in the *NS-ARPA/1000 Error Message and Recovery Manual*.

Examples

Assuming that the log mask is 161B, all of the following LOGCHG commands set the log mask to 171B (log logging statistics, warnings, severe errors, and disasters).

```
CI>LOGCHG,171B
CI>LOGCHG,171
CI>LOGCHG,+WARNING
CI>LOGCHG,*+WARNING
CI>LOGCHG,+ WARNING
CI>LOGCHG,* 10
CI>LOGCHG,RESOURCELIM DISASTER ERROR WARNING LOGSTATS
```

EVMON Output

EVMON writes log records to a file or device. EVMON output consists of a two-line EVMON *header* followed by the *log records*. A log record contains a *log record header* followed by a *log message*. The log record header format is the same for all records. However, a field's meaning in the log message may vary according to the protocol or service that is logging the message. Log messages are listed in the *NS-ARPA/1000 Error Message and Recovery Manual*. Figure 4-1 shows the format for EVMON output.

```
   currentTime
                              Event Log at nodeName

currentTime
 EventClass  Entity   Location   xx   yy     LogMask   ProcessName/Session

 LogMessage

currentTime
 EventClass  Entity   Location   xx   yy     LogMask   ProcessName/Session

 LogMessage
                                        :
                                        :
```

Figure 4-1. EVMON Output

EVMON Header Fields

currentTime Current system time. The *currentTime* is not printed if there is less than one second difference between it and the last *currentTime* printed.

nodeName Name of the node at which EVMON is running.

Log Record Header Fields

currentTime Current system time. The *currentTime* is not printed if there is less than one second difference between it and the last *currentTime* printed.

EventClass Type of log record. There are seven event classes, as shown earlier in Table 4-1. The event classes listed in the log record header are as follows:

- `LogStat` (Class 0).
- `ProLog` (Class 1, protocol-specific information).
- `Event` (Class 2, event messages).
- `Warning` (Class 3, warnings).
- `Error` (Class 4, severe errors).

- `Disaster` (Class 5, disasters).
- `RsrceLim` (Class 6, resource limitations).

`Entity` Indicates the software module, procedure, process, or protocol handler that reports the log record and/or message. A protocol handler is a set of software procedures that implement a protocol. The procedures that constitute a protocol handler may reside in different processes.

The entity printed in the *`Entity`* field is not necessarily the same as the entity that *encountered* the error.

The entities are as follows:

- `ARM` (Address Resolution Module)
- `ARP` (Address Resolution Protocol)
- `ETHERNET` (Ethernet protocol handler)
- `HP-FTP` (File Transfer Protocol)
- `HP-IFP` (Interface Protocol handler, used for DS/1000-IV Compatible Services over IEEE 802.3)
- `HP-IP` (Internet Protocol handler)
- `HP-NFT` (Network File Transfer subsystem)
- `HP-PROBE` (Probe protocol handler)
- `HP-PXP` (HP Packet Exchange Protocol handler)
- `HP-ROUTER` (Router/1000 protocol handler)
- `HP-RPM` (Remote Process Management)
- `HP-TCP` (Transmission Control Protocol)
- `HP-TN` (TELNET)
- `ICMP` (Internet Control Message Protocol; used for PING)
- `IEEE-802` (IEEE 802.3 protocol handler)
- `INPRO` (Inbound message handler)
- `OUTPRO` (Outbound message handler)
- `PROSW` (NS-ARPA module that delivers event messages between the protocol and event handlers in INPRO and OUTPRO)
- `RUNTIME` (These are Pascal-detected runtime errors)
- `SIGMOD` (Signal module; used by NetIPC and contains some NetIPC routines)
- `SREG` (Socket Registry modules)
- `TIMER` (Timer monitor subsystem)
- `UPLIN` (UPLIN module)

Note The Entity `RunTime` is printed in the *`Entity`* field of the log record header when the NS-ARPA software encounters a Pascal-detected runtime error. These errors require HP notification.

Location One-word integer that indicates the section (procedure, module, or process) of the Entity's software that logged the event. Some Entities use the location code when the event class indicates a warning (Class 3), error (Class 4), disaster (Class 5), or resource limit condition (Class 6). Location codes are documented in *NS-ARPA/1000 Error Messages and Recovery Manual*. The Entities that use the location code in this way are:

- INPRO
- OUTPRO
- HP-PROBE
- PROSW
- SIGMOD
- SREG

xx For HP internal use only.

yy For HP internal use only. LAN driver error codes are printed to the *yy* field. These errors are explained in the *HP 12076A LAN/1000 Link Node Manager's Manual*.

LogMask Octal value of EVMON's log mask at the time the log record was sent.

ProcessName Five-character name stored in the ID segment of the process that sent the log message.

session Session ID (the scheduling terminal's LU) of the session that scheduled the process in the previous field. For NS-ARPA subsystem processes (INPRO, OUTPR, and the NS-ARPA monitors), this is the system session ID. For NetIPC processes, *session* is the user's session ID.

Log Message Field

LogMessage Log Message Codes and ASCII Log Messages are sent to the log file by certain software modules, procedures, processes, or protocol handlers. Some Entities use Log Message Codes when the event class indicates a warning (Warning, Class 3), error (Error, Class 4), disaster (Disaster, Class 5), or resource limit condition (RsrceLim, Class 6). Log Message Codes and ASCII Log Messages are documented in *NS-ARPA/1000 Error Messages and Recovery Manual*.

The following Entities print their Log Message Codes in the following numeric ranges:

- 0-999—generic, includes codes 100 through 599 for HP-NFT
- 1000-1999—Memory Manager
- 2000-2999—Multiuser
- 3000-3999—HP-IP
- 4000-4999—IEEE-802/ETHERNET

- 5000-5999—HP-PXP
- 8000-8999—HP-ROUTER
- 14000-14999—HP-TCP
- 15000-15999—TIMER
- 17000-17999—UPLIN
- 18000-18999—HP-RPM
- 20000-20999—HP-TN
- 21000-21999—HP-FTP
- 24000-24999—ARM
- 25000-25999—ARP
- 26000-26999—ICMP

The following Entities have ASCII Log Messages:

- HP-IFP
- HP-ROUTER
- INPRO
- OUTPRO

5

NS-ARPA Message Tracing

Overview

This section describes the NS-ARPA message tracing utility.

Message tracing is the recording of messages as they enter and exit a node. You can use message tracing to help you debug user applications or to analyze and troubleshoot network problems between two nodes. However, because message tracing may degrade system performance, HP recommends that you do not enable tracing unless you are actively debugging a user application or a network problem. Messages are listed in the *NS-ARPA/1000 Error Messages and Recovery Manual*.

NSTRC, which is part of the NS-ARPA/1000 message tracing utility, allows you to record messages, including data, as they are sent by NetIPC via a socket, and as they are physically transmitted by an interface card through the network. NSTRC records these messages in binary form to a file.

The message tracing utility consists of three parts:

- NSTRC, which enables message tracing
- BRTRC, which terminates message tracing
- FMTRC, which formats the trace file and allows you to select trace records according to nodes, Link Interfaces, and sockets

This section contains the following subsections:

- Using Message Tracing
- NSTRC
- BRTRC
- FMTRC
- Trace File Formats

Using Message Tracing

The procedure for using message tracing is as follows:

1. Enable message tracing.
2. Run the user program or generate NS-ARPA activity (for example, use NFT or FTP to copy a file). If you are debugging a user NetIPC program, use the NSINF `P` command to get the program's Global Socket Descriptors (GSDs).
3. When the program or activity terminates, halt tracing by running BRTRC.
4. Format and examine the trace file using FMTRC. If you are debugging a user NetIPC program, examine the socket level trace records for the program's GSDs. If you are debugging a program that uses DS/1000-IV Compatible Services, format RTR messages.
5. If you suspect a link problem, run tracing at two nodes and examine the network trace records at both ends of the link for symmetry.

NSTRC

Enables message tracing.

Runstring

```
[XQ,]NSTRC[,traceFile][,errorLog][,recordLength] [,N[ETWORK]]
                                                 [,S[OCKET] ]
                                                 [,B[OTH]   ]
```

Parameters

traceFile

File to which you want NSTRC to write the trace records. While NSTRC is executing, it writes the trace records to its VMA partition. NSTRC writes to the partition in a circular manner; if NSTRC has filled the partition, it overwrites the earliest records. NSTRC uses approximately one-fourth of the trace file for accounting. For each remaining page in the partition, NSTRC can write 10 trace records. Therefore, if NSTRC's VMA partition is *n* pages long, NSTRC can write approximately `(n * 3/4) * 10` trace records before it must overwrite previous records.

If you enter the name of an existing file, NSTRC overwrites the existing file. If you use the default *traceFile* and a file by that name exists, NSTRC prints an error message and terminates.

Default: `NS_TRACE.TRC` in the current working directory.

errorLog

Device (file or device LU) to which you want NSTRC to report run-time errors.

Default: local LU 1 (scheduling terminal).

recordLength

The trace record length, in words. For network level tracing, the trace record size is approximately the total size of all protocol headers. If a message is longer than `recordLength`, NSTRC truncates the message.

Range: 24 to 60 words.

Default: 60 words.

`NETWORK`	(*Default*) Trace messages at the network level. NSTRC records messages at the LI software level as they enter or leave the node. In addition, NSTRC records any Probe, Socket Registry, NFT, TCP, and IP control messages (control messages are messages sent by the protocol modules as part of the protocol and do not contain user data), and HDLC/Multidrop "Link Up" messages. (You can use FMTRC to format the IP or LI header (OSI layer 3s--IEEE 802.3 LAN, Ethernet LAN, or Router/1000). FMTRC does not format higher-level protocol headers, such as TCP or PXP unless the FMTRC `nice` option is used. See the FMTRC Dialogue subsection for more information on the conversion mode `nice` option.) *Useful for analyzing network problems*.
`SOCKET`	Trace messages at the socket level. NSTRC records messages as they are sent and received on user sockets. (You can select messages according to Global Socket Descriptors when you format the trace file via FMTRC.) *Useful for debugging user applications*.
`BOTH`	Trace messages at both the network and socket level.
	Default: `NETWORK`

Discussion

If the NSTRC program is loaded with its working set size equivalent to its VMA size, the entire trace file will be in memory when NSTRC runs. If the NSTRC VMA size is larger than its working set size, there will be a page fault every 10 trace records, slowing the performance of NSTRC.

NSTRC must be loaded with a VMA size that is a multiple of 32.

BRTRC

BRTRC terminates NSTRC and halts tracing.

Runstring

```
[RU,]BRTRC
```

Discussion

To format and examine the current trace file, you must terminate NSTRC. To do this, run BRTRC. BRTRC sets NSTRC's break bit and then sends a termination message to NSTRC. The termination message is necessary because NSTRC is normally suspended, waiting on its class number for a trace record, and NSTRC will not terminate until it receives another trace record. Before terminating, NSTRC releases its class number, updates the file so that it knows where the first trace record is, and closes the trace file.

Caution If you abort NSTRC (instead of terminating it with BRTRC), FMTRC will not be able to format the trace file.

FMTRC

FMTRC formats NSTRC trace files so that they are easier to read. Unformatted NSTRC files (raw files), are binary files. FMTRC formats these files by separating header fields and data, and by labeling header fields. In addition, FMTRC allows you to select messages by time, link interface, source and destination nodes, user services, and sockets.

Runstring

```
[RU,]FMTRC[,inputDevice][,rawTraceFile][,formattedDevice]
          [,titleField][,logDevice]
```

Parameters

inputDevice Device (answer file name or interactive device LU) that provides input for FMTRC dialogue.

Default: local LU 1 (scheduling terminal).

rawTraceFile The raw trace file for FMTRC to format. This must be a disk file; if NSTRC posted trace messages to a tape, you must restore the tape's contents to a disk file before formatting it. NSTRC must not have the file open for tracing.

Default: `NS_TRACE.TRC`.

formattedDevice Device (file name or device LU) to which FMTRC outputs the formatted trace file.

Default: If you specified a file for *rawTraceFile*, FMTRC takes that file name, strips any existing type extension, and adds the type extension FMT to form the name for the formatted (output) file. If you did not specify a *rawTraceFile*, the default is the file `NS_TRACE.FMT`.

titleField An ASCII string of up to 72 characters to label the output file. FMTRC prints *titleField* in a heading at the beginning of the output file. The string for *titleField* may be empty; however, if it is not empty it must not contain embedded blanks. This is because the command interpreter interprets embedded blank characters as parameter delimiters.

logDevice Device (file name or device LU) to which FMTRC logs any errors.

Default: local LU 1 (scheduling terminal) if you are running FMTRC interactively, `FMTRC.LOG` if you specified an answer file. If `FMTRC.LOG` already exists, FMTRC overwrites it.

Discussion

You can schedule FMTRC interactively or programmatically with an EXEC or FMP call. FMTRC uses a dialogue to prompt the user for formatting options. You can also specify an answer file (input device) that contains the responses to the dialogue. The FMTRC dialogue is further explained later in "FMTRC Dialogue."

VMA Size

FMTRC's VMA size must equal the size of the raw trace file that it is to format. The size of the trace file may be determined by the RTE `DL` command (directory list). The `DL` command shows file sizes in blocks (128 words per block). However, the VMA size given to LINK is in pages (1024 words per page). To determine the correct size for FMTRC, divide the raw trace file size as shown by the `DL` command by eight. Relink FMTRC if necessary.

Error Handling

If FMTRC terminates with an error, it returns a positive error value in its first return parameter. For an explanation of the errors, refer to *NS-ARPA/1000 Error Messages and Recovery Manual*.

If FMTRC encounters a trace record that it cannot format, it prints an error message to its log device and in the formatted file. In the formatted file, FMTRC prints the contents of the trace record below the error message. The trace record contains the message and the control buffer, unformatted, in octal (or hexadecimal) and ASCII.

Dialogue Syntax

You can use commas or blank spaces to separate items in a multi-item response. For example, you can enter `RTR,802` or `RTR 802`.

FMTRC upshifts any lowercase replies.

Besides replying to FMTRC questions with specific values, you can enter the following responses:

RETURN or /D	Use the default value.
/A or /AB	Abort FMTRC processing.

Answer Files

FMTRC treats any line beginning with an asterisk (*) as a comment.

Blank lines are interpreted as RETURN (use the default value).

If an EOF is reached before all FMTRC questions are satisfied, FMTRC uses the default values for the remaining questions.

FMTRC Dialogue

If you are running FMTRC with an answer file, FMTRC skips to read a response to the prompt "Enter a raw trace file name". If you are running FMTRC interactively, FMTRC prints the FMTRC default parameters and options. The meanings of these parameters and options are explained below.

```
FMTRC Default Parameters and Options :

Raw trace filename : NS_TRACE.TRC
Formatted trace filename : NS_TRACE.FMT
Title field :   { Blank }
Trace level :         both   (Socket and Network)
Time window, start :  00:00:00:00 (all times)
Time window, stop :   23:59:59:99
Link Interface(s) :  all   (802, RTR)
Source and Destination nodes :  all  ( all traffic)
Monitor(s)/Service(s) :  all  (NFT, SR, MA, RR, INPRO, TRFAS,
                               PTOP, RFA, EXECM, EXECW, DLIST, OR)
Socket(s) :  all   (all Sockets)
Conversion :  nice
```

FMTRC displays the above defaults if you did not specify any parameters in the runstring.

If you specified a raw trace file name, formatted trace file name, or title field, FMTRC prints the names or title field that you specified as the defaults.

If you specified a raw trace file name but no formatted file name, FMTRC takes the raw trace file name, strips any existing type extension, and adds the type extension FMT to form the name for the default formatted (output) file.

FMTRC asks the following question only if you are running it interactively. Do not enter a response to this question if you are preparing an answer file.

```
Do you want to modify these parameters and options [ Y/(N) ]
Fmtrc:
```

> Enter Y (YES) to modify any of the above FMTRC parameters; otherwise, enter N (NO). If you enter N, FMTRC attempts to format all messages in the file NS_TRACE.TRC.

If you enter N, FMTRC skips the remaining questions and attempts to format all messages in the file NS_TRACE.TRC. Otherwise, FMTRC prompts:

```
Enter a raw trace file name [ Default = NS_TRACE.TRC ]
Fmtrc:
```

> Enter the name of the raw trace file for FMTRC to format. This must be a disk file; if NSTRC posted trace messages to a tape, you must restore the tape's contents to a disk file before formatting it.

```
Enter a formatted trace file name [ Default = NS_TRACE.FMT ]
Fmtrc:
```

Enter the file name or device LU to which FMTRC will write the formatted trace file.

If you enter the name of an existing file, FMTRC will ask for your permission to overwrite the file at the end of the FMTRC dialogue.

```
Enter the title you would like to have on the formatted output.
Fmtrc:
```

Enter an ASCII string of up to 72 characters to label the output file. FMTRC will print this string in the heading at the beginning of the output file.

```
Enter Trace level [ Default = Socket and Network ]
Fmtrc:
```

Enter `S` (Socket) to format socket trace records—socket trace records messages, including data—as they are sent by NetIPC via sockets.

Enter `N` (Network) to format network trace records—network trace records messages, including data—as they are physically transmitted by interface cards through the network.

```
Enter a start time [ Default = 00:00:00:00 ]
Fmtrc:
```

FMTRC only formats trace records with a time stamp that is as late or later than the start time you specify. Enter the start time in the form *hh:mm:ss:cc*, where:

- *hh* is the hour, using a 24-hour clock
- *mm* is the minute
- *ss* is the second
- *cc* is the centisecond

For example, 20:05:37:00 is 8:05:37 PM.

```
Enter a stop time [ Default = 23:59:59:99 ]
Fmtrc:
```

FMTRC does not format any trace records with a time stamp later than the stop time you specify. Enter a stop time using the same format as the start time.

```
Enter Link Interface(s) [Default = 802 and RTR ]
Fmtrc:
```

Enter the names of the LI types for which you want FMTRC to format trace records, using the following naming convention:

LI Type	Name
Both IEEE 802.3 and Ethernet LAN	`802`
Router/1000	`RTR`

If you want to format only the messages between two specific nodes, enter `NONE`, and go to the next question.

FMTRC can selectively format trace records according to the Link Interface (LI) over which the traced message was transmitted. FMTRC formats the trace records of the messages transmitted to or from the local node over the specified LI type. If you specify RTR (Router/1000), FMTRC will also format any messages used for DS/1000-IV Compatible Services (RTE-RTE), even if the messages were transmitted over other links. This feature is useful for tracing DS/1000-IV Compatible Services (RTE-RTE).

`Enter Link Interface, Source, Destination nodes [ Default = all ]`
`Fmtrc:`

Enter the LI (using the LI naming convention from the previous question), and the addresses of the the source and destination nodes of the messages for FMTRC to format.

Name	Address format
`RTR`	Router/1000 address; an integer between 1 and 32767
`802`	LAN station address; *`hh-hh-hh-hh-hh-hh`*, where *`h`* is a hexadecimal digit. Either IEEE 802.3, Ethernet, or both.

For example, `802,08-00-09-02-33-9C,09-00-09-00-00-01` is a valid entry.

Refer to the section "Network Configuration" in the *NS-ARPA/1000 Generation and Initialization Manual* for more information on address forms.

In addition to formatting messages according to LI type or types specified for the previous question, FMTRC can selectively format trace records according to the Link Interface (LI) that the traced message was transmitted over and according to the source node and destination node.

FMTRC formats messages between the specified source and destination node, including messages that originate at the destination node. FMTRC does not format any store-and-forward messages through the two nodes (a store-and-forward message is a message that originates at a node other than one of the specified nodes).

FMTRC formats the records selected according to what you enter here *in addition* to any trace records selected according to the LI type or types that you specified for the previous question. For example, if you specified the `802` LI for the previous question and entered `RTR,50,2` here, FMTRC would format the trace records of all messages transmitted to and from the local node via the IEEE 802.3 LI *and* all messages transmitted between node 50 and node 2 via a Router/1000 LI.

If you specify RTR (Router/1000), FMTRC formats any messages between the two nodes that are used for DS/1000-IV Compatible Services (RTE-RTE), even if the messages were transmitted over other links. This feature is useful for tracing DS/1000-IV Compatible Services (RTE-RTE).

`Enter (up to 3) Monitors/Services [ Default = all]`
`Fmtrc:`

If you are running FMTRC interactively, it gives the above prompt only if you specified Socket level or RTR LI trace formatting.

If you did not specify Socket level or RTR LI trace formatting and you are running FMTRC with an answer file, FMTRC ignores the reply to this question.

FMTRC can selectively format trace records of monitors or user services. The user services are Network File Transfer and Socket Registry, and are valid for Socket level tracing only. Enter `/D` or RETURN if you want FMTRC to format trace records for all the listed monitors and user services. Otherwise, enter the names of up to three services that you want messages formatted for, using the naming convention listed below.

Name	Service
DLIST	Remote Directory List (Stream number 1). Relevant for DS/1000-IV Compatible Services only.
EXECM	Remote EXEC (Stream number 5). Relevant for DS/1000-IV Compatible Services only.
EXECW	Remote EXEC (Stream number 3). Relevant for DS/1000-IV Compatible Services only.
INPRO	NS Common Services sent over Router/1000 links or DS/1000-IV Compatible Service messages encapsulated in TCP/IP headers and sent over Router/1000 links. Therefore, it is relevant for Router/1000 LIs only.
MA	Message Accounting control messages. Only relevant for DS/1000-IV Compatible Services.
NFT	Network File Transfer messages. Relevant for Socket level tracing only.
OR	Remote Operator Requests (Stream number 7). Relevant for DS/1000-IV Compatible Services only.
PTOP	Program-to-Program Communication (Stream number 4). Relevant for DS/1000-IV Compatible Services only.
RFA	Remote File Access (Stream number 6). Relevant for DS/1000-IV Compatible Services only.
RR	Dynamic Rerouting control messages. Only relevant for Router/1000 LIs with Dynamic Rerouting.
SR	Socket Registry Relevant for Socket level tracing only.
TRFAS	RTE Transparent File Access (Stream number 12). Relevant for DS/1000-IV Compatible Services only.

The trace records for NFT and SR are subsets of Socket level tracing. FMTRC formats trace records for these services only if you have also specified Socket level tracing. The trace records for DLIST, EXECM, EXECW, INPRO, MA, OR, PTOP, RFA, RR, and TRFAS are subsets of Network level tracing for DS/1000-IV Compatible Services. FMTRC formats trace records for these services only if you have also specified Network level tracing for RTR LIs.

If you are running FMTRC interactively, it prompts for the following only if you specified Socket level trace formatting.

```
Enter (up to 3) Sockets [ Default = all]
Fmtrc:
```

If you did not specify Socket level trace formatting and you are running FMTRC with an answer file, FMTRC ignores the reply to this question.

Enter the Global Socket Descriptors of up to three sockets that you want FMTRC to format trace records for, or enter `/D` or RETURN and FMTRC will format trace records for all sockets. Use the NSINF `P` command to get the value of a program's Global Socket Descriptors.

```
Enter conversion type [ Default = nice]
Fmtrc:
```

FMTRC can format the trace data in octal, hexadecimal, or nice mode (`O, H, or N`). Nice mode parses protocol messages (for example, tcp, ip, probe) and displays the fields for each protocol. Some protocols will not be formatted in nice mode, but the type and a hex dump will be output. For a description of the protocols refer to the *NS Message Formats Reference Manual*, part number 5958-8523.

Duplicate Formatted File

At the end of FMTRC dialogue, if the file you specified for the formatted trace file already exists, FMTRC prints the following message:

OK? to overwrite formatted file: *fileName* [(Y)/N] :

Enter `Y` if you want FMTRC to overwrite the file; otherwise, enter any other character, and FMTRC will terminate.

Dialogue Examples

The following interactive and answer file dialogues produce the same FMTRC formatting.

Interactive

```
FMTRC Default Parameters and Options :

Raw trace filename : NS_TRACE.TRC
Formatted trace filename : NS_TRACE.FMT
Title field :   { Blank }
Trace level :         both   (Socket and Network)
Time window, start :  00:00:00:00 (all times)
Time window, stop :   23:59:59:99
Link Interface(s) :  all   (802, RTR)
Source and Destination nodes :  all    ( all traffic)
Monitor(s)/Service(s) :  all  (NFT, SR, MA, RR, INPRO, TRFAS,
                               PTOP, RFA, EXECM, EXECW, DLIST, OR)
Socket(s) :  all   (all Sockets)
Conversion :  nice

Do you want to modify these parameters and options [ Y/(N) ]
Fmtrc:  Y

Enter a raw trace filename [ Default = NS_TRACE.TRC ]
Fmtrc:  TRACE.ONE

Enter a formatted trace filename [ Default = TRACE.FMT ]
Fmtrc:  FORMAT.ONE

Enter any title you would like to have on the formatted output
Fmtrc:
Testing with nodes 50 and 71

Enter Trace level? [ Default = Socket and Network ]
Fmtrc:  RETURN

Enter a start time [ Default = 00:00:00:00 ]
Fmtrc:  12:30:00:00

Enter a stop time [ Default = 23:59:59:99 ]
Fmtrc:  12:35:00:00

Enter Link Interface(s) [ Default = 802 and RTR ]
Fmtrc:  802

Enter Link Interface, Source, Destination nodes [ Default = all ]
Fmtrc:  RTR 50 70
```

```
Enter (up to 3) Services [ Default = all ]
Fmtrc:  NFT DLIST

Enter (up to 3) Sockets [ Default = all ]
Fmtrc:  RETURN

Enter conversion type [ Default = nice]
Fmtrc:  RETURN
```

Answer File

```
**Filename:  FMTRC1.CMD
**Created August 14, 1989
**COMMAND FILE ONE FOR NS-ARPA/1000, Node 60
**
**Raw trace
TRACE.ONE
**
**Formatted trace
FORMAT.ONE
**
**Title field
Testing with nodes 50 and 71
**
**Trace level,both
/D
**
**Start time
12:30:00:00
**
**Stop time 12:35:00:00
**
**Link Interface
802
**
**Link Interface, Source node, Destination node pair
RTR,50,70
**
**Services
NFT DLIST
**
**Socket, all
/D
**
**Conversion mode
/D
**
**END
```

Trace File Formats

FMTRC prints a header at the beginning of each formatted trace file that specifies the following information:

- the time at which FMTRC began formatting the raw trace file
- the node at which the raw trace file was created
- the title, as specified by the user
- the name of the raw trace file
- the time and date that tracing was enabled and disabled
- the number of trace records that NSTRC logged
- the number of trace records NSTRC dropped
- the number of trace records NSTRC overwrote (NSTRC treats the raw trace file as a circular file)
- the formatting options selected

The following is an example of a trace file header.

```
Network Services/1000                6:34 PM TUE., 10 NOV., 1992

Originating node of trace : JOAN.LAN2.HP
SAMPLE USER TRACE

Trace file : TIME.TRC

Trace started : 6:28 PM TUE., 10 NOV., 1992
Trace stopped : 6:28 PM TUE., 10 NOV., 1992

     8 trace records logged.     0 trace records overwritten.
     0 trace records dropped.

Format file : TIME.FMT

Trace Level : SOCKET NETWORK
Time window, start : 00:00:00:00
Time window, stop : 23:59:59:99
Protocol(s) : 802 RTR
Source and Destination nodes : all traffic
Monitor(s)/Service(s) : all monitors/services
Socket(s) : all sockets
Conversion mode : NICE
```

Figure 5-1. Example Trace File Header

Trace Record Formats

FMTRC prints trace records in the following formats:

- Socket level
- IEEE 802.3, Ethernet (network level)
- Router/1000 (network level)

In addition, FMTRC prints messages from HDLC and Multidrop interface drivers when logical link connections are established (Link Up messages).

The trace record borders have the following meanings:

- the letter n (`n`) indicates network level trace records
- the asterisks (`*`) indicate socket level trace records
- the plus signs (`+`) indicate Router/1000 trace records for DS/1000-IV Compatible Services over an 802 link.

FMTRC formats some protocol header information, and prints the protocol headers from left to right, top to bottom, one word at a time in octal, hexadecimal, or nice. Protocol header formats are fully described in the *NS Message Formats Reference Manual*. Note that the following descriptions of the message contents assume that the entire message fits into the trace record. If the trace record is shorter than the message length, NSTRC truncates the message and some of the items listed may not appear in the formatted record.

NICE Formatted Records

The NICE conversion mode is the default mode for formatting trace records. Following are examples of messages formatted with the NICE option:

```
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
n      18:34:08:56                                    NS/1000          n
n      AMELIA.HP.COM                                                   n
n                    Message type = Network Level                      n
n                                INBOUND                               n
n   Message length   =     30          Link Interface =    ETHER      n
n   Sequence No.     =  18688          Socket         =               n
n   Node One = 08-00-09-00-02-21   Node Two = 08-00-09-09-1D-68       n
n---------------------------- IP Header ----------------------------n
n  Version:  4   IHL:  5   TOS:  0x00      Length:  40 (0x28)         n
n    Ident:  0x7197      Flags:  NONE    Frag.Off:  0                 n
n      TTL:  30       Protocol:  TCP     Checksum:  0x5B50           n
n  Source:  015.000.089.049      Destination: 015.000.088.184         n
n--------------------------- TCP Header-----------------------------n
n  SPort:  1180       DPort:  23          Data Len: 0                 n
n  Seq:  1208704384 (0x480B5D80)   Ack:  41675 (0xA2CB)               n
n       Window:  4096 (0x1000)       Flags:  ACK                      n
n       Offset:  5           Checksum:  0x82E1       Urgent:  0      n
n--------------------------Control Buffer-----------------------------n
n   Socket: 0     Sequence: 18688    LI: LAN    LU: 132               n
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
```

Figure 5-2. Example TCP Trace Record in Nice Format

```
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
n      18:34:08:59                                    NS/1000          n
n      AMELIA.HP.COM                                                   n
n                    Message type = Network Level                      n
n                                INBOUND                               n
n   Message length =       47          Link Interface =      802      n
n   Sequence No.   =    18689          Socket         =               n
n   Node One = 09-00-09-00-00-01    Node Two = 08-00-09-00-A2-1F      n
n--------------------------LAN Header-------------------------------n
n      Probe FC-FC Control=03  DXSAP=05-03 SXSAP=05-03                n
n----------------------- PROBE Header-------------------------------n
n  Version: 0  Sequence: 0  Message: 21 Unsolicted Reply              n
n  Name (1) GROUCHO1K.HP.COM                                          n
n  Version: 0   Domain: 1   IP Addr: 015.000.088.209                  n
n   Path Report       Serv.PID: FF    Serv. Map: 177777               n
n    Trans.PID: FE  Trans. Map: 177777   IP PID: 08  IP SAP: 0       n
n     IEEE PID: 00    IEEE SAP: 1    Address: C0-06-46-0B-00-0C      n
n--------------------------Control Buffer-----------------------------n
n   Socket: 0     Sequence: 18689    LI: LAN    LU: 132               n
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
```

Figure 5-3. Example Probe Trace Record in Nice Format

```
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
n      18:34:12:49                                    NS/1000           n
n      AMELIA.HP.COM                                                    n
n                    Message type = Network Level                       n
n                                INBOUND                                n
n   Message length   =     60         Link Interface =       802      n
n   Sequence No.     =  18706         Socket         =         58     n
n   Node One = 08-00-09-00-02-21    Node Two = 08-00-09-15-E5-5F      n
n----------------------------LAN Header-------------------------------n
n      length=205  DSAP=06   SSAP=06   Control=03                      n
n---------------------------- IP Header-------------------------------n
n  Version:  4   IHL:  5   TOS:  0x00       Length:  202 (0xCA)       n
n    Ident:  0x52BA      Flags:  NONE     Frag.Off:  0                 n
n      TTL:  30       Protocol:  TCP      Checksum:  0x798A           n
n  Source:  015.000.089.050      Destination:  015.000.088.184        n
n-------------------------- TCP Header-------------------------------n
n  SPort:  1242       DPort:  1536       Data Len:  162 (0xA2)        n
n  Seq:  56523362 (0x35E7A62)      Ack:  32081 (0x7D51)               n
n       Window:  700 (0x2BC)         Flags:  ACK PSH                  n
n       Offset:  5           Checksum:  0x813D       Urgent:   0      n
n------------------------ NFT Header-------------------------------n
n    EOD  Length is 162                                               n
n------------------------------Message--------------------------------n
n 00 00 00 99 31 20 20 20 20 50 61 73 63 61 6C 2F  ....1    Pascal/  n
n 31 30 30 30 20 20 20 20 20 20 20 20 20 20 20 20  1000              n
n 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                    n
n 20 20 20 20 20 20 20 54 68 75 20 00                      Thu .     n
n-------------------------Control Buffer-----------------------------n
n   Socket: 0     Sequence: 18706    LI: LAN    LU: 132               n
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
```

Figure 5-4. Example NFT Trace Record in Nice Format

```
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
n      18:34:12:82                                    NS/1000           n
n      AMELIA.HP.COM                                                    n
n                    Message type = Network Level                       n
n                                INBOUND                                n
n   Message length   =     30           Link Interface =    ETHER     n
n   Sequence No.     =  18710           Socket         =              n
n   Node One = FF-FF-FF-FF-FF-FF    Node Two = 08-00-09-02-A3-0A      n
n-----------------------------ARP REQUEST----------------------------n
n           Hardware:  Ethernet             Protocol:  IP             n
n  Hardware addr len:  6           Protocol addr len:  4              n
n             Source:  015.000.088.082    08-00-09-02-A3-0A          n
n        Destination:  015.000.088.164                                 n
n-------------------------Control Buffer-----------------------------n
n   Socket: 0     Sequence: 18710    LI: LAN    LU: 132               n
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
```

Figure 5-5. Example ARP Request Trace Record in Nice Format

Please refer to the *NS Message Formats Reference Manual* for information on the specific field outputs.

Socket Trace Records

Socket trace records have the format shown in Figure 5-6.

```
**********************************************************************
*      time                                                NS/1000     *
*      originating node                                                *
*                    Message type  = Socket Level                      *
*                                direction                             *
*   Message length   =  length           Monitor/Service ID = service*
*                                        Socket       =      gsd     *
*-----------------------------Message--------------------------------*
*   nnnnnn  nnnnnn  nnnnnn  nnnnnn  nnnnnn  nnnnnnn  nnnnnn  nnnnnn  *
*   xx xx   xx xx   xx xx   xx xx   xx xx    xx xx   xx xx   xx xx   *
*   nnnnnn  nnnnnn  nnnnnn  nnnnnn  nnnnnn  nnnnnnn  nnnnnn  nnnnnn  *
*   xx xx   xx xx   xx xx   xx xx   xx xx    xx xx   xx xx   xx xx   *
**********************************************************************
```

Figure 5-6. Socket Trace Record Format

Socket Trace Record Fields

time The time that the NSTRC posted the trace record, using a 24-hour clock.

originating node The node at which the raw trace file was created.

direction The direction of the message (INBOUND or OUTBOUND), relative to the originating node. FMTRC prints inbound messages flush with the left margin and outbound messages flush with the right margin.

length The length of the data buffer, as sent or received by the socket, in words. FMTRC will round up this value if the message has an odd number of bytes.

service The service that owns the socket (NFT, Network File Transfer, or SR, Socket Registry). FMTRC determines this from bits 8 and 9 of the second word in the data buffer. If bit 8 is set but not bit 9, FMTRC labels the service NFT. If bits 8 and 9 are set, FMTRC labels the service SR. For messages sent by user sockets, FMTRC usually leaves this field blank. However, if a user data buffer matches one of the above bit patterns, FMTRC will label the service field. Check the Global Socket Descriptor to determine the socket owner or filter trace records according to the program's Global Socket Descriptor.

gsd The Global Socket Descriptor of the socket that sent or received the data buffer.

nnnnn	The octal (or hexadecimal) value of a word in the data buffer. The data buffer is printed from left to right, top to bottom, one word at a time, in octal (or hexadecimal). Below the octal (or hexadecimal) value of the word is the ASCII representation.
xx xx	The ASCII representation of a word in the data buffer.

DS/1000-IV Compatible Services

Socket level tracing may include outbound messages for DS/1000-IV Compatible Services sent over an 802 link. This is because these messages are moved from `#MAST` or `#SLAV` to IFP's socket for transmission. For these messages, the data buffer contains (from left to right, top to bottom):

- the IFP header
- the data portion of the Router/1000 message
- the header portion of the Router/1000 message

The value of *length* is the sum of the lengths of the three above items.

LAN Trace Records

LAN trace records have the format shown in Figure 5-7 when the octal (O) or hexadecimal (H) formatting conversion type is specified.

```
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
n      time                                                  NS/1000     n
n      originating node                                                  n
n                    Message type  = Network Level                       n
n                                direction                               n
n   Message length   = length           Link Interface = 802/ETHER      n
n   Sequence No.     = seq #            Socket         =      gsd        n
n   Node One = address one          Node Two = address two               n
n----------------------------LAN Header-------------------------------n
n   nnnnnn  nnnnnn  nnnnnn  nnnnnn  nnnnnn  nnnnnnn  nnnnnn  nnnnnn  n
n   xx xx   xx xx   xx xx   xx xx   xx xx    xx xx   xx xx   xx xx   n
n   nnnnnn  nnnnnn  nnnnnn  nnnnnn  nnnnnn  nnnnnnn  nnnnnn  nnnnnn  n
n   xx xx   xx xx   xx xx   xx xx   xx xx    xx xx   xx xx   xx xx   n
n---------------------------Message----------------------------------n
n   nnnnnn  nnnnnn  nnnnnn  nnnnnn  nnnnnn  nnnnnnn  nnnnnn  nnnnnn  n
n   xx xx   xx xx   xx xx   xx xx   xx xx    xx xx   xx xx   xx xx   n
n---------------------------Control Buffer---------------------------n
n   nnnnnn  nnnnnn  nnnnnn  nnnnnn                                    n
n   xx xx   xx xx   xx xx   xx xx                                    n
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
```

Figure 5-7. LAN Trace Record Format

LAN Trace Record Fields

time — The time that the NSTRC posted the trace record, using a 24-hour clock.

originating node — The node at which the raw trace file was created.

direction — The direction of the message (INBOUND or OUTBOUND), relative to the originating node. FMTRC prints inbound messages flush with the left margin and outbound messages flush with the right margin.

length — The number of words used to store the message in DSAM, including all protocol headers. Therefore, this value may be greater than the actual length of the message if the message does not fill the entire memory structure allocated for it.

seq # — The trace record's sequence number. This number is used internally by FMTRC.

gsd — The Global Socket Descriptor of the socket that sent the message. This field may be blank for inbound messages.

address one	The LAN station address of the destination node. If the address is a multicast address, then the message is either a Probe request or reply.
address two	The station address of the source node. If the address is a multicast address, then the message is either a Probe request or reply.
nnnnn	The octal (or hexadecimal) value of a word in the LAN trace record header, message buffer or control buffer. The data buffer is printed from left to right, top to bottom, one word at a time, in octal (or hexadecimal). Below the octal (or hexadecimal) value of the word is the ASCII representation.
xx xx	The ASCII representation of a word in the LAN trace record header, message buffer or control buffer.

LAN Header

The area below the line marked `LAN Header` contains the LAN trace record header. For a description of the LAN header, refer to *HP 12076A LAN/1000 Link Node Manager's Manual*.

Message

The area below the line marked `Message` contains the data portion of a LAN packet.

If the message is for an NS Common Service or ARPA Service, the data portion contains (from left to right, top to bottom)

- the IP header
- the TCP header
- the socket level data buffer (may include the NFT header)

If the message is used for Socket Registry, the data portion contains (from left to right, top to bottom)

- the IP header
- the PXP header
- the Socket Registry header
- the Socket Registry data

If the message is used for Probe, the data portion contains (from left to right, top to bottom)

- the Probe header
- the Probe data

If the message is for a DS/1000-IV Compatible Service, the data portion contains (from left to right, top to bottom)

- the IP header
- the IFP header
- the data portion of the Router/1000 message
- the header portion of the Router/1000 message

Control Buffer

The area below the line marked `Control Buffer` is the Non-Router Control Buffer. This information is used to handle messages in the local node, but is meaningless to the remote node and is not transmitted. The Non-Router Control Buffer is described in the *NS Message Formats Reference Manual*.

Router/1000 Records

Router/1000 trace records have two types of borders: The first type consists of n's and is used for messages sent over Router/1000 links.

The second type consists of plus signs (+) and is used for DS/1000-IV Compatible Services messages sent over an 802 link. Outbound messages of this format are traced when they move from SAM (from #MAST or #SLAV) to DSAM (to IFP's socket). Inbound messages of this format are traced when they move from SAM (from GRPM) to the appropriate slave monitor or user. These records are written to the trace file in addition to the 802 (network level) trace records.

Router/1000 messages have the format shown in Figure 5-8.

```
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
n      time                                                NS/1000     n
n      originating node                                                n
n                    Message type = Network Level                       n
n                                direction                              n
n   Message length   = length           Link Interface =   LI type     n
n                                       Monitor/Service ID = service    n
n   Source Node      =  source             Destination Node = dest      n
n----------------------------Router Header----------------------------n
n   nnnnnn  nnnnnn  nnnnnn  nnnnnn  nnnnnn  nnnnnnn  nnnnnn  nnnnnn   n
n   xx xx   xx xx   xx xx   xx xx   xx xx    xx xx   xx xx   xx xx    n
n   nnnnnn  nnnnnn  nnnnnn  nnnnnn  nnnnnn  nnnnnnn  nnnnnn  nnnnnn   n
n   xx xx   xx xx   xx xx   xx xx   xx xx    xx xx   xx xx   xx xx    n
n-----------------------------Message-------------------------------n
n   nnnnnn  nnnnnn  nnnnnn  nnnnnn  nnnnnn  nnnnnnn  nnnnnn  nnnnnn   n
n   xx xx   xx xx   xx xx   xx xx   xx xx    xx xx   xx xx   xx xx    n
n-------------------------Control Buffer----------------------------n
n   nnnnnn  nnnnnn                                                    n
n   xx xx   xx xx                                                     n
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
```

Figure 5-8. Router/1000 Trace Record Format

Router/1000 Trace Record Fields

time The time that the NSTRC posted the trace record, using a 24-hour clock.

originating node The node at which the raw trace file was created.

direction The direction of the message (REQUEST or REPLY). This is not relative to the originating node. FMTRC prints request messages flush with the left margin and reply messages flush with the right margin.

length	The length of the data portion of the Router/1000 message, in words. FMTRC will round up this value if the message has an odd number of bytes.
LI type	The LI type over which the message was transmitted or received. For messages sent over a RTR LI, this field will be ROUTER. For messages sent over a LAN LI, this field will be NON-RTR.
service	The monitor or service that sent or received the message. For NS Common Services or ARPA Service, this field will be INPRO. The abbreviations are the same as those used when specifying monitors or services for FMTRC.
source	The Router/1000 address of the source node (the node sending the request or the node receiving the reply).
dest	The Router/1000 address of the destination node (the node receiving the request or sending the reply). For Dynamic Rerouting update messages, this is the negative value of the transmit LU.
nnnnn	The octal (or hexadecimal) value of a word in the Router/1000 header, data buffer, or control buffer. This information is printed from left to right, top to bottom, one word at a time, in octal (or hexadecimal). Below the octal (or hexadecimal) value of the word is the ASCII representation.
xx xx	The ASCII representation of a word in the Router/1000 header, data buffer, or control buffer.

Router/1000 Header

The area below the line marked Router Header contains the Router/1000 header. The Router/1000 header is described in the *NS Message Formats Reference Manual*.

Message

The area below the line marked Message contains the data portion of a Router/1000 message.

If the message is for an NS Common Service or ARPA Service, the data portion contains (from left to right, top to bottom)

- the IP header
- the TCP header
- the socket level data buffer (may include the NFT header)

If the message is used for Socket Registry, the data portion contains (from left to right, top to bottom)

- the IP header
- the PXP header
- the Socket Registry header

If the message is for a DS/1000-IV Compatible Service, the data portion contains (from left to right, top to bottom)

- the data portion of the DS/1000-IV Compatible Services message

Control Buffer

The area below the line marked `Control Buffer` contains information needed to handle the requests in the local node, but is meaningless to the remote node and is not transmitted. If the record contains a DS/1000-IV Compatible Services message, the control buffer is the DS/1000-IV local appendage. If the record contains an NS-ARPA message sent over a Router/1000 link, the control buffer is a Router/1000 Control Buffer. The DS/1000-IV local appendage, and Router/1000 Control Buffer are described in the *NS Message Formats Reference Manual*.

HDLC Link Up Messages

FMTRC also prints link up messages from the HDLC driver, which has the following format:

```
currentTime     Link up message from driver.  LU  nnn
```

These messages are recorded when an HDLC card establishes a logical link connection with its peer.

6

Nodal Registry List Utility, NRLIST

Overview

NRLIST lists the contents of the Nodal Registry which lists all the nodes you can communicate with and provides the mapping between a node name and a node's IP address. This information is helpful if you are having problems communicating with a node through its node name.

Runstring

```
[RU,]NRLIST
```

Help Menu

When you run NRLIST, it prints the following menu and prompts you for a command:

```
Please select one of the following functions:
D[,<filename>] = Dump Nodal Registry to file
R[,<filename>] = D + raw table info
?              = Print Help Menu
Enter /E to exit
NRLIST>
```

Enter `D` for Dump mode, `R` for Raw mode, `?` to display the help menu, or `/E` to exit. For Dump and Raw mode, you can also enter a file name preceded by a comma. If you do not specify a file, NRLIST will print the Nodal Registry contents to the scheduling terminal.

If you do enter a file name for Dump or Raw mode, NRLIST will print the contents of the Nodal Registry to that file; if the file already exists, NRLIST will print the following prompt:

```
file already exists. [(R)=replace / E=Exit / N=newfile]?
```

Enter `R` to replace the file, `E` to exit, or `N` to specify a new file name. If you enter `N`, NRLIST will prompt you for a new file name:

```
Enter file name :
```

NRLIST will then process your command, as described in the following subsections.

Output Files

You can use NRLIST output files as input files for NRINIT. NRLIST writes the raw NPRs (Nodal Path Reports) for each NPR in a form that you can use as input for NRINIT, the Nodal Registry initialization utility. NRLIST writes all other information with an asterisk (`*`) in the first column, and NRINIT ignores all lines with an asterisk in the first column.

Output Formats

A *raw NPR* is a text string that NRINIT can use to build an NPR for the Nodal Registry. For the syntax of raw NPRs, refer to the section "Nodal Registry Configuration" in *NS-ARPA/1000 Generation and Installation Reference Manual*.

A *raw internal record* shows the NPR as it is stored in the Nodal Registry, with the decimal value shown for each byte. This information is intended for HP internal use only.

Interactive Display

If NRLIST's *outputDevice* is a terminal, NRLIST will print one screen of data. If there is more data, NRLIST will ask if you want to display more data:

```
More...('a' to abort)?
```

Enter `a` to abort the display and terminate NRLIST; RETURN to display the rest of the output; enter any other key to display another screen of data, and NRLIST will re-prompt you. If you enter RETURN, and if CM is the secondary program on your system, you can strike any key, get the CM prompt and enter `BR,NRLIST`. This will set NRLIST's break bit and NRLIST will again ask if you want to display more data.

Banner

NRLIST prints the following banner each time the user specifies `D` or `R` for the operation mode.

```
* <revisionDate>
* Nodal Registry Dump File, Version 1.0
* File created: date time
* At node nodeName
```

revisionDate	software revision date
date time	current time and date
nodeName	name of the node at which you are running NRLIST

Dump Mode

In Dump mode, NRLIST lists the contents as raw NPRs. In addition, NRLIST prints the number of Nodal Path Reports in the Nodal Registry.

Example

```
CI> nrlist
Please select one of the following functions:
D[,<filename>] = Dump Nodal Registry to file
R[,<filename>] = D + raw table info
?              = Print Help Menu

Enter /E to exit
NRLIST> d
*<850910.1716>
* Nodal Registry Dump File, Version 1.0
* File created: Tue Sep 10, 1985   5:05 pm
* At node RICKY.IND.HP
BEGIN H.IND.HP 192.006.001.211  IP END
BEGIN A.HERE.HP 192.006.250.002  IP IEEE802 11-11-11.11-11-11 END
*Nodal Registry contains     2 entries.
NRLIST> /e
CI>
```

Raw Mode

In Raw mode, NRLIST lists the contents of the Nodal Registry as raw NPRs, and the raw internal NPR record. In addition, NRLIST prints the number of Nodal Path Reports in the Nodal Registry.

This mode is intended for HP internal use only.

7

DS/1000-IV (RTE-MPE) Message Tracing Utilities

Overview

LOG3K and TRC3K allow you to trace and format DS/1000-IV Compatible Service (RTE-MPE) message records over Bisync and X.25 links. QUEX and D3KMS write the message records as they are sent and received from the HP 3000. The messages are written to a magnetic tape device. LOG3K is an interactive program that allows you to enable, disable, and specify tracing options. TRC3K formats the tracing output and prints the output to a file or device.

You can use message tracing to debug user applications between two nodes and troubleshoot the network. A DS/1000-IV Compatible Service (RTE-MPE) message record has three parts:

- *Message header*, which contains control information, such as the message type, and source and destination node addresses.
- *Appendage*, which contains parameters specific to the message.
- *Data* sent (optional), such as lines of text.

LOG3K and TRC3K allow you to select which parts of the messages to trace and format. In addition, TRC3K allows you to select messages according to message class and stream type. Message classes and stream types differentiate messages according to function. For more information on message classes and stream types, refer to Table 7-3 at the end of this section.

Note You cannot use CS/80 magnetic tape devices for DS/1000-IV (RTE-MPE) message tracing.

LOG3K

LOG3K allows you to enable DS/1000-IV Compatible Service (RTE-MPE) message tracing, disable tracing, reset tracing, and specify the type of information that you want to trace.

Runstring

```
[RU,]LOG3K[,consoleLU]
```

Parameters

consoleLU	The logical unit that LOG3K uses for command input and output. *Default*: Local LU 1 (scheduling terminal).

LOG3K Operation

If the network manager did not specify DS/1000-IV Compatible Services (RTE-MPE) at network initialization time, LOG3K prints an error message and terminates.

Otherwise, LOG3K prints the following message:

```
DS/1000-3000 LOGGING STATUS
 LOG LU nn
```

nn is the current LU to which LOG3K is writing DS/1000-IV Compatible Service (RTE-MPE) message records, or <NONE> if tracing is not enabled.

In addition, if tracing is enabled, LOG3K prints a message that indicates the type of information it is tracing. For example:

```
LOGGING 50 WORDS DATA
```

The above message indicates that LOG3K is tracing the first 50 words of each data buffer.

LOG3K Command Summary

LOG3K prompts you for commands by printing the following message:

```
CHANGES?
```

LOG3K commands are summarized in Table 7-1, and discussed in the following subsections. An example appears in "LOG3K Example" later in this section.

Table 7-1. LOG3K Commands

Command	Function
??	Print a description of LOG3K commands and options.
EN EX /E NO	Exit LOG3K.
FI	Invalid command on RTE-A systems. Specify new log file.
LU	Enable tracing, disable tracing or specify LU of magnetic tape device to which trace records will be logged.
TY	Specify the type of tracing (header, appendage and/or data).
UP	Restart tracing if an error is encountered while writing to the magnetic tape.

Using a Magnetic Tape

The trace messages are written to non-CS/80 magnetic tape devices. Before enabling tracing, you must perform the following tasks:

- Check that the magnetic tape device is available.
- Mount a write-enabled tape to the device.

Since LOG3K does not write an End-of-File (EOF) mark to the tape or rewind the tape, you can enable and disable tracing several times with the same tape. If you do not rewind the tape, each period of tracing will be appended to the previous period.

After the last period of tracing, you must write an End-of-File (EOF) mark to the tape and rewind the tape. To do this, enter the following RTE CI command sequence:

```
CI>CN,magTapeLU,EOF       write an EOF to the tape
CI>CN,magTapeLU           rewind the tape
```

??

??

Print a description of LOG3K commands and options.

Syntax

```
??
```

Discussion

The `??` command prints the following description of LOG3K commands and options:

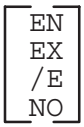
```
CHANGES? ??
LOG3K SETS UP FIVE WORDS IN SUBSYSTEM GLOBAL WHICH ARE USED BY THE HP3000
COMMUNICATIONS MODULE QUEX. THESE WORDS INDICATE WHAT TRACING SHOULD
TAKE PLACE FOR COMMUNICATIONS BUFFERS AND WHETHER DRIVER FUNCTIONS SHOULD
BE RECORDED. LOG3K REPORTS CURRENT OPTIONS AND ALLOWS YOU TO CHANGE THEM.

POSSIBLE CHANGES:
    LU    NEW LOG LOGICAL UNIT
    FI    NEW LOG FILE (SPOOL SYSTEM ONLY)
    UP    RESET LOG LU TO BE "UP"
    TY    NEW LOGGING TYPE

POSSIBLE LOGGING TYPES:
    NO    NOTHING
    HE    COMM. BUFFER HEADER ONLY
    AP    HEADER AND APPENDAGE
    DA:n  HEADER, APPENDAGE, AND n WORDS DATA
    DR    DRIVER EVENTS AND STATES (MAY BE SECOND OPTN)
EXAMPLE-- DA:50,DR  TRACES 50 WORDS DATA AND DRIVER.
```

Note The `FI` command and `DR` option are invalid on RTE-A systems.

EN, EX, /E, NO

Exit LOG3K when issued in response to the `CHANGES?` prompt.

Syntax



LU

Enable tracing, disable tracing, or specify LU of magnetic tape device to which LOG3K will log trace records.

Syntax

```
LU [,magTapeLU]
   [,0        ]
```

Parameters

magTapeLU	The LU of the magnetic tape device. Must not be a CS/80 magnetic tape device. QUEX/D3KMS will write the DS/1000-IV Compatible Service (RTE-MPE) message records to the tape mounted at that device. By specifying an LU, you enable tracing.
0	Disables tracing.

Discussion

If you do not specify *magTapeLU* or 0, LOG3K will print the following prompt:

```
NEW LOG LU:
```

Enter the *magTapeLU*, 0 or /E, to return to the CHANGES? prompt.

To turn off all tracing, set the log LU to 0.

After you change the magnetic tape device LU, LOG3K prints the new tracing options and returns to the CHANGES? prompt.

For more information on magnetic tape output, refer to the subsection “Using a Magnetic Tape” earlier in this section.

TY

Specify the type of information that you want to trace.

Syntax

```
      ┌          ┐
      │ AP       │
TY    │ DA:numWords │
      │ HE       │
      │ NO       │
      └          ┘
```

Parameters

`AP`	Trace message header and appendage.
`DA:` *`numWords`*	Trace message header, appendage, and *`numWords`* words of data. *`numWords`* must be an integer between 1 and 135. The total length of the message will not exceed 143 words.
`HE`	Trace message header.
`NO`	Trace nothing. QUEX/D3KMS will not write any records to the trace file, but the file will still be open and tracing will still be enabled.

Discussion

If you do not specify a type of tracing in the `TY` command, LOG3K will print the following prompt:

```
NEW LOG TYPE:
```

Enter `AP`, `DA:`*`numWords`*, `HE`, or `NO`, as described above or enter `/E` to return to the `CHANGES?` prompt.

After you change the tracing options, LOG3K prints the new tracing options and returns to the `CHANGES?` prompt.

The `DR` (driver events and states) option in DS/1000-IV is invalid in RTE-A systems and cannot be used with the `TY` command.

UP

Restart tracing if an error has occurred while writing to the magnetic tape.

Syntax

```
UP
```

Discussion

If QUEX or D3KMS gets an error while writing to the magnetic tape, then tracing is down.

You should then check the magnetic tape device and make sure that there is a write-enabled tape mounted. When the magnetic tape device is ready and you want to restart tracing, enter the LOG3K `UP` command.

LOG3K Example

In the following example LOG3K example, it is assumed that the user has checked the magnetic tape device, LU 7, and mounted a write-enabled tape.

```
CI>RU,LOG3K
DS/1000-3000  LOGGING STATUS
 LOG LU <NONE>

CHANGES?LU,7
DS/1000-3000  LOGGING STATUS
 LOG LU   7
 LOGGING NOTHING

CHANGES?TY,DA:50          enable tracing to LU 7
DS/1000-3000  LOGGING STATUS
 LOG LU    7
 LOGGING 50 WORDS DATA

CHANGES?EX                exit LOG3K
END LOG3K

CI>RU,PTOP3               run PTOP3, a user RTE-MPE PTOP program
     :
     :
CI>RU,LOG3K
DS/1000-3000  LOGGING STATUS
 LOG LU   7
 LOGGING 50 WORDS DATA

CHANGES?LU,0              disable tracing
DS/1000-3000  LOGGING STATUS
 LOG LU <NONE>

CHANGES?EX                exit LOG3K
END LOG3K

CI>CN,magTapeLU,EOF       write an EOF to the tape
CI>CN,magTapeLU,RW        rewind the tape
```

TRC3K

TRC3K formats and prints the message records traced by QUEX/D3KMS.

Runstring

```
[RU,]TRC3K[,commandInput][,loggingInput][,outputDevice]
```

Parameters

commandInput	The LU or file which provides the TRC3K commands. *Default*: Local LU 1 (scheduling terminal).
loggingInput	The LU or file containing the traced records. QUEX/D3KMS must not be writing to this LU while TRC3K is reading from it. If you enter a negative number, TRC3K will terminate. If not specified, TRC3K prompts you for the logging input LU or file name.
outputDevice	The LU or file to which TRC3K will print the formatted output. *Default*: *commandInput* if it is interactive, or local LU 1 (scheduling terminal).

TRC3K Operation

If you did not specify *loggingInput* in the TRC3K runstring, TRC3K prompts you for the LU or file containing the traced records:

```
LOGGING INPUT:
```

Enter the LU or file name.

TRC3K Command Summary

TRC3K prompts you for commands by printing the following message:

```
/TRC3K:
```

TRC3K commands are summarized in Table 7-2, and discussed in the following subsections. An example appears in "TRC3K Example" later in this section.

Table 7-2. TRC3K Commands

Command	Function
`??`	Print a description of TRC3K commands.
`EXIT`	Exit TRC3K.
`FORMAT`	Specify the items (header, appendage and/or data) to format.
`LIST`	Set the list (output) device or file.
`PRINT`	Print the formatted message records to the output device.
`SET`	Set the characteristics of the messages to be printed.

If the *`commandInput`* is not the same as *`outputDevice`*, TRC3K echoes commands at the *`outputDevice`*.

You can delimit TRC3K command parameters and options by entering a comma or one or more blank spaces. The syntax diagrams in the following subsections use blank spaces as delimiters. Command lines must be 80 characters or less.

To stop execution of a TRC3K command, use RTE's `BR` (break) command. TRC3K will ask `CONTINUE?` and wait for a reply. If you do not reply `Y` (yes), TRC3K will abort the command and prompt for a new command.

??

Print a description of the TRC3K commands.

Syntax

```
??
```

Discussion

The `??` command prints the following description of TRC3K commands.

```
/TRC3K:??

 COMMAND           DESCRIPTION

  ??             DISPLAY COMMANDS
  EXIT           END TRC3K
  FORMAT         SET OUTPUT LISTING FORMAT
  LIST           SET LISTING
  PRINT          PRINT SELECTED BUFFERS
  SET            LIMIT RECORDS TO BE PRINTED
```

This command is valid only when *`outputDevice`* is interactive.

EXIT

Exit TRC3K.

Syntax

```
E[XIT]
```

FORMAT

Specify which portion or portions of the message record to print when you enter the PRINT command.

Syntax

```
F[ORMAT] [ H[EADER]     ]
         [ A[PPENDAGE]  ]
         [ D[ATA]       ]
```

Parameters

HEADER	*Default*: Print the message header only.
APPENDAGE	Print the header and an octal and ASCII representation of the appendage.
DATA	Print the header, and an octal and ASCII representation of the appendage and data.

Discussion

The default FORMAT option is HEADER.

LIST

Set the list (output) device or file.

Syntax

```
L[IST]=outputDevice
```

Parameters

outputDevice The LU or file to which TRC3K will print the formatted output.

Default: the second parameter in the runstring (*outputDevice*); if you did not specify *outputDevice* in the runstring, the default is the scheduling terminal.

PRINT

Print the formatted records to the output device.

Syntax

```
          ┌─────────┐
P[RINT]   │A[LL]    │
          │F[IRST]  │
          │N[EXT]   │
          └─────────┘
```

Parameters

`ALL`	Print all the message records that meet the characteristics specified by the `SET` command.
`FIRST`	Print the first message that meets the characteristics specified by the `SET` command.
`NEXT`	*Default*: Print the next message that meets the characteristics specified by the `SET` command.

Discussion

`PRINT` prints the message records that meet the characteristics specified by the `SET` command. The portion of the message records (header, appendage, and/or data) printed is determined by the `FORMAT` command.

The default is `NEXT`.

SET

Set the characteristics of the messages to be printed by the PRINT command.

Syntax

```
S[ET] [ C[LASS] = value
        E[NDREC] = value
        R[TENO] = value
        STA[RTREC] = value
        STR[EAM] = value
        @ ]  . . .
```

Parameters

CLASS	Select only the message records that belong to the class specified by *value* (decimal). For a list of message classes and the corresponding messages, refer to Table 7-3 at the end of this section.
ENDREC	Select the records on the tape with record numbers less than or equal to the number specified by *value*. The record numbers are the tape's sequential record numbers. *Default*: the highest (last) record number in the file.
RTENO	Select only the message records generated by the RTE process number specified by *value*. The RTE process number is the user's terminal LU.
STARTREC	Select the records on the tape with record numbers greater than or equal to the number specified by *value*. The record numbers are the tape's sequential record numbers. *Default*: the first record number (number 1) in the file.
STREAM	Select only the message records that belong to the stream type specified by *value* (octal). For a list of message streams and the corresponding messages, refer to Table 7-3 at the end of this section.
value	Integer value from x to y, or @ to clear the value. For STREAM, this value is an octal number.
@	Clear all the values set.

Discussion

SET defines which message records the PRINT command will list. If no values are set, PRINT will list all the message records traced.

TRC3K Example

The following is a sample of the output from TRC3K. Make sure the magnetic tape is rewound.

```
CI>RU,TRC3K
LOGGING INPUT:8

                    ***** DS/1000-3000 TRACE *****

/TRC3K:PRINT FIRST

RECORD     1,  MESSAGE FROM 1000:
 CLASS  6  STREAM 20  FROM  66  TO   0  SEQ  66055  TOTAL LEN    22
    HELLO REQST
HEADER
  013006 000000 000020 000000 041000 066055 000000 000033*        B l-

/TRC3K:FORMAT DATA
/TRC3K:PRINT FIRST

RECORD     1,  MESSAGE FROM 1000:
 CLASS  6  STREAM 20  FROM  66  TO   0  SEQ  66055  TOTAL LEN    22
    HELLO          REQST
HEADER
  013006 000000 000020 000000 041000 066055 000000 000033*        B l
APPENDAGE LENGTH    14 WORDS
  044105 046114 047440 043111 042514 042056 051525 050120*HELLO FIELD.
  047522 052054 044120 031462 031060 034440              *SUPPORT,
                                                         *HP32209
DATA LENGTH      0 BYTES

/TRC3K:PRINT

RECORD     5,  MESSAGE FROM 1000:
 CLASS  5  STREAM 23  FROM  66  TO  30  SEQ  66057  TOTAL LEN    10
    TERMINAL CNTRL REPLY
HEADER
  005005 000000 100023 000000 041036 066057 000000 000004*        B l/
APPENDAGE LENGTH     2 WORDS
  001000 000000                                          *
DATA LENGTH      0 BYTES
```

In the following example, the user wants to print all messages in which a remote HELLO was sent. Remote HELLO messages are message class 6, stream type 20.

```
/TRC3K:SET STREAM = 20,CLASS=6,STARTREC=1

/TRC3K:PRINT ALL

RECORD     1,  MESSAGE FROM 1000:
 CLASS  6  STREAM 20  FROM  66  TO   0  SEQ  66055  TOTAL LEN    22
    HELLO          REQST
HEADER
  013006 000000 000020 000000 041000 066055 000000 000033*        B l-
APPENDAGE LENGTH    14 WORDS
  044105 046114 047440 043111 042514 042056 051525 050120*HELLO FIELD.
  047522 052054 044120 031462 031060 034440              *SUPPORT,
                                                         *HP32209
DATA LENGTH      0 BYTES

RECORD     8,  MESSAGE FROM 3000:
 CLASS  6  STREAM 20  FROM  30  TO  66  SEQ  66060  TOTAL LEN    8
    HELLO          REPLY
HEADER
  004006 000000 100020 000000 017102 066060 000000 000000*        Bl0
APPENDAGE LENGTH     0 WORDS
DATA LENGTH      0 BYTES
```

Next, the user tries to print all remote BYE messages. Remote BYE messages are message class 6, stream type 21.

```
/TRC3K:SET STREAM = 21      Reset the stream to 21 (class is already 6)
/TRC3K:SET STARTREC = 1     Start at record 1.
/TRC3K:PRINT
** NONE QUALIFIED **
/TRC3K:
```

There were no BYE messages.

```
/TRC3K:EXIT
```

Message Classes and Stream Types

Refer to the SET command and "TRC3K Example" to see how message classes and stream types are used and displayed.

Table 7-3. Message Classes and Stream Types

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

8

Modification Utility, DSMOD

Overview

DSMOD allows you to alter DS/1000-IV Compatible Services parameters set during initialization, or change timing parameters not set through NSINIT questions. Parameter modifications made with DSMOD remain in effect until the node is rebooted or DSMOD is used again to make further modifications.

DSMOD

DSMOD allows you to alter DS/1000-IV Compatible Services parameters set during initialization, or change timing parameters not set through NSINIT questions.

Runstring

```
[RU,]DSMOD[,lu ][,errordevice]
          [,file]
```

lu	The logical unit number of the device from which responses to the DSMOD prompts will be entered. Default is the scheduling terminal.
file	The name of a command file that contains responses to the DSMOD prompts. If the full path is not specified, the default directory is the current working directory.
errordevice	The logical unit number of the device where errors will be logged. Default is the scheduling terminal.

DSMOD Command Summary

When DSMOD is run with an interactive input device, you are prompted with the following question:

```
/DSMOD: OPERATION?
```

You can respond to this prompt with any of the commands listed in the following pages, which are summarized in Table 8-1.

If NS-ARPA is shutdown while DSMOD is running, DSMOD may abort with a request error (RN02).

Table 8-1. DSMOD Commands

Command	Description
??	Prints a list of the DSMOD commands.
/A	Aborts DSMOD.
CN	Changes the Nodal Routing Vector (NRV).
DI	Disables a link. You cannot use the DI command to disable LAN links.
/E	Exits DSMOD.
/I	Changes local and remote ID sequences.
/L	Enables a link. You cannot use the /L command to enable LAN links or X.25 LUs. You can use the DSMOD DI command to return X.25 LUs to the pool.
/N	Displays the Nodal Routing Vector (NRV).
/S	Schedules monitors. (Only the following monitors can be scheduled with DSMOD: CNSLM, DLIST, EXECM, EXECW, OPERM, PTOPM, PROGL, RFAM, RDBAM, VCPMN)
/T	Adjusts network timing values that are used by the DS/1000-IV Compatible Services.

Command Files

When DSMOD is scheduled with a command file (*file* parameter), it obtains the responses to its prompts from the file. Running DSMOD with a command file can be useful if there are changes that need to made on boot-up.

The following DSMOD command file example changes certain timing parameters with the /T command. (This command is explained in detail later in this section.)

```
**REQUEST TIMING CHANGES
/T
**GIVE THE NETWORK MANAGER'S SECURITY CODE
NM
**CHANGE THE MASTER TIMEOUT
20
**LEAVE THE SLAVE TIMEOUT UNCHANGED
22
**LEAVE THE REMOTE BUSY RETRIES UNCHANGED
3
**LEAVE THE REMOTE QUIET RETRIES UNCHANGED
0
**CHANGE THE MAXIMUM HOP COUNT
7
**LEAVE THE MAXIMUM LINE DOWN COUNT UNCHANGED
10
**END DSMOD
/E
```

??—Command List

??—Command List

DSMOD prints a list of the DSMOD commands if you type two consecutive question marks (??) in response to the DSMOD: OPERATION? prompt. (DSMOD will also print a list of commands whenever an invalid command is entered.)

The following is an example of the command list that is printed by DSMOD:

```
/DSMOD: OPERATION? ??

?? : LIST COMMANDS
/A : ABORT
/E : TERMINATE
/I : CHANGE
/L : RE-ENABLE LINE
/N : DISPLAY NRV
/S : SCHEDULE MONITOR(S)
/T : ADJUST TIMING
CN : CHANGE NRV
DI : DISABLE LINE
```

/A

Aborts DSMOD.

Syntax

```
/A
```

Discussion

The `/A` command causes DSMOD to abort. If modifications were made prior to issuing the `/A` command, they can still be in effect. Once aborted, DSMOD prints the following message:

```
/DSMOD: DSMOD ABORTED
```

CN

Changes the Nodal Routing Vector.

Syntax

```
CN
```

Discussion

The CN command allows you to change the IP address, LU, nodal master timeout, upgrade level, and neighbor flag entries for a particular node in your Nodal Routing Vector. Note that if you change an IP address, DSMOD changes the address only in the NRV; it does not change the addresses in any IP tables.

If you specify a non-zero value for the nodal master timeout, this value overrides the network master timeout displayed by the DSMOD /T command. This timeout is used for all messages directed to the node.

Note Rerouting can change the transmit LU and neighbor status as links are restored.

When the CN command is issued, DSMOD prompts you for the network management security code and the Router/1000 node address of the node whose NRV entry you want to change.

In the following example, the network management security code is DS and the node address is 1.

```
/DSMOD: OPERATION? CN
/DSMOD: NETWORK MANAGEMENT SECURITY CODE? DS
/DSMOD: NODE # TO CHANGE? 1
```

DSMOD then displays the current NRV values for the specified node:

```
NODE= 1   IP=192.006.001.001  LU= 64   TO(SEC.)=   15 LEVEL= 1
```

Next, DSMOD prompts you for new NRV values. In the following example, the IP address is changed to 192.006.250.001, the transmit LU is changed to 60, the nodal master timeout to 20 seconds, and the neighbor status to N for neighbor. (You can skip parameters that you do not want to change by entering commas as place holders.)

```
/DSMOD: IP,LU,TIMEOUT,UPGRADE LEVEL [,N]? 192.006.250.001,60,20,1,N
```

DSMOD then displays the new NRV values:

```
NODE= 1   IP=192.006.250.001  LU= 60   TO(SEC.)= 20  LEVEL= 1,(N)
```

DSMOD continues to prompt for node addresses until it is terminated with the /E command.

```
/DSMOD: NODE # TO CHANGE? /E
/DSMOD: OPERATION?
```

DI

Disables a link.

Syntax

```
DI
```

Discussion

The `DI` command can be used to disable X.25 pool LUs, Router/1000 links (HDLC), and Bisync links to the HP 3000.

Note You can *not* use the `DI` command to disable LAN links.

/E

Exits DSMOD.

Syntax

```
/E
```

Discussion

When used in response to a multi-question command query, the `/E` command causes DSMOD to return to the `/DSMOD: OPERATION?` prompt.

When used in response to the `/DSMOD: OPERATION?` prompt, the `/E` command causes DSMOD to print the following message and then terminate:

```
END DSMOD /DSMOD: OPERATION? __
```

/I

Changes local and remote Bisync ID sequences.

Syntax

```
/I
```

Discussion

The /I command allows you to specify new local and remote Bisync ID sequences. Bisync ID sequences are exchanged and compared during the negotiations that takes place between HP 1000 and HP 3000 nodes when a Bisync dial-up link is established. A successful comparison completes the line bid sequence and the link is enabled.

The Bisync line bidding procedure provides a level of network security since callers at each end of the connection must know the ID sequence used by the other end. Because modem connections allow you to dial-up several different HP 3000s, the security function of line bidding is especially desirable for this link type. When a unique ID sequence is used at each HP 3000, you can ensure that no accidental connection to the wrong HP 3000 will occur.

Note To ensure that the firmware on the Bisync board receives the new ID sequence or sequences after they are modified, you can issue the /L command to re-enable the link. Following a disconnect or line re-enable, the local and remote ID sequences are passed to the firmware on the Bisync board. (The /L command is described later in this section.)

In the following example, the local ID sequence is set to HELLO HP 3000 and the remote ID sequence is set to HELLO HP 1000. Note that DSMOD prompts you for these values after the /I command is issued. You can enter from 1 to 15 characters in response to the local and remote ID sequence prompts. Entering /D provides no ID sequence.

```
/DSMOD: OPERATION? /I

/DSMOD: LOCAL ID SEQUENCE? HELLO HP 3000

/DSMOD: REMOTE ID SEQUENCE? HELLO HP 1000
```

/L

Enables a link.

Syntax

```
/L
```

Discussion

The /L command allows you to enable a Router/1000 (HDLC and Multidrop) and Bisync links that have not been previously enabled, recently connected, or recently repaired. You can modify the RTE power-fail restart utility AUTOR to schedule DSMOD with a command file which enables the DS LUs using the /L command. Refer to the appropriate RTE-A reference manual for more information on this utility.

Note You can *not* use the /L command to enable LAN links, HDLC links with Gateway link interfaces, or X.25 LUs. (You can use the DSMOD DI command to return X.25 LUs to the pool.)

When a link is re-enabled with the /L command, any ongoing activity is terminated.

After the /L command is issued, DSMOD prompts you for the logical unit number to be enabled. In the following example, /L is used to enable LU 14 and the /E command is used to return to the /DSMOD: OPERATION: prompt.

```
/DSMOD: OPERATION? /L

/DSMOD: ENABLE LU# ? 14

/DSMOD: ENABLE LU# ? /E

/DSMOD: OPERATION?
```

/N

Displays the Nodal Routing Vector.

Syntax

```
/N
```

Discussion

The /N command causes the Nodal Routing Vector to be displayed. This command can be used to determine whether or not the NRV at each node correctly matches your network connections.

Note The NRV changes made by rerouting are reflected at the time the /N command is issued.

The following is an example of NRV display.

```
/DSMOD: OPERATION? /N

NRV SPECIFICATIONS:
 LOCAL NODE#:   15  NO. OF NODES = 13
 NODE= 1     IP=000.000.000.000   LU=70       TO(SEC.)= 0   LEVEL= 1
 NODE= 2     IP=000.000.000.000   LU=70       TO(SEC.)= 0   LEVEL= 1
 NODE= 3     IP=000.000.000.000   LU=70       TO(SEC.)= 0   LEVEL= 1
 NODE= 4     IP=000.000.000.000   LU=70       TO(SEC.)= 0   LEVEL= 1
 NODE= 5     IP=192.006.001.005   LU=70,(NR)  TO(SEC.)= 0   LEVEL= 1
 NODE= 6     IP=000.000.000.000   LU=70       TO(SEC.)= 0   LEVEL= 1
 NODE= 7     IP=000.000.000.000   LU=70       TO(SEC.)= 0   LEVEL= 1
 NODE= 8     IP=000.000.000.000   LU=70       TO(SEC.)= 0   LEVEL= 1
 NODE= 9     IP=000.000.000.000   LU=70       TO(SEC.)= 0   LEVEL= 1
 NODE= 10    IP=000.000.000.000   LU=70       TO(SEC.)= 0   LEVEL= 1
 NODE= 11    IP=000.000.000.000   LU=70       TO(SEC.)= 0   LEVEL= 1
 NODE= 12    IP=192.006.001.012   LU=70,(NR)  TO(SEC.)= 0   LEVEL= 1
 NODE= 13    IP=192.006.001.013   LU=70       TO(SEC.)= 0   LEVEL= 1,(N)
```

This display indicates that the local node number is 15 and that it presently uses LU 0 to communicate with nodes 1 through 13. The (N) next to the level number for node 13 indicates that node 13 is node 15's neighbor. The level number refers to the software upgrade level. Level 1 indicates that the node has DS/1000-IV or NS-ARPA/1000 software; level 0 indicates that DS/1000 (91740A) is installed.

Node 12 has an IP address of 192.006.001.012. The (NR) next to the LU number for node 12 indicates that the first hop from this node is *not* a Router/1000 link.

/N

Entries for non-NS-ARPA/1000 nodes (DS/1000-IV and DS/1000 nodes) that are not members of the local node's Router/1000 Directly Connected Network (DCN) will have the guardian node's IP address in the IP address field.

Entries for non-NS-ARPA/1000 nodes that are members of the local node's Router/1000 DCN will have IP addresses that consists only of zeroes. For example, nodes 1 through 4 and 6 through 11 are DS/1000-IV nodes that are members of the local node's Router/1000 DCN (they do not have IP addresses and are all level 1).

/S

Schedules monitors.

Syntax

```
/S
```

Discussion

The /S command schedules DS/1000-IV Compatible Services monitors not previously scheduled by NSINIT. When the /S command is issued, DSMOD prompts you for monitor names until the entry list is terminated with the /E command. If you attempt to schedule a monitor that is already active, DSMOD returns DSMOD: ERROR: STAT: *monitor name* and then prompts for another monitor name.

Note Only the following monitors can be scheduled with DSMOD: CNSLM, DLIST, EXECM, EXECW, OPERM, PTOPM, PROGL, RFAM, RDBAM, VCPMN.

To schedule NFTMN, enter the CI command XQ,NFTMN.

In the following example, the monitor PTOPM is scheduled with the /S command.

```
/DSMOD: OPERATION? /S

/DSMOD: MONITOR NAME? PTOPM

/DSMOD: MONITOR NAME? /E

/DSMOD: OPERATION?
```

/T

Adjusts network timing values that are used by the DS/1000-IV Compatible Services and Transport.

Syntax

```
/T
```

Discussion

The /T command allows you to display and modify certain network timing values. The following values can be changed with the /T command and are explained later in "Network Timeouts":

- the network timing values for master and slave requests
- the remote busy retry count
- the remote quiet wait suspension period
- the maximum hop and maximum line down counts
- the idle session timeout.

Note Network timing values for master and slave requests and the idle session time-out can also be altered by running NSINIT.

/T Example

When the /T command is issued, DSMOD displays the current timeout values and prompts for the network management security code and for new timeout values. If you do not want to change selected values, you can respond with a space and carriage return.

The following example shows the timeout values that are displayed when the /T command is issued:

```
/DSMOD: OPERATION? /T

TIMING MODIFICATION--CURRENT VALUES:

MASTER T/O=45
SLAVE T/O=30
REMOTE-BUSY=3
```

```
REMOTE-QUIET=0
MAX. HOP COUNT=12
MAX LINE DWN CNT IN 5 MIN=10
IDLE SESSION T/O = 5
```

After these values are displayed, DSMOD prompts you for the network management security code (in this example, ZZ):

```
/DSMOD: NETWORK MANAGEMENT SECURITY CODE? ZZ
```

Next, DSMOD prompts you for the new timeout values. In the following example, the master timeout is set to 25, the slave timeout to 35, the number of remote busy retries to 5, the remote-quiet wait timeout to 0 seconds, the maximum hopcount to 5, the maximum line down count to 20, and the idle session timeout to 5 hours.

```
/DSMOD: MASTER T/O [5 TO 1275 SECONDS] ? 25

/DSMOD: SLAVE T/O [5 TO 1275 SECONDS] ? 35

/DSMOD: REMOTE-BUSY RETRIES [1 TO 10] ? 5

/DSMOD: REMOTE-QUIET WAIT [0 TO 7200 SEC]? 0

/DSMOD: MAXIMUM HOPCOUNT [1 TO 32767]? 5

/DSMOD: MAX LINE DOWN COUNT IN 5 MIN [2 to 32767]? 20

/DSMOD: IDLE SESSION TIMEOUT [0 TO 45 HRS]? 5
```

Network Timeouts

This subsection explains the network timeout values that you can alter.

A *master timeout* is the number of seconds that can elapse before your master request is timed out by UPLIN, the system transaction monitor. If a master request timeout occurs, system resources allocated for the transaction are returned to the system and the request timeout error DS05(0) or DS05(1) is reported. When message accounting is active in the two nodes, information on whether it was the request or reply that timed out is returned in the error qualifier. The default master timeout is 45 seconds.

You can adjust the *network master timeout* value with the /T command. The network master timeout is used with messages that originate from HP 1000 nodes that have a *nodal master timeout* of zero, and HP 3000 nodes that are connected via Bisync links. The nodal master timeout appears in the Nodal Routing Vector and is set during NS-ARPA initialization. If a value *greater* than zero was assigned to the nodal master timeout during initialization, that value will override the network master timeout. In order to adjust the nodal master timeout, you must use the DSMOD CN command. You cannot use the DSMOD /T command.

A *slave timeout* is the number of seconds that can elapse before an incoming slave request is timed out by UPLIN. If a slave request timeout occurs, system resources allocated for the transaction are returned to the system. The default slave timeout is 30 seconds.

A *remote busy retry* is the number of retries that will be performed to transmit a message during a "remote node is busy" condition. The transmission will be retried at one second intervals from 1 to 10 times, depending on the remote busy retries count. The default is three retries.

A *remote quiet wait* is the number of seconds that master programs are suspended before a master request is resubmitted to a remote system that has returned a node quiescent reply to the initiating node. The initiating node will continue to alternately suspend the master program for the remote quiet wait timing value and then submit the master request until the remote system accepts it. If a remote quiet wait timing value of zero is specified, the master program is not suspended if all the remote busy retries fail, and the error `DS08(0)`, remote busy, is returned.

The default remote quiet wait timing value is zero. HP recommends that you specify zero until you have more experience using your network. If you specify a non-zero value here and error conditions that result in an `DS08(0)` error message are encountered, the master program will attempt to retry the master request indefinitely until the error condition is resolved. Certain system generation errors can result in `DS08` errors that can never be resolved.

The *maximum hop count* is the number of store-and-forward operations that can occur on a Router/1000 message before it is determined to be in an infinite loop. The maximum hop counter is provided to protect against temporary message loop situations which can occur if there are several simultaneous topology changes in the network. It also protects against the change of NRVs being set up incorrectly or a user changing the NRV manually using DSMOD and specifying it incorrectly. Each node processing the message decrements a counter in the message header. When the count reaches zero, the message is flushed. The maximum hop count defaults to the number of nodes in the network.

The *maximum line down count* is the number of times an irrecoverable failure can occur on a line in a five minute period, before the line is declared too unstable for use and removed as a possible message route in the NRV. This parameter has no effect if rerouting is not generated in. The default maximum line down count is 10.

The *idle session timeout* is the length of time a remotely established session will be allowed to remain if it is not actively being accessed. A value of zero implies no time limit. The default is five hours.

9

Principles of Operation

Overview

This section describes the principles of operation and general flow of the following components of NS-ARPA/1000:

- NS Common Services software (excluding DS/1000-IV Compatible Services and ARPA Services software)
- DS/1000-IV Compatible Services software (including information on the interface between NS-ARPA/1000 software and DS/1000-IV Compatible Services software)
- Memory Manager
- Network File Transfer

NS-ARPA/1000 Path Flow

This subsection provides a conceptual overview of the internal organization of the NS-ARPA/1000 software, excluding the DS/1000-IV Compatible Services software. DS/1000-IV Compatible Services software is described in detail in "DS/1000-IV Compatible Services Internals" later in this section.

Before reading this subsection, you should understand the concept of layered network architectures, and the layers of the International Standards Organization's Model of Open Systems Interconnection (OSI).

NS-ARPA Protocol Modules

The majority of NS-ARPA software is contained in two processes: INPRO and OUTPRO. INPRO consists of the modules that perform the functions required for inbound message handling, and OUTPRO consists of the modules required for outbound message handling.

To understand the NS-ARPA architecture, it is helpful to group the modules according to tasks, similar to the way that tasks are divided in the OSI model. These groups are often called *protocol modules*. For example, the TCP protocol module refers to the software routines that, together, implement the Transmission Control Protocol. Each protocol module carries out a distinct

communication task. The NetIPC protocol module contains the software that allows NetIPC users to access the services provided by the other protocol modules.

The NS-ARPA protocol modules and the tasks they perform are

- *NetIPC*. NetIPC provides an interface for user access to network protocols. NetIPC allows two processes to exchange data via communication endpoints called sockets.

- *BSD IPC*. BSD IPC also provides an interface for user access to network protocols, using standards of the Berkeley Software Distribution Interprocess (BSD IPC) 4.3.

- *TCP (Transmission Control Protocol)*. TCP is a transport (OSI layer 4) protocol that provides non-duplicated, in-sequence data delivery to the user (NetIPC). TCP is a stream-based (rather than message-based) protocol. TCP accepts arbitrarily long data buffers (byte streams), segments them into packets, and sends each packet separately. TCP keeps track of the bytes sent and retransmits them if they are not acknowledged within a timeout interval. TCP at the receiving node reassembles the packets, so that they are delivered to a NetIPC socket in order (in-sequence delivery).

  TCP is a connection-based, end-to-end protocol. TCP permits users to establish full-duplex communication channels. When TCP modules at two nodes want to communicate, they establish endpoints for a communication channel, called ports. The TCP modules then establish a communication channel between the two ports, called a connection, or Virtual Circuit (VC). Each port is bound to a NetIPC VC socket in a one-to-one relationship.

  TCP is an end-to-end protocol; all store-and-forward functions are performed by lower-layer protocols. Only the TCP modules at the endpoints of a TCP connection handle data for the connection.

  TCP also provides flow control; the amount of data sent is controlled so that the sender does not overload the receiver.

- *Packet Exchange Protocol (PXP)*. PXP is a low-overhead request/reply datagram transport (OSI layer 4) protocol that is suited for querying data sources.

  As with TCP, PXP communication is conducted between ports. A PXP user can send a request via a PXP request port to a PXP service port. A reply would be sent from the service port back to the request port. PXP keeps track of requests with replies outstanding; if PXP does not receive a reply within a timeout interval, it will retransmit the request. Once PXP receives a reply for a request, it will discard any subsequent responses for that request.

  PXP is also an end-to-end protocol.

- *Internetwork Protocol (IP)*. IP (OSI layer 3i) is primarily used to route messages between networks via gateways. It provides gateway-to-gateway routing, store-and-forward service between gateways, and message fragmentation and reassembly between source and destination networks.

- *Link Interfaces (LIs)*. There are three LIs: Router/1000 (RTR), IEEE 802.3 Local Area Network (802), and Ethernet LAN (ETHER). The LIs serve as interfaces between IP and the data link layer drivers and mask some of the differences of the data links from IP. The LIs map IP addresses to transmit link LUs and subnet addresses. In addition, each of the LIs provide some routing services. A LAN LI can be both 802.3 and Ethernet.

The RTR LI implements the *Router/1000* protocol (OSI layer 3s), which provides subnet routing with store-and-forward and Dynamic Rerouting.

The 802 LI implements some of the IEEE 802.3 addressing (OSI layer 3s) services.

The Ethernet LI implements equivalent functionality to the 802.3 addressing services.

- *Interface Protocol (IFP)*. IFP provides the interface between NS Common Services/Transport and DS/1000-IV Compatible Services/Transport.

Figure 9-1 illustrates the protocol modules contained in INPRO and OUTPRO.

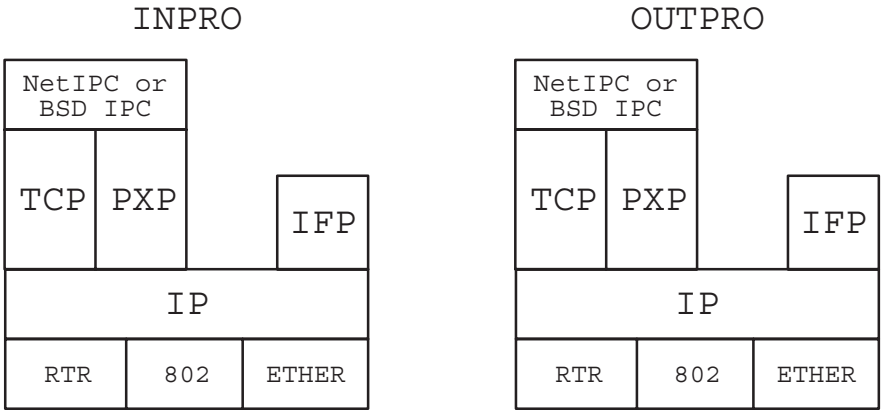


Figure 9-1. INPRO and OUTPRO Protocol Modules

Logical Data Flow

This subsection describes the conceptual flow of data as it travels through the NS-ARPA subsystem. The physical movement of data through the NS-ARPA subsystem is described in a later subsection.

When a user sends data to a remote user, the data passes down through an *NS-ARPA protocol stack* on the source machine. A protocol stack is a sequence of protocols. With NS-ARPA/1000, a given protocol stack would consist of NetIPC or BSD IPC at the top layer; at the next layer would be TCP, PXP, or IFP; IP at the next layer; and at the bottom layer, the RTR, 802, or Ethernet LI.

The logical course that a message takes within a machine through a protocol stack is called a *path*.

When a user sends data via NetIPC or BSD IPC, the data travels down through a protocol stack and each protocol module involved (except NetIPC) adds a protocol header to the data before passing it to the lower-layer protocol.

When the data is received at the destination node, the data travels up through a protocol stack. Each protocol handler involved strips its protocol's protocol header from the data before passing it to the upper-layer protocol. This manual will refer to user data, with protocol headers as they are added or stripped, as a *message*.

Figure 9-2 is a conceptual illustration of the path that a message takes through a source node and destination node.

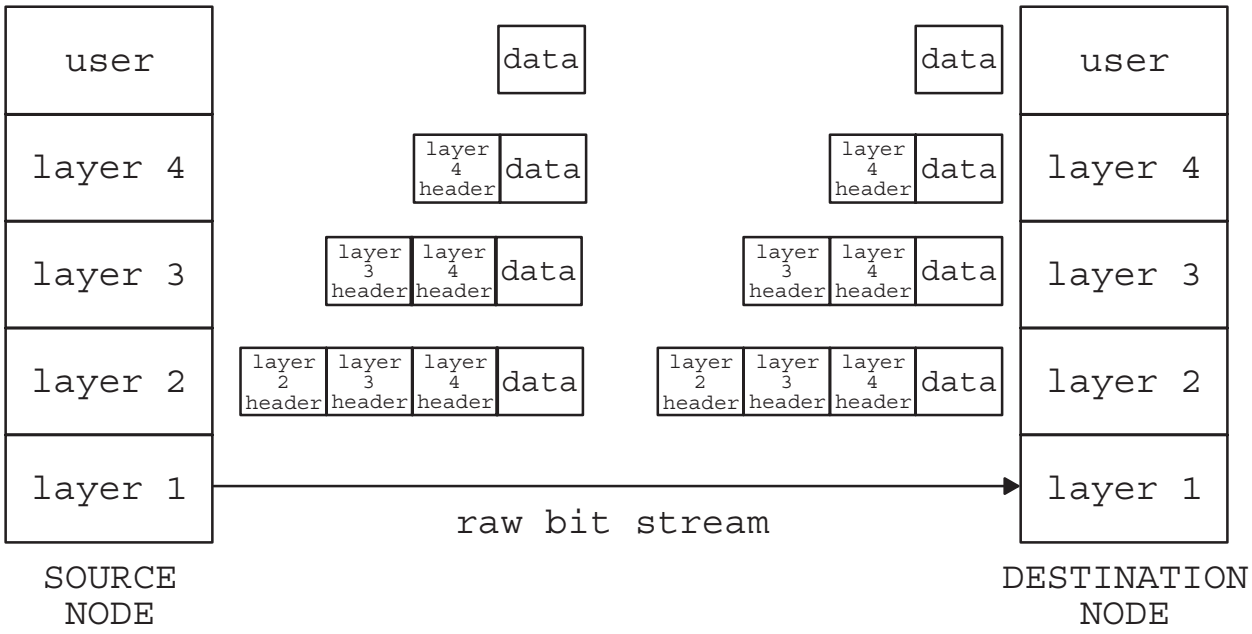


Figure 9-2. Conceptual Message Flow

Path Records

As noted above, the protocol modules add protocol headers as a message moves down the protocol stack. These headers contain addresses used by the protocols to deliver the message to the proper destination. For example, the TCP header contains the source and destination TCP port IDs. When a protocol passes a message to a lower-layer protocol it must also pass some information to help the lower-layer protocol in its addressing. NS-ARPA/1000 protocols keep address and other context information in data structures called *path records*. Therefore, when passing a message down, a protocol will also pass a reference to the appropriate lower-layer path record that the lower-layer protocol is to use to get address information. The protocols use *event messages* to pass this reference information.

The protocol modules also use the path records to guide messages to the next appropriate protocol.

To guide messages, path records contain the following information:

- the identification of the layer immediately below the current layer (the lower-layer protocol ID)
- the reference to the path record to use within the lower-level protocol (the lower-layer path reference)

A path record may also contain

- the identification of the layer immediately above the current layer (the upper-layer protocol ID)
- the reference to the path record to use within the upper-level protocol (the upper-layer path reference)
- context and control information that is protocol specific.

The term *path record* is used as general term for these logical records; the actual data structures that hold path record information are given different names by the different protocols, and usually contain other, protocol-specific information.

Table 9-1 lists the path records for each layer and the allocation conditions.

Table 9-1. Path Record Allocation

Layer	Data Structure used for Path Record	Address Information	Allocation
NetIPC	Socket Record	none	One per socket
BSD IPC	Socket Record	none	One per socket
TCP	D-record and Protocol Control Block (PCB)	Source TCP port; destination TCP port; source IP address; and destination IP address	One per TCP port used for VC socket (defined by unique triplet of: source TCP port ID, destination TCP port ID, and destination IP address). The PCB contains protocol-specific information, such as parameters used for timing, sending, and retransmission.
	I-record and Protocol Control Block (PCB)	Source TCP port; destination TCP port (null); source IP address; and destination IP address (null).	One per TCP port used for NetIPC call socket (PCB is placeholder only; not used).
PXP	I-record	Source PXP port; destination PXP port; source IP address; and destination IP address.	One per service port and per reply.
	D-record	Source PXP port; destination PXP port; source IP address; and destination IP address.	One per request port.
IP	IP path record	Source IP address and destination IP address.	One per unique triplet of: source node IP address, destination node IP address, and upper-layer protocol.
	Appropriate Next Hop (ANH) record	IP address of next node to route to (next hop).	One per next hop IP address.
RTR LI	Nodal Routing Vector (NRV) entry	IP address, Router/1000 address, and transmit link LU.	One per Router/1000 address (built at NS-ARPA initialization time).
802 LI	LAN Routing Table entry	IP address, IEEE 802.3 station address, and transmit link LU.	One per known LAN station address (built at NS-ARPA initialization time and dynamically added via Probe, Socket Registry, and inbound messages.
ETHERNET LI	LAN Routing Table entry	IP address, Ethernet station address, and transmit link LU.	One per known LAN station address (built at NS-ARPA initializtion time and dynamically added via Probe, Socket Registry, and inbound messages).

Figure 9-3 is an example of how protocols use the the address information in the path records to build message headers. For an Ethernet LI, the destination station address is an Ethernet LAN station address.

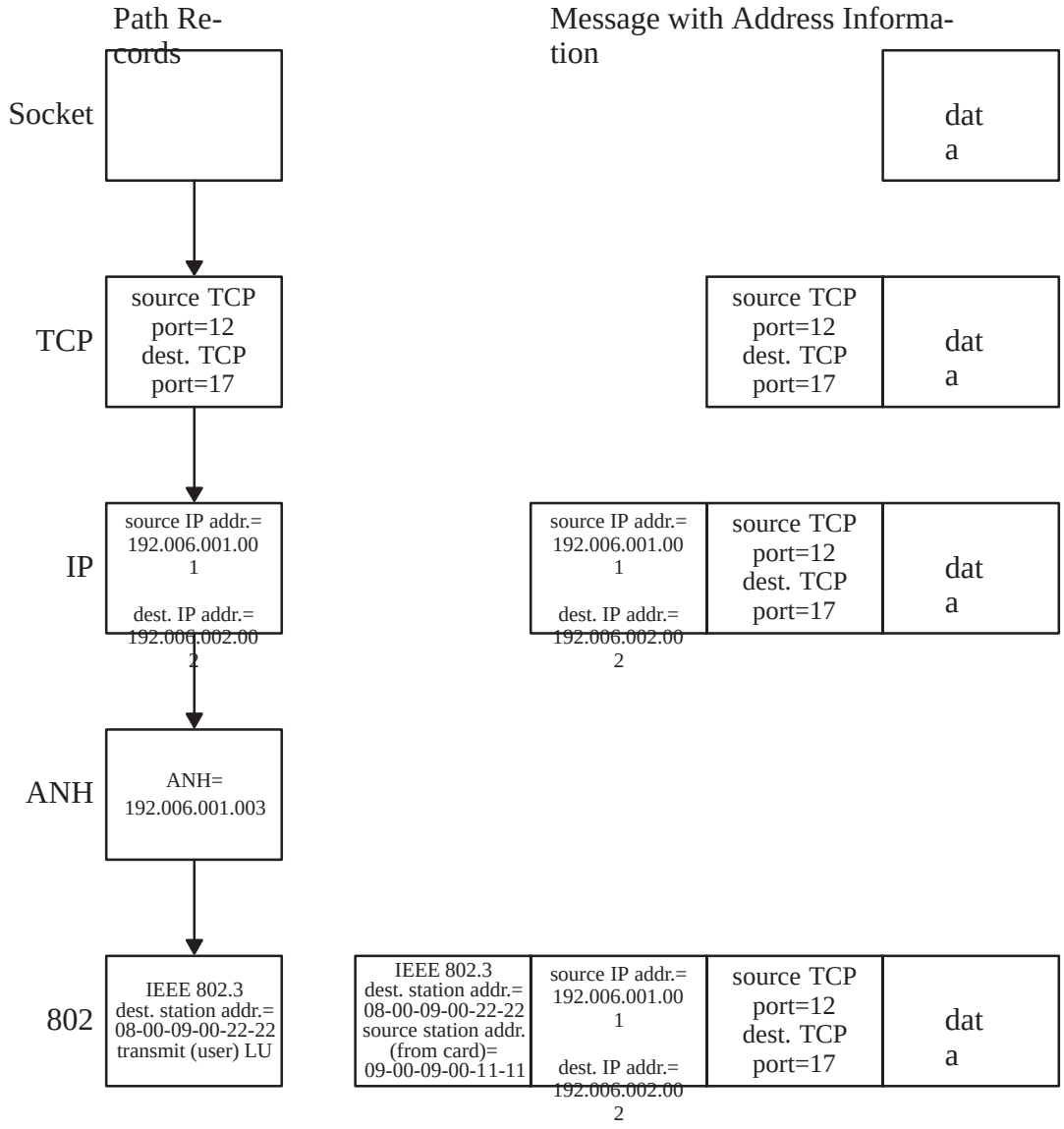


Figure 9-3. Example of Address Information in Path Records and Message Headers

Note that because of the allocation conditions for IP and LI path records, IP and LI path records can be shared by several upper-layer protocol path records. For example, different TCP connections between the same IP source and destination nodes would share an IP path record.

Paths

The path records for a given message are linked together by the references to the lower-layer path records, as illustrated in Figure 9-4. The path that a message takes through a machine can be defined by traversing the linked list of path records for the message. For example, messages sent from a user via a TCP VC socket over a Router/1000 link would have a path defined by the following types of path records: a NetIPC socket record, TCP D-record and PCB, IP path record, IP ANH record, and RTR NRV entry.

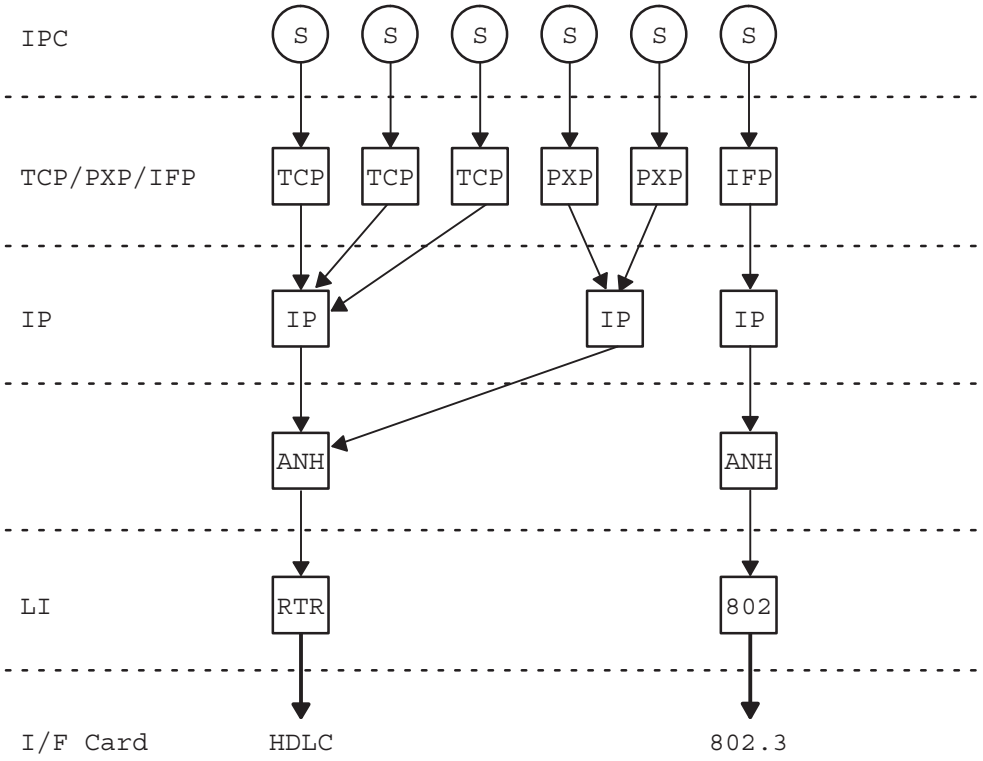


Figure 9-4. Example of Sockets, Path Records and Paths in a Node

In Figure 9-4, the sockets and socket records are indicated by the circled “S” characters, and the protocol path records are indicated by squares. The light arrows indicate the links between the path records and the dark arrows indicate the message being transmitted.

Paths are built when inbound messages are received and move up through the protocol stack. When a protocol passes a message up, the protocol will also pass a reference to the lower-layer path record used to receive the message.

Paths are built for outbound messages by emulating an inbound message in the Socket Registry software.

Nodal Path Reports and Connect Site Path Reports

For two NetIPC processes to communicate, each process must know

- how to reach the remote node
- how to reach the socket within the remote node

The above information is stored in *Nodal Path Reports (NPRs)* and *Connect Site Reports (CSRs)*. A Nodal Path Report (NPR) only provides information on how to reach a specific, named node. Nodal Path Reports are indexed according to node name, and each contain an IP address (or addresses, if the node belongs to multiple networks). If the node has LAN links, the NPR can optionally contain a LAN station address for each such link. In addition, an NPR denotes which protocols and NS Common Services the node supports.

NPRs are stored in *Nodal Registeries*.

A Connect Site Report (CSR) provides information on how to reach a given node within the internet and how to reach a given NetIPC socket *within* that node.

CSRs are stored in *Socket Registeries*.

VC Connection Establishment

Assume there are two processes, Process A and Process B, at Node A and Node B, respectively. The steps required to establish a VC connection between the two processes are as follows:

1. Process A calls `IPCCreate`, which creates a TCP call socket and returns a call socket descriptor. Process B also calls `IPCCreate`.
2. Process B calls `IPCName`, which names the call socket and makes an entry in Node B's Socket Registry.
3. Process A calls `IPCLookUp`, which looks up process B's call socket and returns a path report descriptor, which is a reference to the CSR to Node B and Process B's call socket.
4. Process A calls `IPCConnect`, using its call socket descriptor and the path report descriptor for Process B's call socket. This sets up Process A's half of the VC connection.
5. Process B calls `IPCRecvCn`. This sets up Process B's half of the VC connection and completes the VC connection.

The following subsections describe the above steps in fuller detail.

IPCCreate. If there is no NetIPC user record for the calling process, NetIPC creates a user record and root socket. NetIPC uses the user record to keep track of the process's NetIPC sockets and other resources used for NetIPC. NetIPC uses the root socket to communicate with lower-layer protocols and to set up paths for any call or VC sockets requested by the user.

At NetIPC's request, TCP creates a TCP port for the call socket with a TCP i-rec and PCB. The i-rec's destination TCP port ID and destination IP address fields are null, so that the TCP port can receive connection requests from any TCP port at any node. The TCP PCB is null.

NetIPC creates the call socket and socket record.

IPCName. NetIPC makes an entry in the local node's Socket Registry for the socket name. The entry contains a name record with a reference to the socket.

IPCLookUp. Process A calls `IPCLookUp` with Node B's node name and Process B's call socket's name.

Process A attempts to resolve Node B's node name to an NPR. To do this, Node A's Socket Registry first tries to match Node B's node name with an entry in Node A's Nodal Registry. If this fails and Node A is connected to a LAN, Socket Registry asks Probe to multicast a name request. (For more information on Probe operation, refer to the "Probe" subsection, below.)

Once Node A's Socket Registry has an NPR for Node B, it uses the NPR to build a path to use to send a query to Node B's Socket Registry.

The path consists of: a PXP path record for a PXP request port (PXP d-rec); an IP path record; an IP ANH record; and a LI path record.

The Socket Registry builds the path as follows:

The LI path record (mainly, the station address) is built from the NPR if this information is in the NPR. Otherwise, IP will build the lower portion of the path by calling Probe (for an 802 LI) or ARP (for an Ethernet LI).

The IP and ANH portion of the path is built from the information in the NPR.

Socket registry is able to build the PXP path record because all Socket Registries throughout the internet are bound to the same, well-known PXP port ID.

The Socket Registry then uses the path to send a query to Node B's Socket Registry.

When Node B's LI receives the query, it passes the message to IP and passes IP a reference to the LI's protocol ID (RTR, 802, or ETHER), and a reference to the path record used to receive the message.

When IP receives the message, it either builds or finds a path record suitable for this message. The path will have Node A's IP address as the remote address, Node B's address as the destination address, and PXP as the upper-layer protocol. IP will then either build or find an ANH record suitable for holding the routing information (Down PID and Down Path reference) for any return messages along this path. The ANH record is keyed by the ANH IP address, which is found by consulting the Gateway Table (GT). The ANH address is Node A, if Node A is on one of Node B's DCNs. Otherwise, the ANH is the address of the gateway on Node B's DCN through which the message will be routed.

IP then compares the Down PID and Down Path reference offered by the LI with the current routing information in the ANH record. If the offer matches the current information, or if there is no current information, the offer is accepted and return messages will use this path. If the offer and the current routing information do not match, then the current path will be accepted if the remote node (Node A) is not on Node B's DCN; otherwise, the current route will be replaced by the route offered by the LI.

Node B's PXP builds an i-rec path record (attached to Node B's Socket Registry's PXP service port's i-rec) to use to send a reply. The i-rec's destination PXP port ID contains Node A's Socket Registry's PXP port, and the destination IP address field contains Node A's IP address.

Node B's Socket Registry sends a reply with the CSR for the socket name, via the path consisting of: Node B's Socket Registry's socket record; the PXP path record; and the appropriate IP, ANH, and LI path records.

Node A's Socket Registry receives the CSR via the same path it used to send the query.

A reference to the CSR is returned to the calling process as the path report descriptor.

IPCConnect. With the path report descriptor returned by `IPCLookUp`, Process A calls `IPCConnect`.

Node A's TCP sets up a new TCP port for the VC socket. TCP must send a connection establishment request to the TCP port created for Process B's call socket. To do this, there must be a path consisting of: a TCP path record (d-rec and PCB); IP path record; and associated ANH and LI path records.

NetIPC calls Socket Registry software, which simulates an inbound message, and causes the protocol modules to build path records as they would for an inbound message. The protocol modules are able to do this from the information in the CSR as follows:

- If the LI information is contained in the CSR, the appropriate LI builds a path record, and passes a reference to the path record to IP.

  Otherwise, IP will use Probe to build the LI path when the path is used.
- IP creates an IP path record and ANH record, with Node A's IP address as the local address, Node B's IP address as the remote address, and TCP as the upper-layer protocol, as described in the previous subsection.
- TCP creates a d-rec containing: a source port field with Process A's VC's TCP port ID; a destination port field with Process B call socket's TCP port ID; a source IP address field with Node A's IP address; and a destination IP address field with Node B's IP address.

TCP sends a connection establishment request from Node A to the TCP port created for Process B's call socket.

Node B's IP receives the inbound message and searches for an IP path record with Node B's IP address as the local address, Node A's IP address as the remote address, and TCP as the upper-layer protocol. If one does not exist, IP creates a new IP path record, with references to an ANH record and LI path record, as previously described.

Node B's TCP receives the connection request and builds a new TCP port, and path record (d-rec and PCB). The d-rec for the new TCP port contains: a source port field with the new TCP port's ID; a destination port field with Process A's VC's TCP port ID; a source IP address field with Node B's IP address; and a destination IP address field with Node A's IP address.

Node A and Node B's TCP modules negotiate and establish a connection between the two TCP ports. TCP notifies NetIPC that it has received a connection request.

NetIPC creates a VC socket and socket record. The VC socket is queued on Process B's call socket.

IPCRecvCn. The VC socket created at Node B is de-queued from Process B's call socket and linked to the user.

Probe

The Probe protocol provides the following address resolution features:

- mapping a node name to a Nodal Path Report
- mapping an IP address to an LI address
- Nodal Registry Updates

Mapping a Node Name to a Nodal Path Report. Socket Registry uses Probe to resolve node names to NPRs as follows:

- Socket Registry first checks the local Nodal Registry for a NPR for the node name.
- If the local Nodal Registry is unable to resolve the node name, Socket Registry then uses Probe to send a Name Request to all LANs to which the local node is directly connected. If the named node is on a directly-connected LAN, it will respond to the Name Request with an NPR.
- If no node on the LAN responds, Probe will send a Proxy Name Request. If a node is a Proxy Nodal Registry Server, it can respond to a Proxy Name Request on behalf of any other node in the internet for which it has an NPR.

Once the local Socket Registry has an NPR for the remote node, it queries the remote Socket Registry for information on the named socket.

Mapping an IP address to an LI Address. IP uses this service whenever a message is being sent and the lower-layer portions of the path have not been built. Probe first searches the appropriate LI's local tables (for example, it will search the NRV for RTR LIs). If that search fails and the LI is an 802 LI, Probe will send a Virtual Address Request out on the appropriate LAN. Since RTR LIs do not support Probe, this information must be resolved from local tables if the node has no 802 LIs.

IP may request IP address to LI address mapping in the following situations:

- The destination node is on a Router/1000 DCN. (Router/1000 addresses are not kept in NPRs.)
- The destination node is an NS/3000 node on a directly-connected LAN. (NS/3000 NPRs do not contain LAN station addresses.)
- The destination node is an HP 1000 node on a directly-connected LAN, but the LAN station address was not configured in the local node's NPR for that node.
- The destination node is not on any DCN; an intermediate gateway on the DCN will be used.
- At an IP gateway, IP does not use Socket Registry to build the LI path for outbound store-and-forward messages.
- For DS/1000-IV Compatible Service messages sent over a non-Router/1000 LI (802 or ETHER), IP does not use Socket Registry to build the LI path.

Nodal Registry Updates. NPRs are entered in Nodal Registeries either explicitly by the Network Manager via the NS-ARPA/1000 utility, NRINIT, or by Probe updates. Probe updates occur periodically, and when SR attempts to resolve node names, or when a node is initialized as follows:

- IEEE 802.3 LAN node initialization—when an HP 1000 node is initialized on an IEEE 802.3 LAN, Probe multicasts the node's name and NPR via Probe Unsolicited Replies. If another IEEE 802.3 LAN node's NR already has an entry for that name, NR will overwrite the previous NR entry for the node name. If there is no NR entry for that name, the NR will ignore the Unsolicited Reply.

- Periodic updates. Every 5 minutes, Probe multicasts the local node's name and NPR via Probe Unsolicited Replies. The other NRs on the IEEE 802.3 LAN will process the Unsolicited Replies as described above.
- Node name request—when Probe gets an NPR as a result of a Name Request or Proxy Name Request, it will dynamically add the node's name and the NPR to the local Nodal Registry. Probe will add as many dynamic entries as there are spare Nodal Path Reports. The number of spare NPRs is configurable by the Network Manager.

Physical Data Flow

This subsection describes the physical movement of data through the NS-ARPA subsystem.

Messages

In this manual, the term message is used in a general sense—that is, a discrete amount of data sent by a user, plus the protocol headers as they are added. To a given protocol, a message passed from the upper-layer protocol is data, even if it includes upper-layer protocol headers. A given protocol considers a message to be the data passed from the upper-layer protocol, plus the given protocol's header.

Outbound Messages

- The user sends a buffer of data by calling `IPCSend` using NetIPC or `send()` using BSD IPC.
- NetIPC or BSD IPC copies the data from the user's program space to DSAM. NetIPC or BSD IPC passes a reference to the message's location in DSAM to TCP.
- TCP makes a copy of the message for retransmission and fragments the message, if necessary. (Fragmentation is discussed in the subsection "Message Size," below.) TCP prepends its header to the data and passes a reference to the message's location to IP.
- IP fragments the message, if necessary. IP prepends its header to the data and passes a reference to message's location to the appropriate LI.
- If the LI has a header to add, it prepends its header to the data.
- The LI moves the message from DSAM to SAM, queuing the message on the transmit link LU.
- The device driver reads the message to the interface card, and the card transmits the message.
- If tracing is enabled, INPRO requeues the completion onto NSTRC's class number, so that NSTRC can write the message to the trace file.
- INPRO receives the write completion from the interface card driver and returns the driver status to the LI.

Inbound Messages

- A message arrives at the interface card and the RTE driver associated with the card is entered.
- For LAN links, the LAN driver moves the message from the card to SAM and queues the message on INPRO's class. For RTR links, the RTR driver schedules QUEUE. QUEUE executes a class read with GRPM's class number. The RTR driver moves the message from the card to SAM. GRPM rethreads IP messages to INPRO's class.
- INPRO reads the message from SAM with a class get. If tracing is not active, the SAM buffer is released.
- If tracing is activated, the copy of the message in SAM is rethreaded on NSTRC's class number.
- INPRO processes the message at the LI, IP and TCP levels. At each level, the header information is checked and is used to determine the next level protocol.
- If an IP message is fragmented, or a TCP segment arrives out of order, the messages are buffered in DSAM until reassembly can be completed.
- After TCP processing is complete, any user data is moved to DSAM and queued in the inbound buffer of the socket.
- The user receives the message by calling `IPCRecv` using NetIPC or `recv()` using BSD IPC. NetIPC or BSD IPC moves the message from DSAM to the user's program space.

Message Size

The size of a message may change as the different protocols handle it. Message size is determined by the following variables:

- The size of the user data buffer (*`userData`*).
- The configured TCP segment size (*`TCPSegment`*). This is the size of the buffer that TCP uses to hold the user data and does not include the TCP header.
- The configured network segment size for the Directly Connected Network that IP is to route the message through (*`networkSegment`*). This is the size of the buffer that IP will send to the LI, and includes the IP and TCP headers.
- The maximum buffer size accepted by the link interface (*`LIBuffer`*). For RTR LIs, the device driver will accept buffers up to 8192 bytes long, even though the interface cards will send buffers only up to 1024 bytes long (including any LI header but not including the data link header); this is because the device driver will buffer data for the interface card.

  For 802 LIs, the interface cards will send buffers up to 1490 bytes long (not including the data link header).

The value of *networkSegment* must be less than or equal to *LIBuffer*. In the IP section of the NSINIT dialogue, the maximum allowed values for Network Segment size are equal to the *LIBuffer* for the particular link type.

Assume that the user sends a data buffer that is *userData* bytes long. If *userData* is greater than *TCPSegment*, TCP will fragment the user's data buffers to fit *TCPSegment*.

If *userData* is less than *TCPSegment*, TCP will concatenate the user's data buffers to fit *TCPSegment*, unless the user sends an end of data indication. In that case, TCP will immediately send the user's data buffer.

If *TCPSegment* plus the length of TCP's and IP's headers is more than *networkSegment*, IP will fragment the message and request TCP to use *networkSegment* minus the length of IP's header for the TCP segment size for subsequent messages sent via that IP path record.

If *TCPSegment* plus the length of TCP's and IP's headers is less than *networkSegment*, IP will send buffers that are as big as *TCPSegment* plus the TCP and IP headers.

In the case of store-and-forward traffic, IP may receive a message that is larger than *networkSegment* for the Appropriate Next Hop. IP will fragment the message, but the message will not be reassembled until it reaches the destination node.

DS/1000-IV Compatible Services Internals

This subsection describes the internal organization of the DS/1000-IV Compatible Services software. Included in this subsection is a detailed description of each DS/1000-IV module and the functional relationships between major modules. The DS/1000-IV Compatible Services are described in the *NS-ARPA/1000 DS/1000-IV Compatible Services Reference Manual*.

Data Flow through the DS/1000-IV Software

The first part of this discussion describes RTE-RTE communication paths. RTE-MPE communication is described in "DS/1000-IV Compatible Services Communication (RTE-MPE)" later in this subsection.

Figure 9-5 is a conceptual view of the major DS/1000-IV modules in an RTE system. The arrows indicate the path followed by a simple NS-ARPA request after it is issued from a user program and is transmitted to a destination node over an HDLC link.

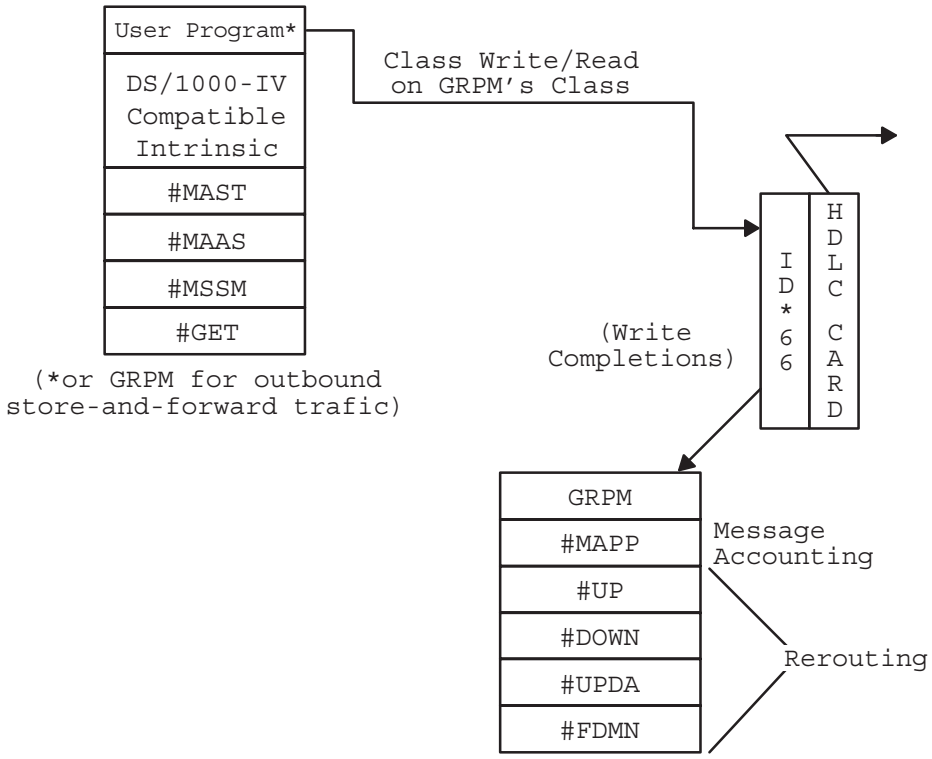


Figure 9-5. Sending Node Data Flow, HDLC Link

Figure 9-6 is similar to Figure 9-5, but illustrates the path followed by a request that is transmitted over an IEEE 802.3 link. Unlike the path flow illustrated in Figure 9-5, a request transmitted over an IEEE 802.3 link passes through the NS-ARPA modules IFPM and OUTPRO, where 802 and IP headers are added, before being sent to the driver. In addition, read and write completions pass through INPRO before they are sent to GRPM.

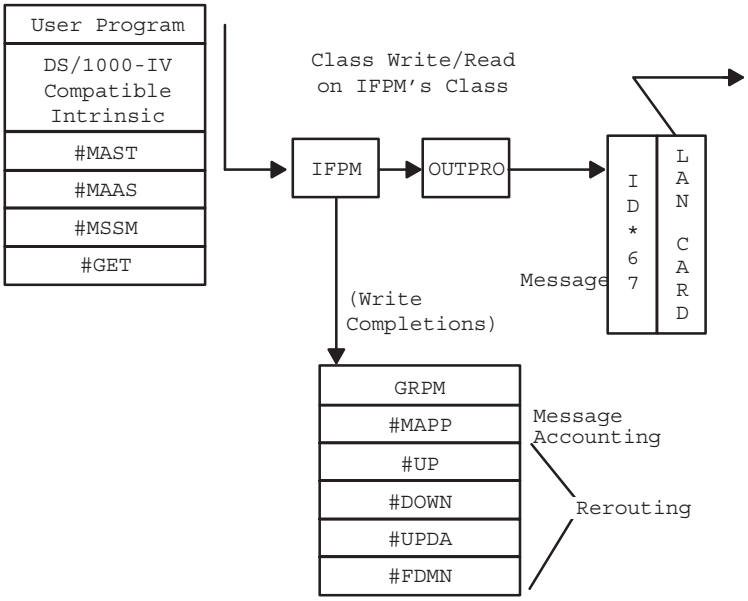


Figure 9-6. Sending Node Data Flow, 802 Link

Figure 9-7 is a conceptual view of the major modules and their relationships at the destination HP 1000 node. The arrows indicate the path followed by a message when it is received by the node over an HDLC link.

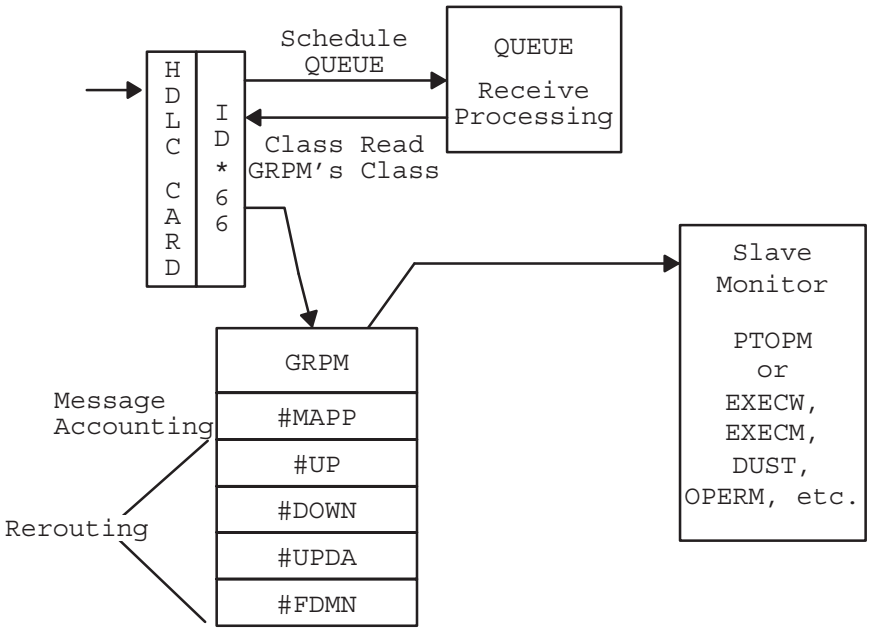


Figure 9-7. Receiving Node Data Flow, HDLC Link

Figure 9-8 is similar to Figure 9-7, but illustrates the path followed by an incoming message when it is received over a 802 link. Unlike the path flow illustrated in Figure 9-7, a request received over a 802 link passes through INPRO, where the IP and 802 headers are stripped, before being sent to GRPM.

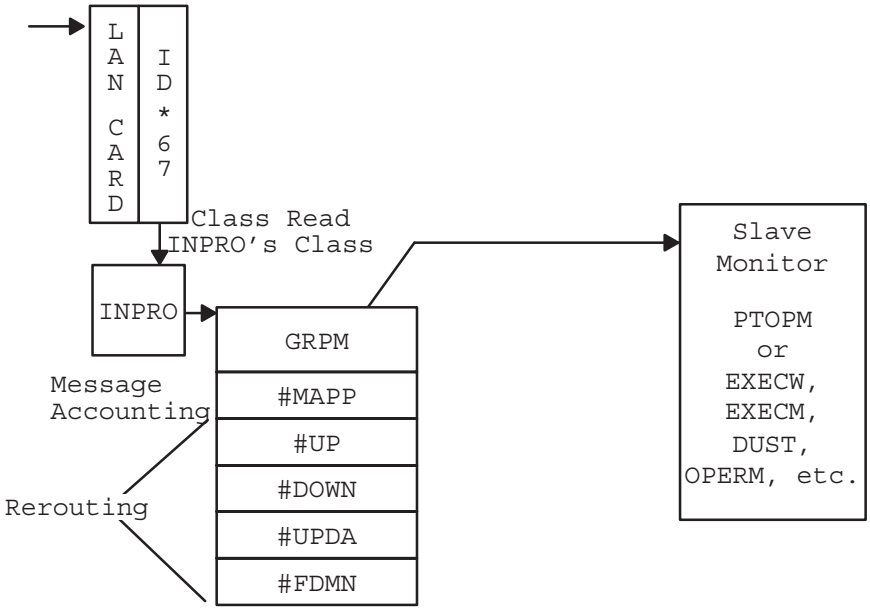


Figure 9-8. Receiving Node Data Flow, 802 Link

The following paragraphs refer to Figure 9-5, Figure 9-6, Figure 9-7, and Figure 9-8 and explain the progress of a NS-ARPA request through the major DS/1000-IV modules.

Note Refer to the "NS-ARPA/1000 Path Flow" subsection earlier in this section for an explanation of the NS-ARPA modules IFPM, INPRO, and OUTPRO.

An NS-ARPA request is initiated when a user program makes a *master request subroutine call* to one of the internal DS/1000-IV master subroutines. The DS/1000-IV master request subroutine calls include

- the DEXEC calls
- the PTOP master calls (`POPEN`, `PREAD`, `PWRIT`, `PCONT`)
- the HP 1000 RFA calls (`DAPOS`, `DCLOS`, `DCONT`, `DCRET`, `DLOCF`, `DNAME`, `DOPEN`, `DPOSN`, `DPURG`, `DREAD`, `DSTAT`, `DWIND`, and `DWRIT`)
- the utility subroutines `DLGON`, `DMESS`, `DLGOF`, `DLGNS`, `DMSG`, and `FCOPY`.

The function of the internal DS/1000-IV master subroutines is to build a request buffer containing the parameters necessary to describe the request. This buffer is passed into the communications management software by the subroutine `#MAST`. Depending upon the request, data may be sent along with the request.

`#MAST` allocates a class number to be used later for receiving the reply. If sufficient class numbers do not exist in the RTE system, the master program is suspended until one is free. `#MAST` then calls `#NRVS` which converts the nodal address to the appropriate NS-ARPA LU. `#NRVS` performs a table lookup on the *Nodal Routing Vector* (NRV). `#NRVS` determines whether the outgoing link is a Router/1000 or non-Router/1000 link. If the link is Router/1000, `#NRVS` determines the NS-ARPA LU of the link. If the link is non-Router/1000, `#MAST` performs a Class I/O write/read call to LU 0 using IFPM's class number. If the link is Router/1000, `#MAST` performs a Class I/O write/read call to the NS-ARPA LU using GRPM's class number. A `DS04(0)` error is returned to the caller if the node does not exist in the NRV.

If *Message Accounting* (MA) has been generated into the local node and/or is present at the destination node, the MA routine `#MAAS` is called to assign a MA sequence number to the message. A sequence number is assigned if the number of unacknowledged outstanding messages has not been exceeded.

A *Transaction Control Block* (TCB) entry is allocated (taken from the free list of TCBs) and entered in the Master TCB list. The TCB is used to keep track of the request until a reply is received or the request times out. (TCB allocation is described in detail in "UPLIN and the Transaction Control Block (TCB)" later in this subsection.) The number of TCBs is established by NSINIT when the node is initialized. If no TCBs are available, the master program is suspended until one becomes free.

If there is remote session capability anywhere in the network, `#MAST` calls the Remote Session routine `#MSSM` to assign a Remote Session ID to the message. The remote session will not exist if

- there has been no prior logon via `DLOGN`
- the user program was not scheduled by a father with an existing remote session

- this is the user program's first message sent to the destination node
- the user program was not scheduled from a remote session.

When the first message is sent and acknowledged, a remote session is established and the session ID is returned. This ID is saved in the *Network Account Table* (NAT) and is used when additional messages are generated to the same remote node. The NAT is contained in the module `#MSSM` which is appended to the user program. This table has room for 16 entries, enabling a user program to have remote sessions active at up to 16 different remote nodes. Only one remote session per node per program is allowed; any number of programs may each have their own set of 16 sessions.

When a message is sent, MA waits for an *acknowledge indication*. The acknowledge indication is contained in message traffic returning from the destination node. MA sends up to 15 messages to a particular destination before requiring an acknowledgement. When the maximum number of messages has been sent and no acknowledgement messages returned, `#MAAS` returns a `DS08(1)` node busy error to any program that attempts to initiate additional traffic to that particular MA destination node. `#MAAS` updates the MA tables in System Available Memory (SAM) and returns to the calling routine (`#MAST` in this case).

If the destination node has the same upgrade level as the local node, `#MAST` uses a Class I/O write/read call, specifying the transmission LU (zero if destined for local execution) to transmit the request. Two buffers are sent using the double buffer feature of Class I/O. If the message contains data, it is placed in the first buffer; the DS/1000-IV request header is placed in the second buffer.

When the request is transmitted from `#MAST`, the *General Request Preprocess Monitor's* (GRPM) class number, `#GRPM`, is used. GRPM receives the status of the I/O transmission upon completion. If the destination node is also the local node, all processing is done via Class I/O without communication line activity. Refer to the appropriate RTE reference manuals for a description of Class I/O.

`#MAST` calls `#GET` to wait for the reply, or master timeout, utilizing a Class I/O get on the master requestor's class. (The master class is allocated by `#MAST` for use with a particular message.) The class number is stored in the TCB so that, when a reply arrives or a timeout occurs, the reply data or error can be returned to the master via a Class I/O write/read. When the reply arrives, the class number and the buffer in SAM is deallocated and becomes available for other uses.

UPLIN and the Transaction Control Block (TCB)

It is possible at any given moment for several programs in one computer to be waiting for a reply from another computer or computers. There is no requirement that the replies arrive in the same order that the requests were sent. The *Transaction Control Block* (TCB) is used by the DS/1000-IV Compatible software to locate the proper receiver for each incoming reply.

If a reply is not received, the system protects the master caller from waiting indefinitely by placing a timeout counter in the TCB. This counter is incremented by `UPLIN`, a program which runs every five seconds. If the counter reaches 377 octal before the reply arrives, UPLIN issues a Class I/O write/read request with a data length of zero on the master requestor's class number. This indicates a timeout condition, error `DS05(0)`.

The TCB also contains a *local sequence number*. A local sequence number is a unique number allocated to each newly-created TCB by #RSAX. This sequence number allows the system to recognize and ignore replies that return when the requestor has been aborted. It is also possible for the aborted program to be re-run and to make a similar, or even identical, request before the first reply arrives. The system is able to distinguish between the first and subsequent replies because the local sequence number is incremented for each request and is stored in the TCB and in the header of each request. The slave side software echoes the last sequence number which, in incoming replies, is checked against the sequence number in the TCB. If a reply does not match any existing TCB it is ignored and the error message TCB NOT FOUND, POSSIBLE TIMEOUT is printed on the system console of the receiving side.

The TCB also contains the class number to be used for sending the reply back to the master and the master's ID segment address. Refer to the *NS Message Formats Manual* for specific formats.

Because of the actions of UPLIN, system resources are never unrecoverable. Class numbers and SAM buffers are recovered when one of the following events occurs:

- a reply arrives
- the master is kept waiting too long and a master timeout occurs
- the master program is aborted.

Class I/O Rethreads

All transactions, except those which must be processed by the message converters, are sent from one processing stage to another by calling DSQ to re-thread them onto another class. This process links the class buffer into the list of buffers waiting on another class. The buffer could be sent to the new class by issuing a Class I/O write/read, but re-threading (changing the buffer linkage from one class list to another) is much more efficient. The software at each stage of processing would otherwise require maximum-size buffers, which, in the case of program-to-program communication, is 4096 words. Using the rethread method, the software modules require a buffer only large enough to contain the message header, excluding the data.

The number of requests that can be queued onto a slave program's class (this applies to a user's slave program in a PTOP mode as well as to a slave monitor such as EXECM) is limited to prevent a node from being flooded by requests sent to only one program. This top limit plus one is stored as a 1's complement of the value in #QLIM. (The default is -11. The actual queue limit defaults to 10.) If DSQ detects that the limit has been exceeded when it attempts to add a new request onto the queue, it rejects the request and a DS08(4) error is returned to the originating node. This check is bypassed when the request is re-threaded to a DVT, such as in the case of a store-and-forward.

Failures on HDLC Links with Dynamic Rerouting

The *dynamic rerouting software* exists as a collection of subroutines appended to NSINIT, DSMOD, and GRPM. When Dynamic Rerouting is present in the system, the GRPM error processing sequence changes. When an error occurs, the appropriate number of retries have already been attempted by the card and GRPM calls the appended rerouting modules to down the link and

attempt to find a new path to the destination node. If all paths to the destination node are down, a DS04(1) is returned to the user.

If a new path is found, the local NRV is updated to reflect the failed link and the new path. The rerouting program #SEND is scheduled and sends NRV update messages to all neighbor nodes. #SEND examines the NRV for change flags set by #DOWN, which is a rerouting routine appended to GRPM. When a changed entry is located, the Router/1000 node address and cost value are packaged and sent to all neighbor nodes. The neighbor nodes send any changes they make to their own NRVs on to their neighbor nodes, and so on, until all the nodes in the network have been updated with the new routing information. The algorithm converges rapidly because no new updates are sent upon reception of an update which does not cause a change to the receiving node's NRV.

When a link is restored to service, the driver receives an indication from the card that the link is functioning again. The driver passes this information on to GRPM. GRPM calls the rerouting link up processor to update the link vector and NRV and schedules #SEND to broadcast the change to all the neighbor nodes.

Failures on Non-Rerouting HDLC Links

When HDLC links without rerouting have failures, the message currently being transmitted is retried until the retry limit is reached and then a failure is returned to the driver. The link is marked down, but the failure is not returned immediately to the master program. This is because the master request has already received a successful completion when the message was received by the card. This request will receive a DS05(0) timeout error after the master timeout period.

Any further traffic to the failed link receives an error immediately (DS01, DS02, and so forth), until the link is restored. When the link is restored, the HDLC card is returned to service automatically and begins accepting messages.

HDLC Message Processing

When a message is queued to the HDLC driver, ID*66, from #MAST, ID*66 sends a message to the card to determine how much space is available on the card. The space on the card is managed in memory blocks called *frames*. The frame size is user selectable and is usually set to 1024 bytes for hardwired links and 128 bytes for modem links. When the card returns with the amount of space available, the driver breaks up the message into frames and sends the indicated amount of frames to the card.

Appended to the front of the first frame are two words containing the request header length and data length. The interaction between card and driver is accomplished using LIA instructions when sending commands to and from the card, while the actual frame transfers occur using DMA. The first two length words of a message are latched into memory using LIA instructions; the rest of the frame is transfered via DMA.

Once the card receives one frame, it can begin transmitting it to the interface at the other end of the link. If there is space on the card, additional frames of the message are transferred to the card. If the card is full, additional DCPC transfers can occur bringing additional portions of the message to the card as the frame buffers become empty. On hardwired links, where the frame size is 1024 bytes, the complete messages usually fit in one frame. A burst of traffic should go to the

card at the rate of the DCPC channel until the card memory is filled, then the card empties its buffers at the line rate. The card has 7K bytes of transmit buffers and 7K bytes of receive buffers. At any one time, several messages may be contained entirely on the card.

At the other side of the link, an interrupt is generated to notify the driver once the first frame is received. The driver reads the first two words of the first frame to get the length words used to schedule QUEUE. The length data is then stored in the DVT and QUEUE is scheduled. QUEUE performs a Class I/O read back to the driver. If insufficient SAM exists to receive the message, QUEUE attempts to receive just the request header so that GRPM can return a `DS08(0)` remote busy indication to the origin node. If sufficient SAM is available, QUEUE issues a `SEND STOP` control request to the driver. This condition is eventually treated as a remote busy condition by the sending side.

The message is read from the card one frame at a time. The read takes place on GRPM's class number and the I/O completion is received by GRPM for processing when the complete message is transferred from the card.

I/O Completion Processing

When the driver completes any operation (read or write), GRPM processes the status. If an error occurs, GRPM may retry, return an error code, or print a message depending on the circumstances. If the operation was a successful transmission, the class buffer is released and GRPM awaits the next I/O completion.

If the operation was an unsuccessful transmission, GRPM examines the cause of the error. All retries are performed on the card. Only in the case of the remote busy error does GRPM route an ID*66 request to RTRY for additional processing. The number of retries is determined by the number of remote busy retries configured for the node. This value can be altered with the DSMOD `/T` command. (The default is 3.)

When the rerouting option is generated in and the driver receives a link failure indication from the card, the rerouting software appended to GRPM is activated as described earlier.

In the case of successful reception, the addressed node (destination node for requests, origination node for replies) is compared to the local node number. If address does not indicate the local node, the NRV is used to find the logical unit for transmission. A store-and-forward operation is initiated by threading the buffer on the transmission device using GRPM's class number. (Note that the buffer does not have to be moved, only re-threaded, which saves time and CPU utilization. Later, when the transmission completes, GRPM receives the status as a write completion and deals with it as described above.)

If the message received was addressed to the local node, further processing depends on whether it is a request or a reply. If it is a request, a slave TCB is allocated and the class buffer is threaded onto the monitor's class number. If it is a reply, the master TCB is found by matching the transaction sequence number of the message (not the MA sequence number) with the TCB entry with the same sequence number; the class buffer is threaded onto the master's class number. GRPM then awaits the next I/O completion.

When Message Accounting is present in the system, GRPM does additional processing on each I/O completion. On transmission, the message accounting Post Processor Module (`#MAPP`) is called by GRPM to determine whether the message can be regenerated in case of failure. In master programs, all messages can be regenerated. If the master is using the no-wait feature of

PTOP, no Message Accounting support is provided. In the slave monitor case or in the case of PTOP slave routines, the message cannot be regenerated so the reply messages are requeued to a holding class (a queue in SAM) until an acknowledgement message arrives to indicate that the reply was successfully received at its destination. At this time #MAPP deallocates the message from the hold class.

On transmit completions, messages that can be regenerated are deallocated from SAM. On receive processing, GRPM calls #MAPP to determine if the message is valid. The validity check involves checking to see if the message sequence number falls within a proper range or window of sequence numbers valid for the node. This window consists of 15 consecutive MA sequence numbers, beginning with the sequence number assigned to the message that has been waiting longest for an acknowledgement. As acknowledgements come in, the window of valid sequence numbers advances. The first received message may have been previously cancelled due to a time out at the origin node or routing changes that caused the cancel message to arrive prior to the actual message. The message may also have already been received, in which case it is discarded.

The message may also be an MA internal message which is processed only by #MAPP and discarded. These include the message cancel requests from other MA nodes and no-response messages from the driver. Message Accounting processing is discussed in detail later in this chapter.

Slave-Side Processing

All slave monitors await requests by performing a Class I/O get request, usually using the routine #GETR (which has expanded capabilities over #GET). When the request (and data, if any) is received, the buffers are moved from SAM into the monitor's local buffers and the class buffer (SAM area) is deallocated. For some monitors, such as PTOPM, where there is no reason to examine the data, the request header is received by the monitor and examined to determine what action is required. Generally in this case, the message is simply requeued to a slave program's class. In any case, the class number is not deallocated. #GETR returns to the monitor which then performs the master's requested function. When the monitor has completed servicing the request, it builds a reply buffer, supplying the appropriate information and calls subroutine #SLAV to return the reply (and data, if any). #SLAV deallocates the slave TCB and then calls #NRVS which converts the originator's nodal address to the appropriate output LU by performing a lookup on the NRV. After the NRV lookup, #NRVS determines whether the LU is an Router/1000 link or a non-Router/1000 link. If it is a non-Router/1000 link, #SLAV performs a Class I/O write/read on LU 0 using IFPM's class number. If it is a Router/1000 link, #SLAV does a Class I/O write/read onto the output LU using GRPM's class number. #SLAV then returns to the monitor which in turn calls #GET to await its next task, by suspending on a Class I/O get.

The handling of data transmission, store-and-forward, and error retries for replies, is essentially the same as for the initial request handling by ID*66, GRPM, and RTRY. The fundamental difference is that the message is directed toward the originating nodal address rather than the destination nodal address. If an error occurs that cannot be corrected by retries, or all retries have been exhausted, then the reply is flushed. The node which does this prints an error REPLY FLUSHED on its system console and the master receives a DS05 timeout error. If MA is generated in and active, the error qualification is used to distinguish between requests and replies being lost.

When the reply (and data, if any) arrives at the originating node, GRPM locates the TCB for the master requestor and, finding the class number, rethreads the reply (and data, if any) onto the master's class.

Recall that #GET is left suspended on the master's class number which was allocated for this purpose. The master program resumes execution when the reply is threaded onto its class number. #GET checks that the reply (and data, if any) has a length not larger than the allowable limit. If either one is larger, a DS03 error is returned. Otherwise, the data, if any, is moved into the caller's data buffer. The class number and TCB are deallocated. Control returns to the user's program along with any error information. In a DEXEC request, if the no abort bit is not set and an abortive error occurs, #TILT is called to abort the master program.

If UPLIN has timed out the master request, it writes a zero-length buffer to the master's class number, which #MAST interprets and returns to the caller as a DS05 error. In any case, the class number and TCB are always deallocated when the master program receives control or is terminated.

A more detailed description of each of the communications management software modules follows.

X.25 Links

X.25 links use the DS customizing subroutine CXL66 attached to the X.25 driver DDX00.REL instead of driver ID*66.

A general description of X.25 is included in the *NS-ARPA/1000 Generation and Initialization Manual*. For a detailed description of X.25, refer to the *X.25/1000 Reference Manual* and the *X.25/1000 Advanced Guide*.

#MAST

#MAST performs the following functions:

- Clears certain fields in the request header and enters the maximum loop count used to ensure messages cannot become caught in a loop due to manually entered incorrect routing information or possible transient conditions during topology changes. This field is decremented by each node that processes the message. The message is flushed if the count becomes zero.
- Returns DS00(1) to the caller if the system is not initialized. Returns DS00(3) to the caller if the system is quiescent.
- Allocates a class number for the request.
- Locks the RESA table access resource number, thereby guaranteeing that no other programs can modify the table. If there are no TCBs available, this RN is already locked, causing the calling program to be suspended until one is available.

- Removes a TCB from the TCB pool, and links it into the master-request list, fills in the table entry information then clears the table-lock resource number (#RSAX performs this operation).
- Determines if the outgoing link is Router/1000 or non-Router/1000 (#NRVS).
- If outgoing link is Router/1000, #NRVS converts destination Router/1000 node address to a logical unit number.
- Assigns the Message Accounting Sequence number to the message by calling #MAAS.
- Assigns a Remote and a Local Session ID by calling #MSSM.
- If outgoing link is Router/1000, sends the request, with data, if any, on the link. If outgoing link is non-Router/1000, performs a Class I/O write/read on LU 0 using IFPM's class number.
- Calls #GET to await the reply (and data, if any).
- When the reply arrives, if it was a quiescent reject, puts the calling program in the time-list, awaiting retry later, if the Remote Quiet Wait value is non-zero. Otherwise, an error is returned.
- If the error flag in the seventh reply word equals one, the master class number and TCB are deallocated and an error is returned.
- Deallocates the master class number and TCB. If the master program is aborted while waiting for a reply, the system receives the reply eventually, acknowledges it as usual, and links it to the TCB. The TCB is eventually timed out by UPLIN. UPLIN clears the TCB and releases the class buffer and class number. If UPLIN detects that the program has aborted prior to the reply message arriving, UPLIN deallocates the TCB. When the reply does arrive, if GRPM fails to find a matching TCB, it flushes the reply, and prints TCB NOT FOUND, POSSIBLE TIMEOUT via QCLM.
- Calls #MSSM to store the returned Remote Session ID, if any, for use in additional requests from the master program.
- If the reply is successful, returns the received lengths in the A and B registers.

#MAST performs the following error processing:

- If the sign bit (bit 15) of the passed control word parameter is set, ASCII error codes are supplied to the caller in the A- and B-registers.
- If the sign bit is not set, the routine #TILT prints the error message, program name, reporting Router/1000 node address, and aborts the calling program. (#TILT is a subroutine internal to #MAST.)

GRPM

GRPM is the general request pre-process module for DS/1000-IV Compatible Services (RTE-RTE) communications. It handles incoming messages (requests and replies), and outgoing transmissions.

Most of the rerouting processing is implemented by routines appended to and called by GRPM. In addition, a large portion of the MA feature is implemented by appending the MA Pre- and Post-Processor (`#MAPP`) to GRPM. The following paragraphs explain this process.

Incoming Requests

GRPM awaits new input via a Class I/O get on its class number. The class number is allocated by NSINIT when the node is initialized and kept in RESA at entry point `#GRPM`. GRPM processes completion status of messages transmitted by `#MAST` (Router/1000) or IFPM (non-Router/1000), incoming messages received by QUEUE, and update messages for rerouting and MA tables.

NRV update messages (routing) and MA acknowledge requests are sent across the network on stream zero and are processed by the rerouting and MA routines appended to GRPM.

The incoming messages are rethreaded onto GRPM's class by QUEUE.

Store-and-Forward Traffic. If the request destination is not the local node, the line error retry counter in the stream word of the request is cleared, the hop count is incremented and the request (and data, if any) are re-threaded for output to the appropriate communication line logical unit on GRPM's class number.

Requests. If the incoming request is a new request to the local node, then:

- GRPM checks for stream zero messages (MA and rerouting). If so, it is processed locally by the `#MAPP` or one of the rerouting routines appended to GRPM.
- If TCBs are not available, GRPM sets the reply and remote-busy flags in the stream-type word and returns the request to the originator by rethreading for output on GRPM's own class, queued to a device for transmission back to the sender.
- If the system is going quiescent, or if the required monitor is in unavailable-memory suspend (state 4), the busy flag is set in the request stream-type then the entire transaction is returned to the originator. Otherwise, a slave TCB is allocated from the available pool.

Replies. If the incoming transaction is a reply and destined for the local node, GRPM checks the busy flag:

- If it is set and the remote-busy retry count has not been exhausted, the flag is cleared and the buffer re-threaded onto RTRY's class number with a delay of one second. If the busy flag is set and no remote-busy retries remain, the master requestor receives a `DS08` error.

- If no error conditions exist, the list of master TCBs is searched to find the master program for this request. If found, the reply is re-threaded onto the master's class number. If the master TCB is not found, the message in SAM is released, and a block is sent to QCLM indicating a communications error. The message `TCB NOT FOUND, POSSIBLE TIMEOUT` is printed on the local operator's console. GRPM then awaits the next incoming data block via a Class I/O get.

Outgoing Line Completions

Transmit completion processing is performed by GRPM only to handle the error conditions that may occur and pass messages to the Transmission Retry Processor, RTRY, when necessary. When no errors exist, GRPM calls MA to perform post-processing on the transmitted message and then returns to the Class I/O get to wait for the next request.

The MA post-processing in the case of outbound slave reply messages consists of requeuing the message to a holding class. For requests, MA post-processing simply releases the SAM buffer.

GRPM processes completion status of all communication write operations (except system download messages which are handled by PROGL). If a write operation is successful, the message is deallocated from SAM. On remote-busy errors, GRPM checks the retry count in the stream word of the request. If all retries have been exhausted, the error is treated the same as a line error and a DS08 error code is returned. If another retry is possible, the absolute system time at which the retry should be attempted is computed and stored in the local appendage in the NS-ARPA message header. The class buffer is then re-threaded onto RTRY's class.

Retries are handled by the card. If a catastrophic failure occurs, the card returns the failure status to the driver and downs the link. ID*66 returns the failure condition to GRPM. If this is a rerouting link, the rerouting module `#DOWN` attempts to find an alternate route. If successful, the NRV is updated with the new LU and `#SEND` sends NRV update messages to all neighbor nodes. If no alternate exists, an error is returned to the master program and the LU in the NRV for this destination node is set to zero. Future requests on this LU are returned with `DS04` errors.

Dynamic Rerouting

As mentioned earlier, the Dynamic Rerouting feature consists of several subroutines appended to GRPM and routing tables in SAM which are used to dynamically determine optimum routing in the network at any given time.

The routing tables are the Cost Matrix, Link Vector, and the NRV.

Cost Matrix

The *Cost Matrix* is a two-dimensional matrix holding the relative cost value to reach each destination over all possible rerouting links out of this node. Each entry in the table is 2 words. The first word is the cost value to get to a particular destination using a particular rerouting link. The cost value is summed to indicate the total cost to get to the destination; this is maintained by rerouting message traffic. The second word is the loop count (number of intervening nodes) to the destination node from this node. The entry point `#CM` in RESA contains the base address of the

Cost Matrix. The number of entries is equal to the number of Rerouting links in the node times the number of nodes.

Link Vector

The *Link Vector* is a table of six-word entries, one entry per link. Each entry contains

- Link Status (up or down).
- LU of the link.
- System Time (2 words). Used to keep track of the number of times the link went down in a given 5-minute period.
- Up/Down Counter. Counts the number of link failures. When the number of temporary failures in a given 5-minute period exceeds the maximum line down count (alterable with the DSMOD `/T` command), the link is declared down and rerouting attempts to find other paths around the failed link.
- Neighbor Node number. Used by `#SEND` when sending NRV update messages to all neighbor nodes. Used by Rerouting to indicate the current best path (LU) to a given destination node.

Nodal Routing Vector (NRV)

Router/1000 uses the *Nodal Routing Vector* (NRV) for subnet routing within Router/1000 networks. The NRV contains the following information:

- an entry for every node on the local network connected via a Router/1000 link
- an entry for every remote HP 1000 in the catenet that supports DS/1000-IV Compatible Services, including DS/1000-IV (91750) nodes.

Dynamic Rerouting Processing

As mentioned earlier, GRPM contains most of the software required to implement rerouting. GRPM contains a module to process rerouting link failures, link restorations, and a module to process routing update information from other nodes in the network.

Link Failure Procedure

When the driver completes any I/O operation to an NS-ARPA/1000 communication link, GRPM processes the result. If the status is an unsuccessful transmission, GRPM examines the cause of the error and does necessary retries above the driver level. If the retries report a link/card failure, GRPM returns a `DS01` or `DS02` error to the user. With rerouting, instead of returning a link/card failure condition to the user, GRPM will enter a link failure procedure. In the link failure routine `#DOWN`, GRPM notes the downed link and issues the following message, `NS/1000: LU # lu WENT DOWN time`.

It then tries to find an alternate route to the destination if the link failure causes a change in the minimum-cost route to the destination. It also tries to broadcast the failure condition to the network by setting the change bit in the NRV so that the #SEND process can send out a Network Update Message to its neighbors.

Finally, the MA routines regenerate the user message for retry if there is an alternate route, or return a DS04(1) error to the user if there is no path to the destination (the LU entry in the NRV is set to zero so that further messages to that destination will also result in a DS04(1) error).

Up Link Procedure

When the link level software discovers an operational link, it sends a link connection indication to GRPM. GRPM enters the up link procedure. GRPM notes the return of the link by issuing the message NS/1000: LU # *lu* CAME UP *time*.

GRPM also sends an update message describing its knowledge of the network through the newly-connected link via #SEND. GRPM then expects to receive an update message from the newly-connected neighbor showing its view of the network. These messages go to the network update message processor which set up new, minimum cost routes.

Network Update Message Procedure

GRPM recognizes an incoming Network Update Message and then enters the update procedure, #UPDA. This routine updates the cost to a destination referred by the message. If the minimum-cost route is changed due to the update, the new least-cost path is reflected in the NRV and routing changes are passed to the neighbors by setting the change bits in the NRV and then scheduling #SEND.

The #SEND Process

#SEND is scheduled without wait by the link failure procedure #DOWN or the update procedure #UPDA to send a Network Update Message to all the neighbor nodes. It first checks to see if the send signal (#CMCT in RESA) has been set by the link failure procedure or the update procedure. If the send signal is not set, #SEND exits. If the send signal is set, #SEND goes through the NRV looking for the change bit. When found, the Router/1000 node address and its cost value from the cost matrix are packaged to be sent to all the neighbor nodes. When this process is completed, #SEND returns to test for the send signal.

Compatibility

Rerouting nodes are completely compatible with non-rerouting nodes; that is, traffic between them may exist. However, rerouting will only occur between nodes that have rerouting software. In Figure 9-9, R indicates rerouting nodes and N indicates non-rerouting nodes.

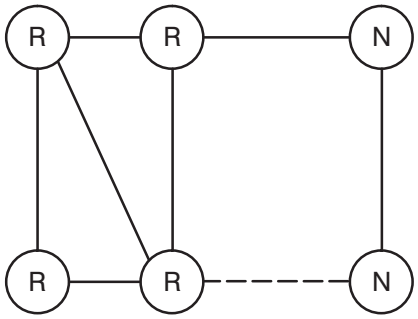


Figure 9-9. Rerouting Topology Considerations

Dynamic rerouting can only occur between the R nodes. The extra link to N (denoted by the broken line) does not provide any more reliability for the network because the software can only use a fixed path as long as the node on the other side of a link has initialized the link as non-rerouting. A manual switch of the LU using the DSMOD CN command can restore communications on the non-rerouting links.

For a link to be a rerouting link, both nodes on either side must have initialized the link as a rerouting link. If an LU is specified during initialization, the link is treated as a non-rerouting link.

However, while compatible in the network, there are some topology restrictions that make it inadvisable to leave a non-rerouting node in critical locations in the network. There are cases where a non-rerouting link can cause some problems with transmission being possible in one direction but not the other after a topology change. In Figure 9-10, N indicates nodes that have not initialized rerouting, and R indicates a node that has initialized rerouting.

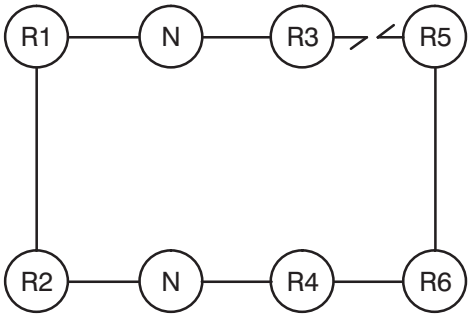


Figure 9-10. Rerouting Topology Considerations

If the link between rerouting nodes R3 and R5 fails, requests will be able to go from R5 to R1 but replies will not flow from R1 to R5. This is because routing updates generated by R5 indicating link R3 to R5 is down are rejected as bad requests by N nodes, thereby preventing R1 and R2 from updating their tables. Since R2 and R1 can not receive routing messages from the rest of the network, they effectively are non-rerouting nodes also. When the failure of R3 to R5 occurs, R5 reroutes through R6 to R4 to N and gets its messages to R1. R1, having not received routing information, continues to use the R1 to N to R3 path to get to R5.

Since higher-level software (for example, MA) will eventually prevent requests from going to R1, the Network Manager is cautioned to set up a mixed network carefully.

It is recommended to set up your network such that non-rerouting links connect end nodes. The example in Figure 9-11 below would work well and shows many sub-networks which would also work.

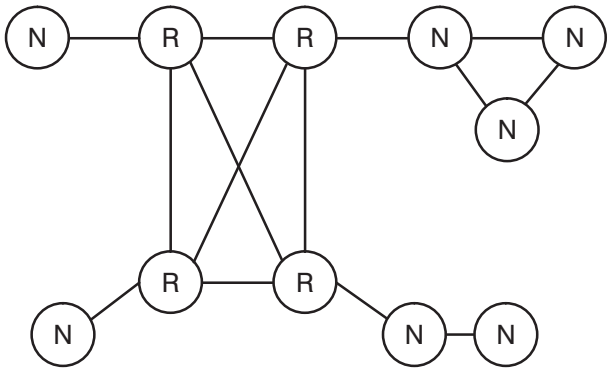


Figure 9-11. Rerouting Topology Considerations

Message Accounting

Message Accounting (MA) was developed to solve potential problems caused by the impossibility of any link-level protocol to guarantee a message is delivered to the ultimate destination, and that it is delivered only once. When messages are transmitted from the HP 1000 to the HDLC card, the card must transmit the message to the next node in the network. This is accomplished using the HDLC protocol which can assure delivery over the point-to-point link where the message becomes the responsibility of the next HP 1000 and HDLC card.

Using the HDLC card, MA also allows NS-ARPA to take advantage of the performance improvement achieved when I/O is considered complete when the DMA transfer to the card is complete. The transmission can be considered complete when the card has the message because MA has the capability of automatically retransmitting the messages if an acknowledgement is not received for a particular message. The additional performance is gained when a burst of traffic is processed by the HP 1000 and messages are transferred to several HDLC cards which then act as buffers for the CPU.

When the card has the message, the driver returns a successful completion and GRPM deallocates the message from SAM. In this situation, at any point along the path to the destination node there is the possibility that the message could be duplicated or lost with no regeneration of the message possible. Any link failure will cause the message to be lost and purged from the HDLC card. Rerouting occurs as a result of the failure, but the messages present only on the card attached to the failed link are lost.

Note MA is not used for NS Common Services or for messages to and from the HP 3000.

The MA software was developed to provide an end-to-end protocol where message regeneration capability is maintained. As opposed to the link-level protocol, the MA end-to-end protocol insures that the message travels from the origin node to the slave monitor class queue at the destination node. For replies, MA insures that the reply is received by GRPM at the origin node. No guarantee is made that the master program is still present in the system; MA guarantees only that the reply is received by the communications management level software at the origin node if a path between them still exists. If the master requestor has not been aborted, the message is queued to the master and the transaction complete. If the master has been aborted, the message is flushed and an error message is printed on the system console.

When a link fails and rerouting is in the system but not MA, there is no record of which messages were received at the destination and which were not. Through the use of MA sequence numbers in the messages, MA causes the retransmission of lost messages and insures that no duplicate messages are passed on to the user.

Lost Message Retransmission

When a message has not been acknowledged within a certain amount of time, a message is sent out to verify that the MA channel is still connected. The current MA sequence numbers and flags are returned as an acknowledgement that the channel is still working. The sending side then resends any messages that the other side has not received as indicated by the MA variables it just received.

By assigning sequence numbers to outgoing messages and by the receiving end keeping track of which sequence numbers have already been received from each possible destination node, MA will flush any duplicate messages.

Message Accounting Terminology

The end-to-end connection is referred to as a *channel*. This channel is a logical connection and has no relationship to the physical link, link characteristics, or path (in fact, with rerouting the path may vary).

The following three variables are found in every message.

- NS—the current MA send sequence number
- NR—the current MA receive sequence number
- NC—the current MA cancel flags

The following variables are kept in the MA table. There is a separate set for each MA destination node (each channel).

- VS—the next send sequence number
- VR—the next expected sequence number
- VA—the last sequence number acknowledged
- VF—the receive flags

- VC—the cancel flags
- VT1—the acknowledgement timer
- VT2—the idle timer
- VCC—the consecutive cancel counter.

Message Accounting Initial Connection

Before any valid sequence numbers are assigned, the two nodes across the channel must be logically connected (their sequence numbers must be synchronized). The two nodes may be anywhere in the network with any number of intervening nodes; MA is a logical channel between the end points.

Any traffic to a node whose connect state is down (logically disconnected) will cause the initialization handshake to be initiated to establish synchronization between the two nodes and their MA variables.

When `#MAAS` is called requesting an MA sequence number to a down node, it assigns -1, an invalid number, which causes the message to be dropped when it reaches its destination. This is done to ensure that no messages carry an MA sequence number unless it can be assured that they are valid. An initialization sequence is initiated between the two MA nodes and eventually the dropped message is sent via MA retries.

The write completion of the above message causes `#MAPP` to begin the initialization handshake. This handshake consists of an `INIT` message, answered by either an `IR` (Initialize Response), `RR` (Receiver Ready), or `CAN` (Cancel Outstanding Messages) message.

During a transition from a down link, once the connection has been completed, any non-delivered messages will be retransmitted via the MA retry mechanisms.

If the connect request is not successful, the state changes to down. All requests are returned to the master routine and all slave monitor replies are cleared from the holding class (`#MAHC`). If a message is received on a channel thought to be down, an idle message is generated from the receiving side which causes the processing described above to take place.

Message Accounting Sequence Number Assignment

An MA sequence number is obtained by a call to `#MAAS`. `#MAAS` checks the number of unacknowledged messages for this channel. If the number exceeds 15, a `DS08(1)` error indication is returned.

If the number does not exceed 15, the current values of `VS` (Next Send Sequence Number), `VR` (Next Receive), and `VC` (Cancel Flags) are set in the request/reply header. VS is then incremented and stored back in the MA table entry for this channel.

Message Accounting Receive Processing

Before GRPM passes incoming messages to the appropriate receiver, the messages are first filtered by #MAPP. Any invalid messages are dropped at this point by #MAPP.

Invalid messages are those with an NS number outside of the allowable window or those with an NS already received. This window arrangement is illustrated in Figure 9-12 below. (NS indicates the Sequence Number in received messages; VR is the Next Expected Sequence Number to receive; and W is the window value for incoming messages (constant 15).)

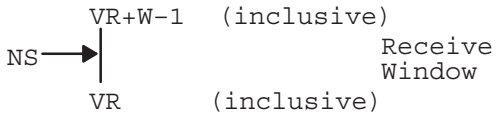


Figure 9-12. Message Accounting Receive Processing Window

If the message is acceptable, but NS is not equal to VR (i.e., a message received out of order), then VR is not incremented. Instead the appropriate bit is set in the receive flags, VF.

Each bit in VF represents a sequence number (NS) relative to VR; bit 0 corresponds to VR, bit 1 to VR+1, bit 2 to VR+2, etc.

When an NS is received equal to VR, VR is incremented and VF and VC are shifted over. The amount of the increment/shift is dependent upon the bits set in VF. These bits in VF indicate messages that have already been received out of sequence. The least significant bit of VF is tested and if a 1, VR is incremented and VF is shifted. The increment/shift is done until bit 0 of VF equals zero. Zeros are shifted into the high order of VF and VC. This procedure updates VR catching all messages received out of order.

The exception to the above processing is the receipt of a *cancel* or *no-response* message. A cancel message can be thought of as multiple messages. Because of this, the checks for *already received* are ignored. The VR/VF/VC increment/shift amount equals (NS-VR+1). The cancel message cancels all messages in the range VR, the expected sequence number, to NS, the sequence number contained in the cancel message. Those messages not received are considered cancelled.

A *no-response* message also does not follow the normal receive processing. If a *no-response* message is received, the channel becomes logically disconnected. This message is received when MATIC has attempted the maximum number of cancel messages and receives no acknowledge from the destination node. MATIC generates the no response message to GRPM and the MA routines. #MAPP changes the state to down. Timeouts (DS05) are sent to all waiting masters, and all replies for that particular channel are flushed from the MA holding queue (#MAHC). After any message is received, with the exception of idle messages (RR), the idle timer is set (VT2).

Message Accounting Message Acknowledgement

Incoming messages, regardless of their receive status, may carry acknowledgements for previous outgoing messages. The acknowledgement indicates that MA on the other side received a valid message.

Messages are acknowledged when the receive sequence number from the message, NR, falls within the following window. In Figure 9-13, NR indicates the current MA receive sequence number; VS

indicates the next send sequence number; and VA indicates the last sequence number acknowledged.

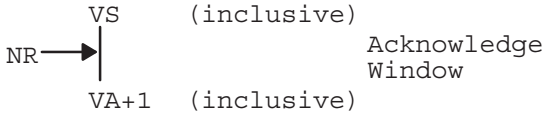


Figure 9-13. Message Accounting Acknowledgement Window

When a message is acknowledged, VA will advance by the amount (NR-VA). Note that several messages can be acknowledged with one received message. Each increment value of VA causes the cancel flags in the message (NC) to be checked for acknowledgement of a cancel message. The relationship between NR, VA, and NC is shown below:

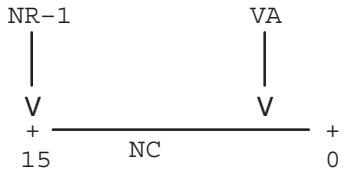


Figure 9-14. Message Accounting Incrementation

The cancel flags field (NC) incoming message provides the acknowledge that cancel messages have been received by the destination node which now allows the original message to be regenerated. Each bit in NC represents a relative position to NR; NR-1 corresponds to bit 15, NR-2 to bit 14, etc. Each bit set in NC between NR-1 and VA will cause a message retransmission attempt.

If the currently accessed bit in NC is not set, one of two things will happen. If the acknowledged message was a request, then the MA identifier in the TCB is cleared and the *request acknowledged* bit is set (bit 13 of TCB+1). This bit is only used in timeout processing to determine whether the request or the reply timed out. If the acknowledged message is a reply, then the reply is flushed from the MA holding queue (#MAHC).

Message Accounting Message Timeout

The MA watchdog timer, MATIC, runs every second. Its purpose is to check the status of the two timers (VT1 and VT2) for every MA channel. The acknowledge timer, VT1, indicates how long to wait for an acknowledge for the message just sent. The idle timer, VT2, indicates how long to wait in the destination node for return traffic to be generated toward the origin node before special MA traffic is generated to carry the message acknowledgement parameters back to the origin. Both timers are checked to see if they are running (i.e., non-zero). If so, each is decremented and checked for zero (a timeout).

If an acknowledgement is not received in time, VT1 times out. If this happens, VCC is incremented indicating another timeout. If this value is less than or equal to the MA retry limit (#MARL), a cancel message is sent across the channel. The NS for this message is VS-1 (i.e., the same as the last message sent). Acknowledgement of this cancel message causes one or more messages to be retransmitted.

If #MARL consecutive cancel messages have been sent without acknowledgement, MA concludes that the logical connection to this channel has been severed. MATIC informs the MA module in GRPM (#MAPP) of this by sending a no response message (NR). This is a local Class I/O message.

If VT2 times out, this indicates that an outbound message has not been sent in time in which to piggyback acknowledgements. Therefore, an idle message (RR) is sent to carry possible acknowledgement information (i.e., the NR and NC fields). The NS for this message is VS-1 (i.e., the same as the last message sent).

Message Accounting Message Retransmission

If a message does not receive an acknowledgement within the timeout interval (VT1), MA does *not* retransmit the message immediately. There are two reasons for this.

First, there are circumstances that could prevent a message from ever getting through. In this situation, the send sequence number NS used by this message would be held until the retry limit (#MARL) was exhausted. The acknowledgement window would thus eventually become frozen and prevent any further traffic. The cancel message allows new sequence numbers to be given to retransmitted messages, thus freeing up the movement of the sequence window.

Secondly, an acknowledgement timeout affects all messages to that channel. Immediate retransmission would cause a flurry of retransmissions including messages that might actually have been received. The cancel message allows selective retransmissions of only those messages that were lost.

When a cancel message is received those messages between NS and VR that have not been received are flagged as cancelled.

Note If the data message arrives late (for example, after the cancel message has been received), it is discarded as a duplicate since that particular NS has already been received. Each acknowledgement at the sending side checks for this cancel indication in NC. If the appropriate cancel bit is set, a retransmission attempt is made.

To retransmit a request, a message of length one (1) is sent to the appropriate master. The correct master is found by searching the TCBs for the correct sequence number. #MAST then carries out the retransmission (including assigning a new MA sequence number).

To retransmit a reply, the holding queue is searched (#MAHC) and the appropriate slave reply is resent (again with a new MA sequence number).

Message Accounting Disconnect

If no acknowledgement for a message has been received after #MARL cancel attempts, MATIC sends a no-response (NR) message to GRPM. This is a local Class I/O message.

When #MAPP receives this message, it begins a cleanup cycle on this channel. An error message is formatted using the last reported error. If there were no errors reported, then DS05(3) is used.

This error message is sent to all masters waiting for replies from this channel. These masters are identified by searching through the TCBs.

The MA holding class (#MAHC) is also searched. Any slave transmissions awaiting acknowledgement from this channel are discarded.

The channel state is set to down and the line-down count is incremented. Any inbound messages (with the exception of an initialize request) are discarded in this state. The channel will stay in this down state until either an INIT request is received or an attempt is made to send a message on this channel.

Message Accounting GRPM Interface

The cooperation of GRPM is essential for MA to function. There are two reasons for this. First, MA must act as a filter for message flow, discarding unwanted messages. GRPM must cooperate in this. Secondly, link level errors are determined, or at least reported, at the GRPM level. Since there is no way for MA to determine that a message is being returned with a link error, GRPM's cooperation is needed to flag these messages.

Before GRPM gives an inbound message to the appropriate user, it calls #MAPP to verify that the message is valid. All invalid messages and MA stream zero messages are discarded by #MAPP. Once the message is discarded, GRPM discontinues any processing of the message and proceeds to the next message.

If GRPM detects a link failure on a request, it attempts to return the message to the master with the appropriate error code. In order to indicate to MA that a link error has occurred, GRPM also flags the NR field (with -1). If MA receives a message with the *link error flag* set it records the error and discards the message. MA's retry mechanism eventually resends the original message. At that time, the link problem (such as in a busy condition) should have been resolved, or rerouting should have found an alternate path. Only after #MARL attempts will MA inform the master (using the last recorded error).

QUEUE

QUEUE is the NS-ARPA/1000 program scheduled by the communications driver when a new request arrives. (This driver could be the NS-ARPA driver for HDLC links, ID*66, or the customizing subroutine, CXL66, for X.25 links.) QUEUE provides a data buffer into which the data is read via a Class I/O call. The request and data lengths are first checked for validity and then the validity of the interrupt is checked. The interrupt must come from an initialized interface driven by ID*66 (or CXL66 for X.25 links) or it is considered a spurious interrupt and ignored. (Interrupts may also be ignored if the RTE interrupt table was generated improperly.)

Note LAN links do not use QUEUE. LAN links use the program READR instead of QUEUE; READR performs similar functions to QUEUE.

If the request length is outside the allowable range (0 or 7 to 63 inclusive), QUEUE calls the driver to send a STOP. If there is no problem, QUEUE issues a Class I/O read using GRPM's (RTE-RTE) or QUEX's (RTE-MPE) class number. (PROGL's class number is used if the request is a CBL download; VCPMN's class number is used if it is a Remote Front Panel request.)

If an immediate reject of the request occurs because there is insufficient SAM, QUEUE calls the driver to send a STOP. Both conditions result in a busy-reject being received on the master side. If there is sufficient SAM, the read is allowable and there are no active TCBs, QUEUE attempts to lock the quiescent resource number. If the system is going into quiescence, QUEUE is suspended here and incoming data buffers receive remote-busy rejects because the driver cannot schedule QUEUE. If the resource number is not locked, QUEUE terminates with the resource number clear.

The following error conditions can occur:

- If the interrupt is not from ID*66 (or CXL66 for X.25 links), it is ignored.
- If the interrupt is not from an initialized communication line, the driver is cleared.
- If the interrupt is from a DVT with no LU pointing to it, it is ignored.
- If the lengths of data or request buffers are out of range, a STOP is sent.
- If there is not enough SAM to hold the request and data, a STOP is sent.
- If GRPM's class is bad, a catastrophic error is reported (*nnnnnn* is the transaction sequence number and *oooooo* is the named register contents):

```
DS ERROR: PROG=QUEUE  SEQ #=nnnnnn P=oooooo A=oooooo B=oooooo
```

#SLAV

#SLAV is called by NS-ARPA/1000 slave monitors to send a reply (and data, if any), back to the origin node. It performs the following steps:

- Deallocates the slave TCB. If this fails, the error return is taken.
- Checks the no-reply bit in the request header. If set, #SLAV just returns to caller.
- Clears the MA send and receive sequence numbers and cancel flags.
- Enters the maximum hop count into the message header.
- Verifies that the reply length is in the range 13 to 37, inclusive, or, if not, returns a DS03 error.
- Clears the MA retry counter in the message appendage.

- Determines whether the message should be sent on a Router/1000 link or a non-Router/1000 link by calling #NRVS. If this search fails, a DS04 error is returned. If the message should be sent on a Router/1000 link, #NRVS converts the Router/1000 node address of the origin node to an output LU. If this conversion fails, a DS04 error is returned. (This can only happen if the NRV for the slave side does not match that of the rest of the network, allowing requests to arrive from a node which cannot be accessed or if the link fails while the reply is being formulated.)
- Calls #MAAS to assign an MA sequence number to the reply.
- Sends the reply to the driver, using GRPM's class if a Router/1000 link is used. If a non-Router/1000 link is used, performs a Class I/O write/read on LU 0 using IFPM's class number.
- Returns to the caller.

Note RFAM ignores all errors returned by #SLAV. PTOP slave routines convert them to -47.

Although transmission errors may occur in the reply, this information is not available to the slave. If all retry attempts fail, the reply is flushed and the master program eventually times out. When MA is present in the node, the slave replies are maintained in a holding queue until the reply is successfully received by the origin node.

#GET

#GET is called by routines waiting to receive requests (and possibly data) on their class numbers. #GET performs the following functions:

- issues a read (get) on the class number passed to it
- when the reply arrives, returns a DS03 if the request length exceeds the maximum allowed
- moves the request into the user's buffer (up to the number of words specified in the call)
- if the specified data length is exceeded, returns a DS03 error
- deallocates the class buffer
- returns the received request and data sizes in the A- and B-registers.

UPLIN

UPLIN protects the system users from waiting forever if one of the nodes in the network should fail or if transaction processing takes too long due to delays such as insufficient swap tracks, SAM, or lost messages. UPLIN also releases NS-ARPA resources allocated to programs which aborted or terminated without releasing them, such as TCBs, PNL entries, class buffers, and class numbers. UPLIN is scheduled for execution every five seconds and performs the following functions:

- Returns NS-ARPA resources owned by any program that has aborted or terminated since UPLIN last executed. This includes sockets, resource numbers, and sessions.

- Updates slave TCB timeout values. If a transaction has timed out, the TCB is returned to the available pool and, if the monitor is OPERM, CNSLM, or EXECW, the monitor is aborted. This allows other transactions to be processed, even if one is submitted which can never be satisfied, for whatever reason.

- After processing each slave TCB list, UPLIN checks to see if the corresponding monitor is dormant. If so, it is rescheduled and passed its class number. Thus, the system can recover if an operator should abort one of the monitors, although the transaction it was processing may be lost (the master will receive a timeout error). If a slave monitor's ID segment no longer exists, the class buffers and all slave TCBs are released. A one-word message is sent to RPCNV for each slave TCB if it is an HP 3000 slave TCB. Note that the class number itself is not released. This allows you to initialize NS-ARPA and to specify slave monitors which are not yet present, and then to download or `RP` them. UPLIN automatically enables them at its next execution cycle.

- Updates the master TCB timeout field. If a TCB times out or if it is dormant or if the bad flag in the TCB is set, the master class number and the TCB are released. If the program is in any wait state (program status not equal to 1), a null request is written to the requestor's class number.

- Scans the Process Number List and sends an HP 3000 `KILL` or HP 1000 logoff for any abandoned sessions (i.e., the program which signed on to the remote system did not send a `BYE` or logoff before terminating). The Process Number List entry is cleaned. The HP 1000 logoff is handled by calling `#UPSM`.

- Scans the HP 3000 LU Table. For each HP 3000 LU that went down, UPLIN cleans up the HP 3000 tables by: (1) deleting all associated Process Number List entries (PNL); (2) simulating a timeout on all associated master TCBs; (3) clearing all associated Transaction Status Table (TST) entries and corresponding slave TCB entries; and (4) if the LU is an X.25 LU, releasing the X.25 POOL LU and deleting the LU from the 3000 LU Table.

- Calls `#UPSM` to increment timeout counters in the Remote Session Pool (`#POOL`) entries.

- Reschedules INPRO, OUTPRO, GRPM, QCLM, LUMAP, MATIC, QUEZ, the IFPM timer, QUEX, RPCNV, or RQCNV if they were scheduled by NSINIT but are now dormant.

- If NSTRC is dormant or operator suspended but NSTRC's class and SAM buffers or resource numbers are allocated, both are deallocated.

Accessing Nodes with Session Monitor

Remote Session Monitor allows you to access DS/1000-IV nodes with Session Monitor within the session environment. All monitors attach and detach to specific sessions created either as default sessions or as specific sessions using calls to DLGON, DLGOF, or DLGNS for non-session access to a node.

To implement access to a remote Session Monitor node, NSINIT asks questions necessary to allocate a SMB block for tables used by Remote Session. The DS/1000-IV Compatible Services do not support local session; however, these services may be used to communicate with remote DS/1000-IV nodes that have Session Monitor.

Each master program has part of Remote Session appended to provide remote session ID for outgoing messages. When NSINIT is calculating the size of the SMB block that will be needed for all the NS-ARPA tables during NS-ARPA initialization, it asks for the maximum number of local sessions for remote nodes. The response to this question is multiplied by 7 words per session and returned to NSINIT as the size requirement for the Remote Session ID pool. NSINIT allocates the additional SMB block and initializes a RESA entry point #POOL with the base address of the Remote Session Identifier pool (table).

REMAT Session Module (#RMSM)

#RMSM is appended to REMAT and handles the expanded SW command by extracting the optional account names for Node1 and Node2 and generating logon requests to the module RSM at the remote nodes.

#RMSM processes the attach (AT) and detach (DE) commands and provides special processing for the exit (EX) command. (The AT and DE commands can only be used to attach and detach from accounts at remote Session Monitor nodes.) When the EX command is issued it is possible to have up to 16 sessions active, one at each of 16 different nodes. The EX processor generates a logoff request to each active remote session.

A network account table extension (NATX) is maintained by #RMSM and contains the account names for each of the 16 possible active accounts. When the SW command is issued without parameters, the account names at Node1 and Node2 are displayed from the NATX contained in #RMSM.

Distributed LOGON/LOGOFF (DLGON)

DLGON is appended to each master routine that calls DLGON, DLGOF, and DLGNS. This module generates requests to Remote Session Monitor (RSM) in the destination DS/1000-IV Session Monitor node to perform the logon/logoff. The messages go out on stream 7 which is normally used by OPERM. To distinguish the logon/logoff requests from regular OPERM requests an illegal operation code of XX is used. GRPM notes the absence of a destination session ID and passes the request to RSM rather than OPERM. RSM checks the command code, finds the XX and processes the request as a logon/logoff request. OPERM never receives these special requests.

Network Account Table (#NAT)

#NAT maintains the Network Account Table (NAT). The NAT is used to keep track of the 16 remote session ID's, each at a different node, that the program can have active at any one time. Each entry in the NAT consists of

- Word 1—the Router/1000 node address.
- Word 2—the Remote Session ID, Remote Sub-level and ownership flags.
- Word 3—the address of user name in NATX. (Used only for REMAT.)
- Word 4—the EXECW Sequence Number.
- Word 5—the PNL entry pointer.

Word 3 is used by REMAT/#RMSM to keep track of the 16 possible account names associated with the 16 different remote sessions REMAT may be attached to at any one time.

Word 4, the EXECW Sequence number, is used when the master program is memory-resident, scheduled from a remote node via EXECW, and has to insure that the NAT gets cleared between each execution of the program.

Master Program Session Module (#MSSM)

The main function of #MSSM is to provide remote session ID numbers when called by #MAST, which attempts to send a message. The Remote session ID's are stored in the NAT if they exist. If no explicit logon has been performed, #MSSM searches for a father program that may have established a session at the destination node already. If a father is found that has an active session at the destination, that session ID is used. If no father is found, #MSSM checks if any other program scheduled from a session at this terminal has a session at the destination node and will use that session ID. Failing that, #MSSM checks if the program or an ancestor was scheduled from a session at the destination node and will use that session ID. If no active session can be found the first message sent into the destination node is routed through RSM where a default session is created. The session ID is entered into the Request header and the request required to the appropriate monitor, when the reply returns #MAST calls #MSSM who enters the new remote session ID into the NAT. Any further requests generated by this master will use the already-established session ID and bypass RSM processing in the destination node.

Router/1000 Message Header

The Router/1000 message header contains the session ID word at offset #SID. The left and right bytes are source LU and destination session ID, respectively, if any.

Session ID 253 represents the permanent old node default session. Session ID 254 represents special non-session access to Session Monitor node.

Program-to-Program Communication (PTOPM)

PTOPM is the monitor which receives all PTOP communications messages on the slave side. It services POPEN, PWRIT, PREAD, PCONT, PCLOS, FINIS, and requests from the REMAT commands SL (Slave List) and SO (Slave Off). The REMAT commands SL and SO, and PCLOS and FINIS, are processed entirely by PTOPM. POPEN, PWRIT, PREAD, and PCONT are passed to the slave program, where the slave routines GET, ACEPT, and REJCT process them.

POPEN Processing

For POPEN, PTOPM checks to see if the slave program exists. If it does not, an error code of -41 is returned and PTOPM processes the next request.

#SCSM is called to perform optional cloning and remote session functions. If the slave program exists, it is scheduled regardless of whether or not it is already in PTOPM's table. This ensures that a POPEN call will always operate without error regardless whether the slave program has previously been aborted or terminated without a PCLOS or a FINIS call.

PTOPM maintains a table of the slave programs currently open. A maximum of 20 are allowed. If this table is full and a new POPEN request arrives for a new slave, an error code of -42 is returned. If the program named already exists in this list, then that entry and class number will be used, and the request is threaded onto the program's class number and PTOPM processes the next request. If the program does not already exist, a new entry is made and a new class number is obtained. If there are no available class numbers or SAM is not available, an error code of -42 is returned to the master. If both exist, the program is scheduled and passed the newly-obtained class number. If an error occurs here, an error code of -48 is returned. If an error does not occur, the request is threaded onto the class number and PTOPM goes back to get the next request.

Certain programming conventions must be followed if the slave program has more than one master. These conventions are explained in detail in the the PTOP section of the *NS-ARPA/1000 DS/1000-IV Compatible Services Reference Manual*.

PREAD, PWRIT, and PCONT Processing

PREAD, PWRIT, and PCONT calls are processed similarly. PTOPM's internal table is searched for the slave program utilizing the ID segment address from the master's PCB. If this has been modified, or for any reason the program is not found in the table (as could happen, for instance, if an SO command was issued to terminate the slave or if PTOPM was re-loaded online), an error code of -44 is returned. The class number from the PCB is compared to PTOPM's internal table entries and error -103 is returned if there is no match. If there is a match, the request is threaded onto the slave program's class number. As with POPEN, if errors occur, an error code of -48 is returned.

If a slave program has too many requests queued onto its class, the new request is returned to the master with an error code of -58. If the slave program is dormant and not in the time list, the request is returned with an error code of -45.

PREAD Processing. PREAD sends a message to the slave containing the data buffer length. This message is retrieved by the GET subroutine and returned to the caller. When the ACEPT routine is called by the user, the user's data is passed back to the master program by building a communications message which is rethreaded to GRPM's class number.

PNRPY Processing

When the no-reply option is active, any PWRIT, PCONT, or PCLOS call has the no-reply bit set in its message header.

On the master side, a TCB is allocated with timeout equal to the *tto* parameter in the PNRPY call. When the TCB times out, UPLIN deallocates the TCB, but does not generate a zero-length record or report an error as is usually done on TCB timeouts.

On the slave side, the slave must still process the master calls as usual by doing an ACEPT or REJCT. #SLAV, which is appended to ACEPT and REJCT, deallocates the slave TCB and checks the no-reply bit. When the no-reply bit is set, the reply message is not sent.

PCLOS Processing

PCLOS aborts the slave and PTOPM generates a reply to the master request. Resources allocated currently by the slave are lost to the system. PCLOS should not be used if the slave has other masters or maintains data tables which should not be destroyed.

GET Processing

The class number specified in the parameter *clas* is used to get the master request and the slave program is provided with the function code of the master request in *func*. On PWRIT, GET moves the request buffer into the user space if the buffer is specified in the GET call.

ACEPT Processing

ACEPT implies that the master request is valid. On PWRIT, the request data is moved into the user space unless it has been previously obtained in the GET call. On PREAD, the read data is included with the reply and SAM is deallocated. On all master calls (PREAD, PWRIT, PCONT, POPEN), the reply bit for accept is set and *tag* is included. The driver is called through #SLAV and clean-up is done for the return to the GET call.

REJCT Processing

REJCT implies that the master request is invalid. A no data reply is created, *tag* is included, and the reply bit is set for reject. The driver is called through #SLAV and clean-up is done for the return to the GET call.

FINIS Processing

If a slave program calls FINIS, the request is serviced by PTOPM. If the slave has other requests it has not yet received, these are cleared out. The class number is released. The program's entry in the table of active programs is cleared out. Note that the slave is not terminated by the FINIS call.

Slave List Processing

The Slave List (SL) is a table used to move the names of all slave programs into the reply buffer, which is sent back to REMAT. This is the only request to PTOPM which generates a non-zero data length reply. The order of listing programs has no relationship to the order in which they were opened.

Slave Off Processing

If the Slave Off (SO) request is used to clear all slaves, the internal table is used to generate PCLOS processing for each program in the list. If the Slave Off command is not used, a PCLOS is only generated for the named program.

Remote File Access Monitor (RFAM)

RFAM, the Remote File Access Monitor, receives all requests built by the RFA master routines (for example, DOPEN, DCLOS, DWRIT, DREAD) and makes FMP subroutine calls on behalf of the master programs. It requires storage for one 144-word Data Control Block (DCB) for each file opened by any master program. RFAM can handle multiple files at the same time, storing DCBs in the remaining unused memory in its partition.

When each request arrives, a copy of the four-word pseudo-DCB stored in the user's program is used to locate the DCB RFAM stores. RFAM handles this DCB as follows:

- DOPEN, DPURG, DNAME, and DCRET are executed by RFAM without any information from the four-word pseudo-DCB. The master subroutine copies information from RFAM's reply into the user's four-word pseudo-DCB. If you have a file open and then call any of these routines with the same pseudo-DCB, RFAM maintains the DCB for the original file and allocates another DCB entry. Although RFAM still has both files open, you have lost the ability to access the first file. Therefore, you must explicitly close all files that are no longer required. REMAT provides a command, FL, that clears out files which have not been closed. (Notice that the analogous FMP subroutine calls are slightly different in this respect. They will automatically close any open file before executing OPEN, PURGE, RNAME, or CREAT calls.)
- All other master RFA routines send a copy of the pseudo-DCB to RFAM which uses the information to locate the DCB, but the reply does not cause the pseudo-DCB to be modified.
- DPURG and DNAME do not use or modify the pseudo-DCB.
- Extended file calls are implemented by providing alternate entry points (DXCRE, DXCLO, DXREA, DXAPO, DXPOS, DXLOC) in the corresponding file access routine. The double word parameters are used where necessary and the function codes used by RFAM at the destination node are 14 through 20 rather than 0 through 12 used by the regular file calls.

DEXEC

The DEXEC utility subroutine is included in user programs which request remote EXEC functions. When the destination node is not local, DEXEC builds a request buffer (and optionally data) for transmission to the remote computer. When the request arrives, it is routed by GRPM to EXECM or, in the case of a remote schedule-with-wait, to EXECW. When the destination node is local, however, the EXEC call is made by DEXEC to improve efficiency. In this case, however, no buffer-length check is made. Programs which make use of this feature will run only when the destination node is local if their buffer lengths exceed 512 words. The special write/read DEXEC call (*code* = 1, bit number 11 of the control word set), is implemented by building a request buffer and sending it to EXECM.

Driver ID*66

ID*66 is the driver for the HDLC PSI cards for DS/1000-IV Compatible Services (RTE-RTE) communications, and the Bisync PSI cards for DS/1000-IV Compatible Services (RTE-MPE) communications.

Caution Hewlett-Packard does not support direct calls by user software to any NS-ARPA/1000 communication driver. Severe disruption of network communication may result. For Hewlett-Packard's support policy for the use of the 12079A LAN/1000 Direct Driver Access (DDA) product with NS-ARPA/1000, refer to the note at the front of this manual.

HDLC Link Connect Processing

On boot-up, ID*66 is entered when NSINIT begins enabling LUs. NSINIT sends a connect request to the driver and the driver disables interrupts from the card. The card responds to the disabled interrupts request and sends an indication to the driver that it has just powered up. The driver responds to this indication by sending a power-up acknowledge command to the card followed by a copy of the DVT timeout value. It then returns to NSINIT's connect request. The connect command is issued to the card followed by a buffer frame size request. The card returns the frame size which is based on a switch setting on the card. A switch indicates either 128-byte frames or 1024-byte frames. The driver stores the frame size in the DVT. The driver then enables unsolicited interrupts on the card and returns a completion to NSINIT.

When the connect command is issued to the card, the card begins sending SARM (Set Asynchronous Receive Mode) frames to the other end of the link. If the other end is initialized, it will be waiting for the SARM and respond with a UA (Unnumbered Acknowledge) frame. Once both cards on the link have exchanged SARM/UA's, the link is considered connected. The cards generate a connect interrupt to their respective drivers.

The driver formulates a zero-length message and schedules QUEUE to pass the message back to GRPM. GRPM detects the zero-length message and calls the rerouting routines to process the newly-available link. This completes link connection processing.

HDLC Retry Processing

Retry processing is performed by the HDLC card. The retry limit is determined by the formula:

```
Line timeout / T1 timeout = Retry count
```

The timer T1 defines the period of time the sending card will wait for acknowledgement before retransmitting the frame. The default values of T1 are used when the line timeout is set to zero. The values are shown in Table 9-2.

Table 9-2. T1 Default Values

Line Speed	T1 (large frame)	T1 (small frame)	Retries
230K bps	.15 sec	.10 sec	10
56K bps	.50 sec	.15 sec	10
19.2K bps	1.50 sec	.20 sec	10
9600 bps	3.00 sec	.40 sec	10
4800 bps	6.00 sec	.70 sec	10
2400 bps	12.00 sec	1.40 sec	10
1200 bps	24.00 sec	2.80 sec	10
300 bps	96.00 sec	11.20 sec	10

The maximum number of retries used by the HDLC card is 10. The card determines the timeout value based on the setting of switches on the card indicating the line speed. When the line timeout value in the DVT is non-zero the card performs the above calculation to make the determination of retries. Whenever the result of the calculation is less than 1 or equal to zero, the default of 10 retries is used. A sample calculation where the DVT timeout is 200 milliseconds and the T1 timeout is 100 milliseconds (.10 seconds) the calculation would be:

```
200 / 100 = 2 retries
```

Control over the retry count is via the DVT timeout value.

Note When using modems that supply the clock, the line speed must be set to a value equal to or less than the modem's speed. For situations where there are long transmission delays (for example, satellite communication) the speed that the card is set to may need to be slower than the modem's so that the T1 time is long enough to compensate for the transmission delay.

HDLC Link Down Processing

When the HDLC card exceeds the maximum retry limit on a frame, the link is declared down and the driver is interrupted and informed of the failure. A message with a length of one is returned to GRPM to inform rerouting of the down link. When the link failure occurs the card resets itself, discarding all other pending frames. Once rerouting has changed the path to the destination, these messages are regenerated by the MA routines if the feature is generated in.

Read and Write Processing

When a message is queued to the driver, the driver examines the size of the message to determine how many frames will be required on the card to transmit the message. The driver then asks the card for the necessary number of frames and then sends the message to the card one frame at a time. If the complete message will not fit on the card the driver sends as many as the card has room for, sets a time out, and suspends until the card has room for more frames.

The first frame of the message includes a Start of Message flag and two length words indicating the lengths of the request header and data portion of the message.

When the first frame is received by the card on the receiving end the card interrupts its driver and indicates it has received a frame with the start of message bit set. The receiving driver then reads the two length words from the frame and schedules QUEUE to provide a class read buffer to receive the message. QUEUE issues a Class I/O read request using GRPM's class. (PROGL's class is used if this is a cold-load request; VCPMN's is used if this is a remote front panel request; and QUEX's class is used on a Bisync link to an HP 3000.) The driver begins reading the frames off the card as they become available. When the complete message is put back together again the read completion status is returned to GRPM.

The HDLC protocol is explained in the hardware manuals describing the HDLC cards.

DSLIN Processing

If you run DSLIN in *primary mode* (initiate communication with the HP 3000), DSLIN sends an initialization request to the HP 3000. When the HP 3000 sends an initialization reply back, the reply is processed by QUEX. QUEX then sets the `BUFFER SIZE` field in the HP 3000 LU Table and sets the Continuation Record (CR) field to the appropriate value: 0 if the HP 3000 does not support Continuation Records, 1 if it does.

If you run DSLIN in *secondary mode* (wait for the HP 3000 to initiate communication), the HP 1000 waits for the DSLINE command to be issued at the HP 3000. DSLINE causes an initialization request to be sent to the HP 1000. QUEX receives the request and sends an initialization reply to the HP 3000. QUEX then updates the HP 3000 LU Table as above.

For more information on DSLIN operation, refer to the *NS-ARPA/1000 DS/1000-IV Compatible Services Reference Manual* and to the DS/1000-IV documentation.

DS/1000-IV Compatible Services Communication (RTE-MPE)

Communication between an HP 1000 and an HP 3000 using the DS/1000-IV Compatible Services (RTE-MPE) is similar in many ways to communication using DS/1000-IV Compatible Services (RTE-RTE). A block diagram of the software organization is shown in Figure 9-15. Master-side communication is shown on the left. Slave-side communication paths are shown on the right. The block in the center, *Communications Access Software*, represents modules which perform I/O to the HP 3000, including the driver.

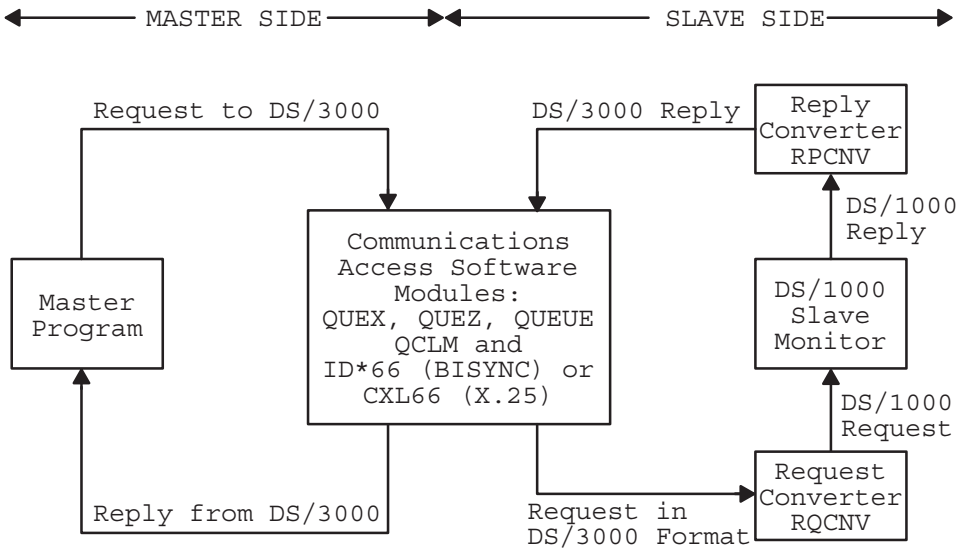


Figure 9-15. Overview of NS-ARPA/1000 to NS/3000 Communications

The DS/1000-IV Compatible Services (RTE-MPE) can be used over modem, direct connect, or connections using Bisync or X.25 protocols. Programmable Serial Interface (PSI) cards are used for MODEM and Direct Connect Bisync as well as X.25. Table 9-3 shows which PSI cards are used for the various connections.

Table 9-3. PSI Communication Interfaces

Connection	Card
Bisync Modem	12073A
Bisync Direct Connect	12082A
X.25 Modem	12075A

Bisync links to HP 3000s use driver ID*66 (the same driver used for RTE-RTE HDLC communication). X.25 links use the customizing subroutine `CXL66` attached to the X.25 driver `DDX00.REL`, just as in RTE-RTE X.25 communication. Multiple Bisync links can co-exist with X.25 links.

Master programs write requests directly to the communications LU, then wait for a reply on their own I/O class. When the reply comes back from the HP 3000, QUEUE receives it and rethreads it to QUEX's class number. The reply is then matched to the proper TCB using the sequence number and back to the master program using the class number from the TCB. All requests and replies are in DS/3000 format.

When a request for a slave monitor arrives from the HP 3000, it is in DS/3000 format. QUEUE passes the request to QUEX, which passes it to the request converter, RQCNV, which converts it to DS/1000-IV format and passes it to the proper slave monitor. When the monitor completes its processing, it checks the 3000 bit in the stream word and, when it is set, passes the reply to RPCNV, the reply converter. RPCNV converts the reply to DS/3000 format and sends it to the HP 3000.

DS/3000 Messages

All communication messages to and from an HP 3000 are in DS/3000 format. Each message has a class and stream associated with it; most of these are equivalent to a DS/1000-IV stream.

DS/3000's message class 0 is a special case. Stream 20 on class 0, the initialization request, establishes the buffer size to be used over the line. Stream 21, the termination request, indicates that the sending computer no longer has any master users on the link. If the receiving computer also has no master users, communication is terminated.

Because various resources must be allocated at initialization of the link, both computers must agree on the buffer size to be used. On the HP 1000, the default size is 1024 words, but this value may be overidden by running DSLIN. (Values must be between 304 and 4096 words and must be multiples of 16. Values larger than 1072 are provided for backward compatibility only.) The HP 3000 gets its buffer size from its I/O configuration or from a parameter in the `DSLINE` system command. The value must be less than 4096 and a multiple of 16.

The side sending the initialization request puts its buffer size divided by 16 in the fourth word of the initialization request. If the other computer can handle that size, it echoes it in the reply. Otherwise it puts its buffer size divided by 16 in the reply and the first system must not exceed it.

DS/3000 messages sent over Bisync can be larger than the communications buffer. For larger messages, continuation records must be used. Over X.25, the X.25 software takes care of segmenting the messages and the buffer library controls only the total amount of data which can be sent to a HP 1000 slave program. Figure 9-16 shows an example of a `PREAD` request which generates 350 words of data and is transmitted over a Bisync line configured for 304 words.

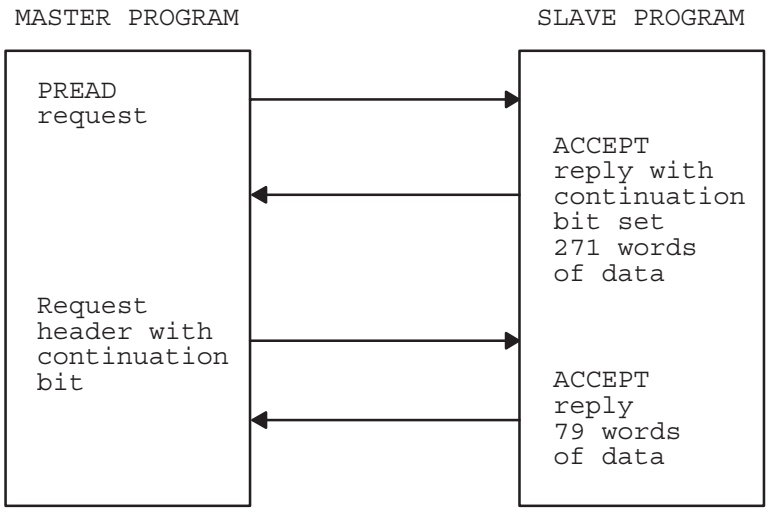


Figure 9-16. Continuation Data Buffer

In this example, the master sends a PREAD request containing DS/3000 header and appendage. The slave sends an ACCEPT reply with header, appendage, and data. Because the entire message cannot fit in the 304-word line buffer, the continuation bit in the stream word of the header is set. When the master receives the reply, it knows to signal the slave for more data by sending just an 8-word header ACCEPT request with the continuation bit set. After the slave receives this continuation request, it sends the remainder of the data in an ACCEPT reply. The master knows the transaction is complete when the continuation bit is not set.

DS/3000 Bisync Protocol

DS/3000 uses a subset of International Business Machines' Binary Synchronous Communication (Bisync or BSC) for its communications protocol.

Figure 9-17 shows a typical DS/3000 exchange. At the beginning, the link between the computers is idle (no transmissions in progress from either side) and the computer on the left has a message to send. That computer bids for the line by writing the Enquiry (ENQ) character. The computer on the right signals that it is ready to receive by responding with an Acknowledgement (ACK0). From that point, the computers exchange messages in *write conversational* mode: the computers send text to one another without intervening acknowledgement characters. (Each block contains the standard low level Bisync protocol characters, STX and ETX, and the CRC parity check word.)

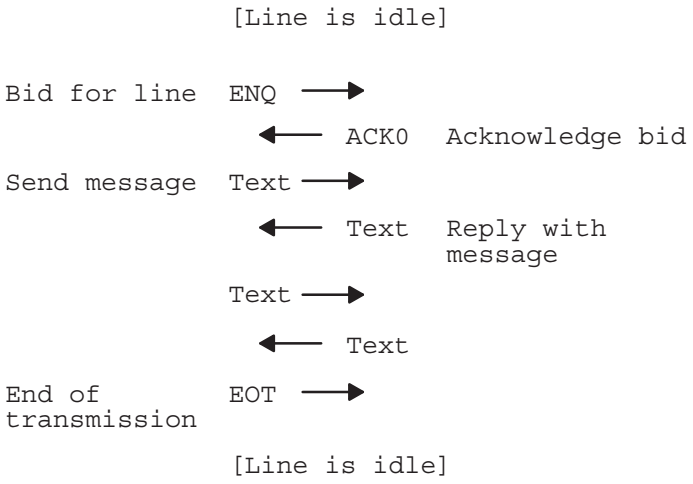


Figure 9-17. DS/3000 Bisync Exchange

If a computer does not have a DS/3000 message to send, its text consists of a null message: one word with all bits set on. When a computer has nothing to send after receiving a null message in response to a null message, it terminates the text exchange by sending an End of Transmission (EOT) character. (The side sending the EOT does not have to be the same side that sent the ENQ.)

ID*66, CXL66 and PSI Links

This subsection describes the modules which go in the Communications Access Software block in Figure 9-18 for Programmable Serial Interface (PSI) links to DS/3000. Included are the monitors QUEX and QUEZ, and the driver ID*66. Also included is the customizing subroutine for X.25 links, CXL66. The relationship between these modules is shown in Figure 9-18.

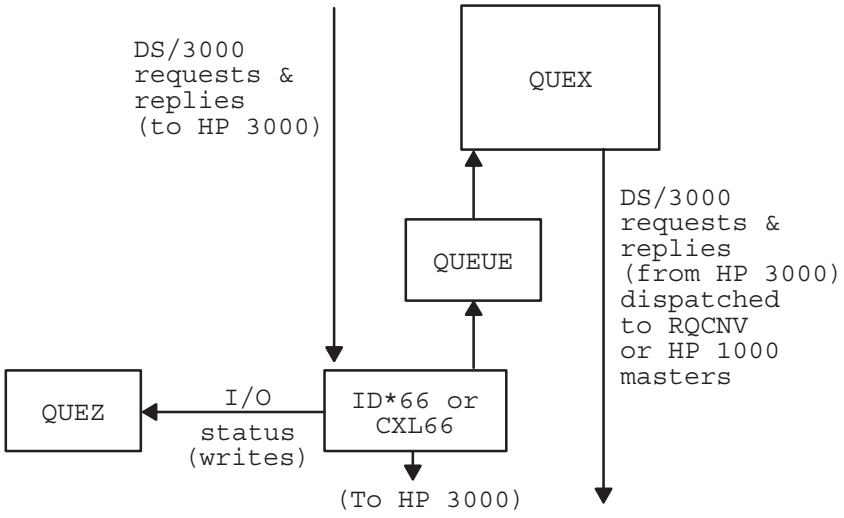


Figure 9-18. Communications Access Software for PSI

ID*66 I/O Requests for Bisync

Caution Hewlett-Packard does not support direct calls by user software to ID*66. Severe disruption of network communication may result.

The following functions are used by the Communications Access Software to communicate with the Bisync firmware on the board:

- *Initialize*. Tells the card what block size will be used in transactions with the card. The card may not be able to handle certain block sizes and can return a block size parameter to override the requested block size. The remote and local ID sequences are sent to the card. These are used to validate connect requests either initiated from the HP 1000 or HP 3000. Once the initialize parameters are sent to the card, DSLIN issues the message `3000 LINK READY FOR CONNECTION` and then proceeds to issue the primary connect command.
- *Primary Connect*. The primary connect request tells the card to initiate the connect sequence with the remote HP 3000.
- *Secondary Connect*. This command tells the card to wait to receive a connect sequence initiated by the HP 3000.
- *Get Parameters*. This command retrieves the board type, firmware revision date code, dip switch settings, diagnostic hood indicator, data block size, and so forth. The block size is used by QUEX to determine if the size asked for is larger than the block size the card will accept. In this case the maximum block size is defined by the card.
- *Read*. Issued by QUEUE to retrieve incoming HP 3000 messages.
- *Write*. The write command is used to send a block of data to the card.

The RTE-MPE low-level communications is similar to the RTE-RTE modules; the driver schedules QUEUE, then QUEUE performs a class I/O read on QUEX's (rather than GRPM's) class.

For incoming messages, the card does not know where the break between the header/appendage and the data occurs. QUEUE performs a Class I/O read of two buffers onto QUEX's class. (The Z-buffer at this point is only large enough to hold the local appendage.) The driver uses the word counts in the HP 3000 message header to modify the class I/O header in SAM to make it look like a Z-buffer was passed containing the data and local appendage.

QUEX and QUEZ for the PSI Link

The interfaces necessary to implement the protocols for the PSI versions of QUEX and QUEZ are handled by either the Bisync firmware on the PSI card or the X.25 software.

Because there can be any number of PSI links, QUEX cannot initialize lines. When NSINIT is run to initialize NS-ARPA, all Bisync links in the 3000 LU Table are brought up in secondary mode (waiting for call). The program DSLIN performs re-initialization as primary or secondary for a

given Bisync line. (Copies of DSLIN may be run by different users for different LUs.) Refer to the *NS-ARPA/1000 DS/1000-IV Compatible Services Reference Manual* for information on running DSLIN.

No NSINIT or DSLIN initialization is needed for X.25 lines connected to an HP 3000. These LUs are placed in the 3000 LU Table as they are allocated, and are deleted from the table by UPLIN when they are deallocated. (X.25 is described in detail below.)

Messages bound for an HP 3000 over PSI links are not combined with other DS/3000 messages into a single communications buffer. All writes to an HP 3000 are made directly from D3KMS (for masters) or RPCNV (for slaves) to the communications LU via class I/O. QUEX does no processing for outgoing messages. QUEUE reads incoming messages, passes them to QUEX via its class number and QUEX dispatches them to the proper master or RQCNV. The dispatching is done via a class rethread.

QUEZ waits on its I/O class number and checks the I/O status for outbound messages. (It also removes the class buffers left over from the write/read.) If the error bit is set in the I/O status, QUEZ marks the LU down in the 3000 LU Table, passes an error message to QCLM, and reinitializes the card to be in secondary mode.

Because QUEZ needs to know which LU the status word is associated with, a word containing the LU number is sent at the beginning of each outbound message. The driver removes this word (and decrements the length) before the message is written to the HP 3000.

If logging is enabled, this LU word is written to the log device along with the message. On inbound messages, the word preceding the logged message is set to the negative of the LU.

X.25 Links

An HP 1000 master program initiates communication with an HP 3000 over an X.25 link by calling `HELLO`. The `HELLO` subroutine calls the DS/1000-IV library $D3X25 which contains a reference to the X.25 routine `ALTAD`. (`ALTAD` is contained in the X.25 library.) `ALTAD` allocates a switched virtual circuit (SVC), and $D3X25 places the POOL LU associated with the SVC in the 3000 LU Table via a call to `D$PUT`. $D3X25 does a connect on the SVC and sends an HP 3000 initialization message to the HP 3000. The POOL LU placed in the 3000 LU Table is then marked up by the routine `D$UP!`.

If an SVC goes down before it is marked up in the 3000 LU Table by `D$UP!`, `HELLO` calls `RELSX` to release the circuit. `RELSX` releases the POOL LU by calling the X.25 routine `RPOOL` contained in the X.25 library.

The HP 1000 terminates communication by calling `BYE`. The `BYE` routine causes an HP 3000 `BYE` to be issued to the HP 3000 and marks the X.25 LU down in the 3000 LU Table. UPLIN checks the 3000 LU Table for down X.25 LUs, deletes them, and calls `RELSX`. `RELSX` calls the X.25 routine `RPOOL`, which returns the X.25 SVC to the pool.

Note If the X.25 connection consists of a Packet Switching Network (PSN) or a genuine modem link (not hardwired with baseband or short haul modems), SVCs are not automatically reconnected if the network or modem(s) goes down and is then restarted. You must use the DI command in DSMOD to disable the LU associated with the circuit and return it to the POOL. Do *not* use the DSMOD /L command.

When an HP 3000 initiates communication over an X.25 link, a line open request arrives at the HP 1000. The X.25 software allocates a POOL LU on behalf of the HP 1000 slave, and the request is routed to the DS/1000-IV request converter, RQCNV. RQCNV checks the HP 1000's Transaction Status Table (TST) for an entry (i.e., a new request from the HP 3000). If there is no TST entry, RQCNV checks the 3000 LU Table for an X.25 LU. If there is an X.25 LU in the table, it is known that the HP 3000 request is not a new X.25 request and RQCNV will not allocate an LU. If there is not an X.25 LU in the table, RQCNV calls D$PUT which places the POOL LU into the 3000 LU Table. $D3X25 does a connect on the SVC associated with the LU and marks the LU up via a call to the routine D$UP!.

When it is finished, the HP 3000 disconnects the X.25 SVC, and the DS/1000-IV monitor QUEZ receives a link down indication at the HP 1000. QUEZ checks the 3000 LU Table for an X.25 LU and, if it finds one, marks it down. UPLIN checks the table for a down LU, deletes the entry from the table, and calls RELSX. RELSX calls the X.25 routine RPOOL which returns the X.25 SVC to the POOL.

Note Because UPLIN only runs every 5 seconds, there may be a delay between the time an X.25 SVC is disconnected and the time the LU is removed from the 3000 LU Table. If there are a limited number of POOL LUs, this delay may temporarily prevent a program from obtaining a POOL LU.

RTE to MPE Master Side Communication

Figure 9-19 is a block diagram of the software which delivers DS/3000 messages from master users at the HP 1000 to the HP 3000.

HELLO, BYE, and Master Subroutines

MPE requires each of its users to be operating under a specific session. Remote users from the HP 1000 establish their session through the subroutine HELLO, either by calling it directly from their own program, or by using the HELLO command in RMOTE.

If the HP 3000 LU parameter in HELLO is an integer less than 256, it is treated as the LU number of a Bisync connection. Otherwise it is treated as the ASCII encoding of an X.25 address. This

X.25 ASCII string may contain leading ASCII characters and up to 15 digits. The total length of the string cannot exceed 28 characters. It is terminated by any non-digit.

When an X.25 address is passed, HELLO converts the digits to Binary Coded Decimal format and calls the ALTAD subroutine (provided in the X.25 library) to allocate a switched virtual circuit (SVC) POOL LU connected to the X.25 address. HELLO uses routines in the DS library $D3X25 to allocate a POOL LU, put it in the 3000 LU Table, and mark it up. After the LU is placed in the 3000 LU table and marked up, HELLO opens the line with wait by making a Primary Connect call. Processing continues with the normal HELLO request.

If the account and (optionally) passwords are correctly provided, MPE assigns a session number, which HELLO returns as an integer parameter.

It is possible for one program to establish a session, then allow a son program to share the session. The father program must schedule its son with wait because two programs cannot access the same MPE session simultaneously. The son program gains access to the session by calling PRCNM. PRCNM is explained in the Utilities section of the *NS-ARPA/1000 DS/1000-IV Compatible Services Reference Manual*. The DS/1000-IV compatible software will kill the session if the program which issues the HELLO goes dormant before calling BYE to log off.

Once the session is established, a program can make PTOP or DS/3000 RFA subroutine calls. In PTOP calls, the HP 3000 is addressed by the negative of its logical unit number. No address is necessary for RFA calls.

HP 1000 users terminate their HP 3000 session by calling the subroutine BYE. After closing the session, BYE sends a DS/3000 termination request if no other HP 1000 programs are using the line. (For X.25, that will always be the case.) BYE does not wait for a reply to the termination, but lets QUEX close the line when the reply is received. For X.25 connections, BYE marks the POOL LU down in the 3000 LU table. UPLIN deletes the down LU from the table and the X.25 routine RPOOL returns the X.25 SVC to the pool.

If the other side is using the line, it rejects the request and the line stays up. Otherwise the line is disconnected (a DLE EOT is sent).

When termination is successful on a Bisync line, QUEX sets the buffer size in the 3000 LU Table to -1 to indicate the line is down.

If the PTOP POPEN call is made using an X.25 pool LU, the PTOP software can make use of the routine LU3K to find out which LU was assigned to the master program by the X.25 software.

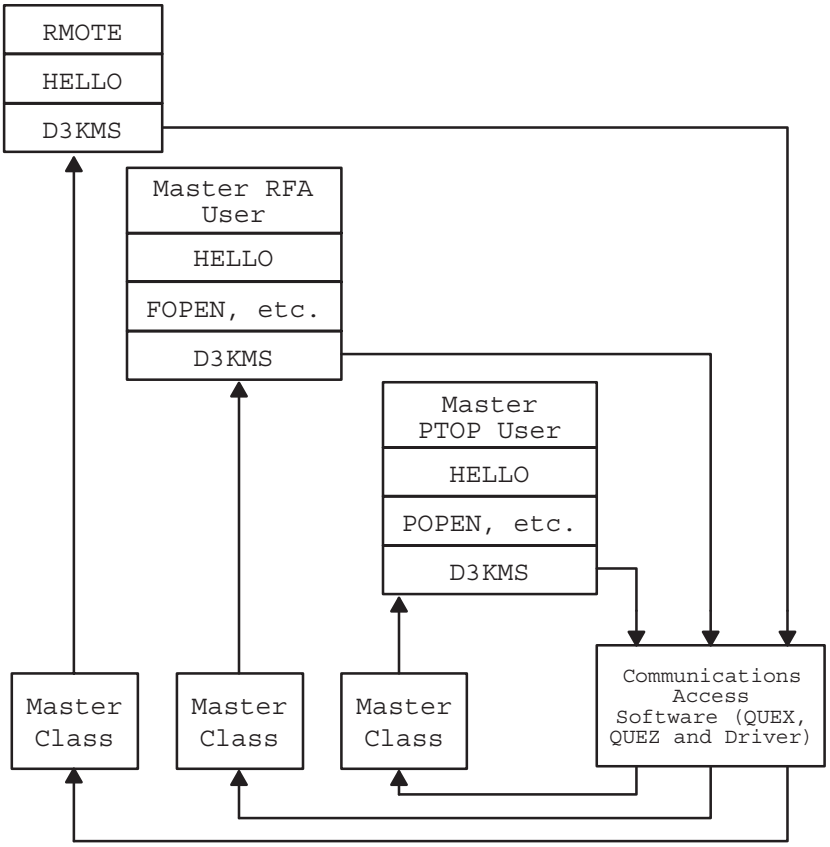


Figure 9-19. DS/1000-IV to DS/3000 Master Side Modules

D3KMS Subroutine

The master subroutine for HP 3000 master requests is D3KMS, which is similar to #MAST. D3KMS obtains a class number for the reply and sends the request by writing the request (and data, if any) to the HP 3000. D3KMS then makes a Class I/O get call, which suspends the user program until the reply arrives or the request times out.

D3KMS performs the following functions not found in #MAST:

- Handles continuation requests/replies for $STDLIST/$STDIN requests from the HP 3000. (Continuation is not needed for X.25 links.) All data to and from the HP 3000 is blocked into records whose length depends upon the buffer length recorded in the 3000 LU Table. At link initialization time, a bit is set in the initialization request and reply to indicate each side can buffer continuation records for these messages. Buffering is done at the end of the partition in the case of RMOTE. When D3KMS is appended to the user programs, the standard 134-word buffer within D3KMS is used to collect these messages.

- Processes intervening $STDLIST/$STDIN requests from the HP 3000. These could be the result of an operator command or a PTOP slave program's I/O request.

- Checks for the RTE BR command during $STDLIST/$STDIN data transfers, and sends a break or CONTROL Y request to the HP 3000, depending on what the user specifies.

D3KMS contains code for the following

- the ICC function for interrogating HP 3000 condition codes
- the PRCNM subroutine, which allows master users to access the session opened by a father program's HELLO
- subroutines for managing the appendage area (storing and receiving parameters) to be used by user-callable subroutines (PTOP and RFA)
- global data words used by HELLO, D3KMS, and other master subroutines.

Caution Hewlett-Packard does not support the use of continuation records by user software for writes to $STDLIST or reads from $STDIN. No guarantee can be made that D3KMS has enough space to buffer these messages.

D3KMS checks the process number list each time it has a message to send. If the link has gone down (and UPLIN has flushed all the entries for that HP 3000), it will not find its session number in the list and will report a DS06 (illegal request) error. If the link goes down while D3KMS has an outstanding request, UPLIN sends it a zero-length record and D3KMS reports a DS05 (timeout) error.

D3KMS writes Bisync and X.25 messages to the communications LU with I/O completion status sent to QUEZ's class. PSI messages have a word containing the HP 3000 LU number added to the beginning of the message; header and appendage are written in the regular data buffer while the data area is transferred in the Z-buffer.

The example in Figure 9-20 shows how RMOTE and D3KMS work together. Suppose an operator runs RMOTE, switches to remote, signs onto an HP 3000, and sends a LISTF command.

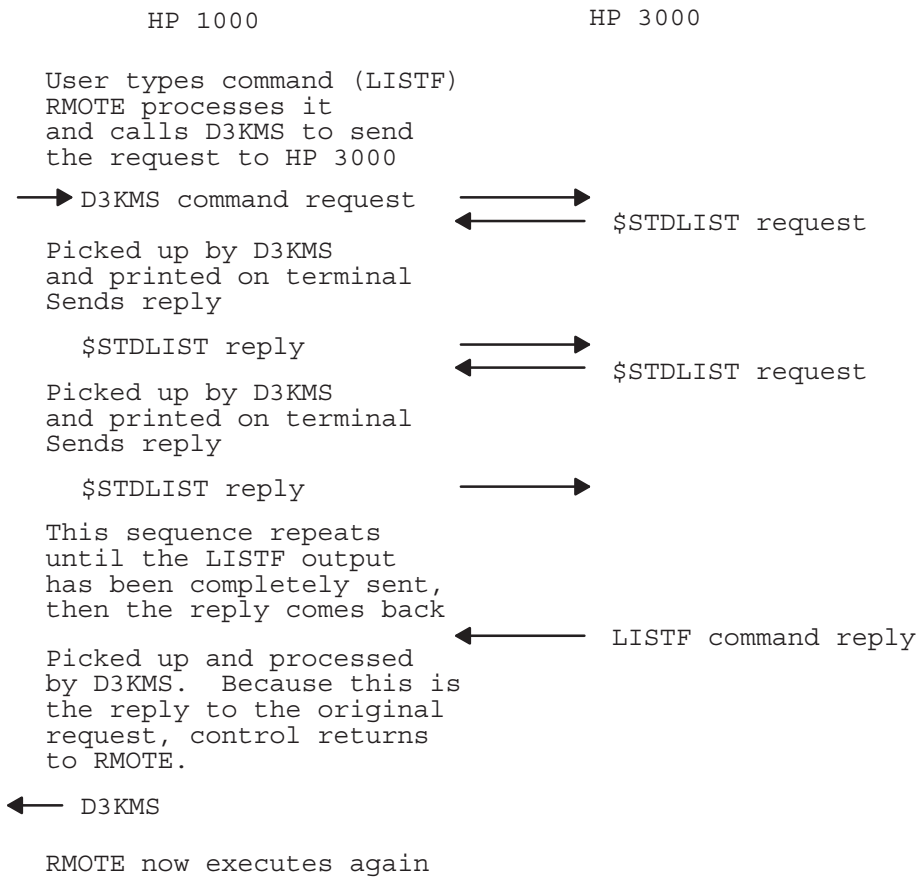


Figure 9-20. RMOTE and D3KMS Operation

RTE to MPE Slave Side Communication

Figure 9-21 is a block diagram of the software which accepts DS/3000 messages from users at the HP 3000, converts them to DS/1000-IV format, presents them to the proper slave monitor, and returns the reply to the HP 3000.

When QUEX receives a request buffer it sends it to RQCNV's I/O class number. RQCNV changes the request buffer from the format used by DS/3000 to that used by DS/1000-IV. It determines the proper monitor and writes the buffer onto that monitor's I/O class number.

The same monitors are used for requests made from a HP 3000 as for requests from RTEs. Their operation is as described previously. When they complete a transaction, they call `#SLAV`. `#SLAV`'s operation is similar to that described for RTE-RTE communication, except it threads the buffer to the RPCNV's I/O class number for conversion to HP 3000 format. RPCNV reformats the reply buffer and sends it to the HP 3000.

CNSLM, the console monitor, prints message directed to the HP 1000 from the MPE operating system for `TELL`, `WARN`, and aborted session messages.

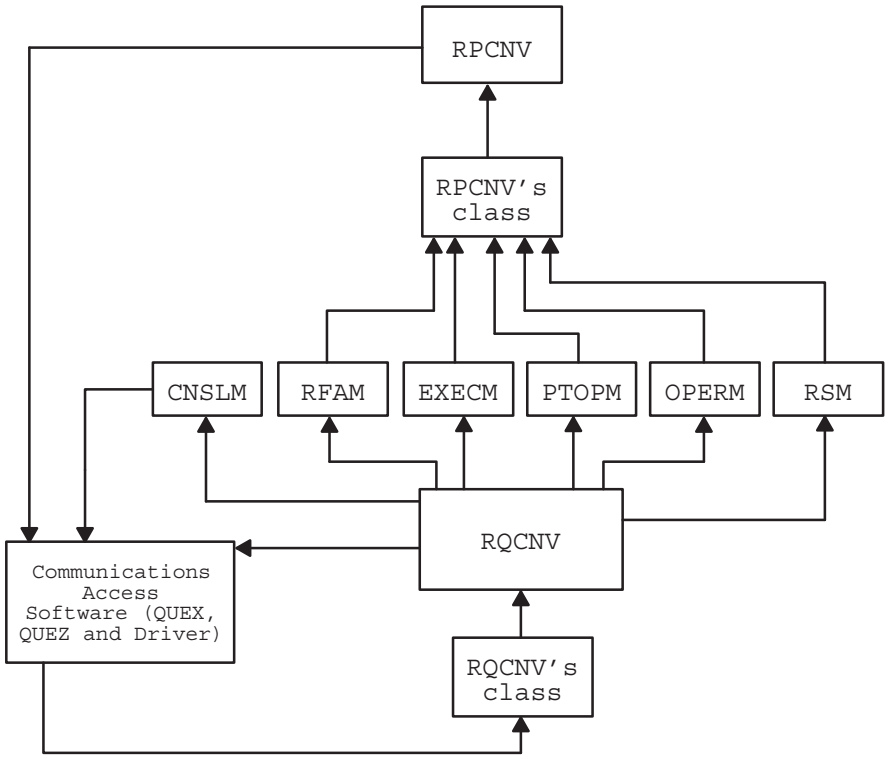


Figure 9-21. DS/1000-IV to DS/3000 Slave Side Modules

Request and Reply Converters

When the RQCNV receives a DS/3000 request, it reformats the parameters into a DS/1000-IV request, allocates a slave TCB, builds a Transaction Status Table entry, and writes the request to the proper slave monitor (CNSLM, RFAM, EXECM, PTOPM, OPERM, or RSM).

The procedure is more complicated when a request is received with its continuation bit set. RQCNV must save the data in the first request (by using an I/O class to hold it temporarily) and send a continuation reply to the HP 3000. When the continuation request is received from the HP 3000, its data is also written to the holding class. When the final continuation request (indicated by no continuation bit set) is received, RQCNV collects all the data off the holding class, deallocates it, and sends the data and its DS/1000-IV request to the monitor.

After the monitor has completed processing a request, it calls `#SLAV`. `#SLAV` tests the HP 3000 Master bit in the stream-type word. If set, indicating that the request came from an HP 3000, it sends the reply to the reply converter, RPCNV.

RPCNV takes DS/1000-IV replies, converts them to DS/3000 replies, and sends them to the 3000. RPCNV uses the information saved in the Transaction Status Table to build the DS/3000 header and to determine which HP 3000 sent the request. (RPCNV deallocates the TST entry when it is done.)

When a read reply is too large to fit in a single communications buffer, RPCNV fits as much data as possible in the reply, sets the continuation bit, sends the reply to the HP 3000, and writes the

rest of the data onto a holding class. When the HP 3000 sends a continuation request, RQCNV passes it through to RPCNV and the next part of the data is sent. When all the data has been sent, the holding class and TST entry are deallocated.

CNSLM

CNSLM receives system (as opposed to user) requests from the HP 3000 for $STDLIST (i.e., standard list output). It directs I/O requests from MPE `TELL` or `WARN` commands to LU 1 and the log-on LU for the process. Logoff messages for sessions aborted by MPE are printed on the system console.

It recognizes stream types 20 (write) and 23 (control) of class 20, rejecting all others. Processing depends on stream type as shown in Table 9-4.

Table 9-4. CNSLM Processing

Stream Type	Description	Processing
20	$STDLIST directed to terminal	If "from process number" equals zero, the message is printed on LU 1 and the LU specified by the "to process number." No reply is sent. Otherwise, an output call is used to send the message to the system console and a reply is sent.
23	FCONTROL to $STDLIST	A reply is sent to the HP 3000.

HP 1000 to HP 3000 Clean-Up

If an HP 1000 program logs on to a HP 3000 but terminates without issuing a `BYE`, resources need to be cleaned up on both sides. UPLIN removes the local PNL entry and terminates the HP 3000 session by issuing a kill job request. When a reply to the kill job request is received from the HP 3000, QUEX scans the PNL and sends a DS/3000 termination request if no other entries exist for the indicated link.

QUEX acts on the HP 3000's reply. Rejects are ignored. Accepts cause QUEX to disconnect the given line and reinitialize it in secondary mode.

For X.25 links, the disconnect from QUEX causes the associated switched virtual circuit to be returned to the POOL. (The secondary mode initialization is rejected by X.25 customizing subroutine `CXL66`.) QUEZ checks the 3000 LU Table for POOL LUs and, if it finds one, marks it down so UPLIN can clean up. When UPLIN finds a down LU in the table, it deletes it and calls RELSX. RELSX calls the X.25 routine RPOOL which returns the SVC to the POOL.

Memory Manager

Note: In this section, the term, *message*, is used in the general sense—that is, a data buffer sent via `IPCSend` in NetIPC or `send()` in BSD IPC, including the protocol headers as they are added.)

Overview

The Memory Manager (MM) manages DSAM, the NS-ARPA/1000 memory area. NS-ARPA/1000 uses DSAM to store tables, global values, and all NS-ARPA messages and messages for DS/1000-IV Compatible Services sent over non-Router/1000 links before transmission.

MM divides DSAM into the following three areas:

- Global area
- Tables area
- Buffer area

Global Area

The *global area* in DSAM is always mapped in (the Page Mapping Registers used to address it are set once, at NS-ARPA initialization time). MM stores information in the Global area that it needs to access frequently or quickly. The information that MM stores here includes the following topics:

- information for its operation, such as the state of the buffer area, MM statistics, and references to any physical memory parity errors
- global, fixed-size NS-ARPA tables, and the information to map and access all other (dynamically-sized) NS-ARPA tables
- sockets and Socket Buffers (SBUF records, defined below)
- user records, which contain information on the protocol handlers (such as TCP and IP), which use MM.

SBUFs. For each NetIPC or BSD IPC socket, MM allocates an inbound and an outbound SBUF (socket buffer) to hold inbound and outbound messages queued on NetIPC or BSD IPC sockets. For each SBUF, MM allocates an SBUF record to hold information about the SBUF. (Inbound messages are said to be queued if they have arrived but have not been read by the socket's user. Outbound messages are queued if the user has sent them (via `IPCSend` for NetIPC or `send()` for BSD IPC), but they have not been processed by the lower-level protocol (such as TCP).)

The size of the global area is determined by responses to the NSINIT dialogue.

Tables Area

The *tables area* is dynamically mapped in as tables are requested. The Page Mapping Registers (PMRs) used to address it are set each time a table is accessed.

The tables area contains the dynamically-sized NS-ARPA tables used by protocols and services, which are referenced in the global area. For example, IP's Gateway Table and Router/1000's Nodal Routing Vector are stored here.

The size of the tables area is determined by responses to the NSINIT dialogue.

Buffer Area

The *buffer area* is also dynamically mapped in and contains Message Buffers (Mbufs).

Mbufs. An Mbuf is one page (2048 bytes)of physical memory; it uses 12 bytes for control information and can hold 2036 bytes of data.

Mbufs are allocated for buffering messages by the protocol modules or for user socket buffers as needed. The Mbufs are not reserved or allocated until they are needed.

When a VC socket is created (via IPConnect or IPCRecvCn in NetIPC or with `connect()` or `accept` in BSD IPC) the maximum socket buffer sizes are compared with the current number of free Mbufs. If there are not enough free Mbufs to accomodate the maximum socket buffer size, the socket will not be created. This prevents creation of sockets when the amount of free memory is low. However, it is still possible for previously created sockets to consume all of the available Mbufs.

Maccts. The size of the buffer area is the size of the DSAM SHEMA left after space has been allocated for the global and tables area.

Network File Transfer

This section describes the NFT message flow and briefly describes its implementation. Before reading this section, you should be familiar with the NFT and NetIPC user interfaces (documented in *NS-ARPA/1000 User/Programmer Reference Manual*).

Modules

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 Data Structures in DSAM

NFT uses two words in DSAM to store NFT values configured at network initialization time (via NSINIT). NFT uses one word to store the value of the Transport Checksum flag, which determines whether or not NFT is to use Level 4 (TCP) checksumming with its peer at the remote node. NFT also uses a word to store the default NFT buffer size. NFT modules use this value to negotiate the amount of data sent per message.

The NFT error message file (`NSERRS.MSG`) is also stored in DSAM.

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 9-5.

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 9-22A.)

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 9-22B.)

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 9-22C.)

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 9-22D.)

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.

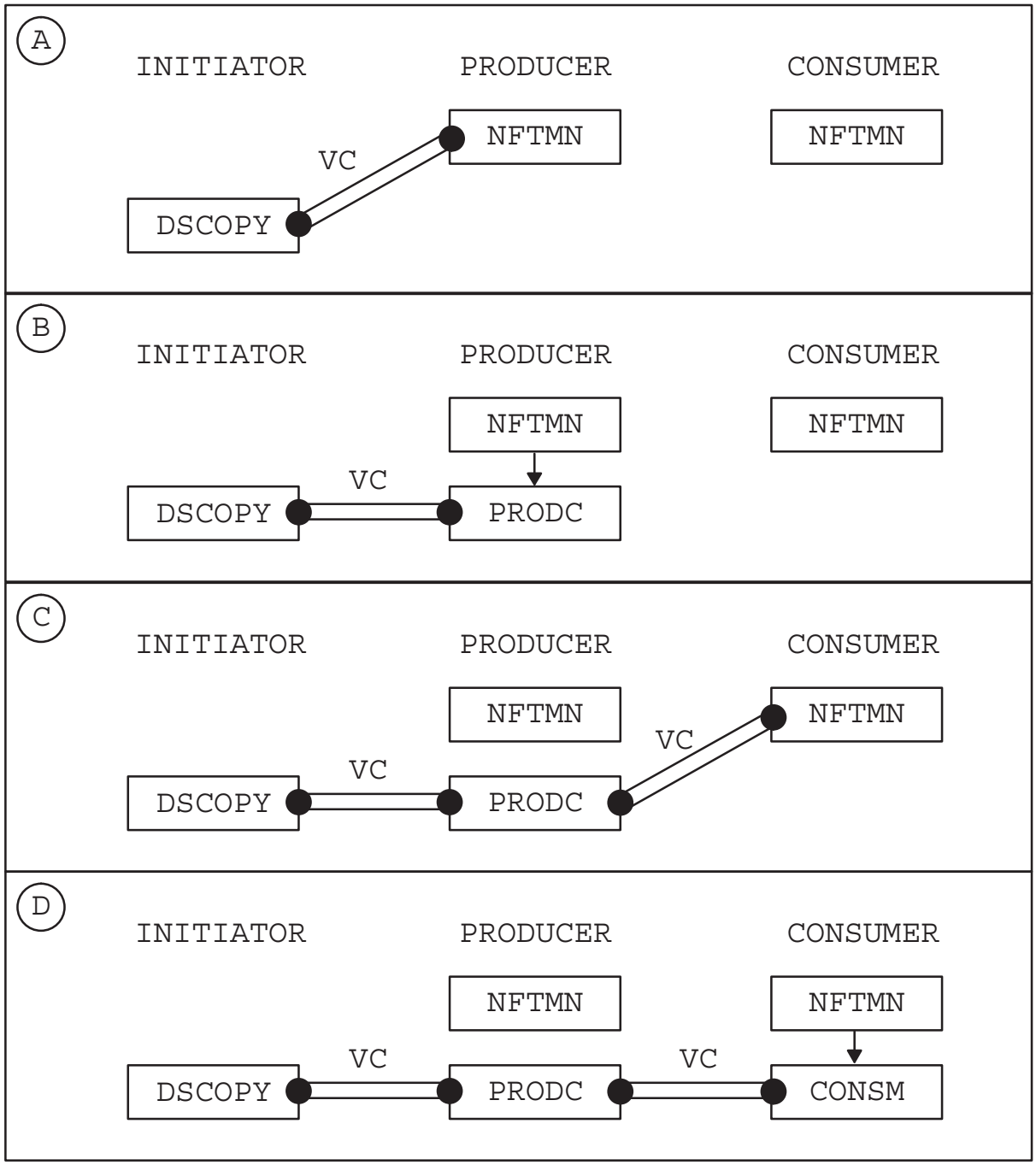


Figure 9-22. NFT Connection Establishment

Message Flow Summary

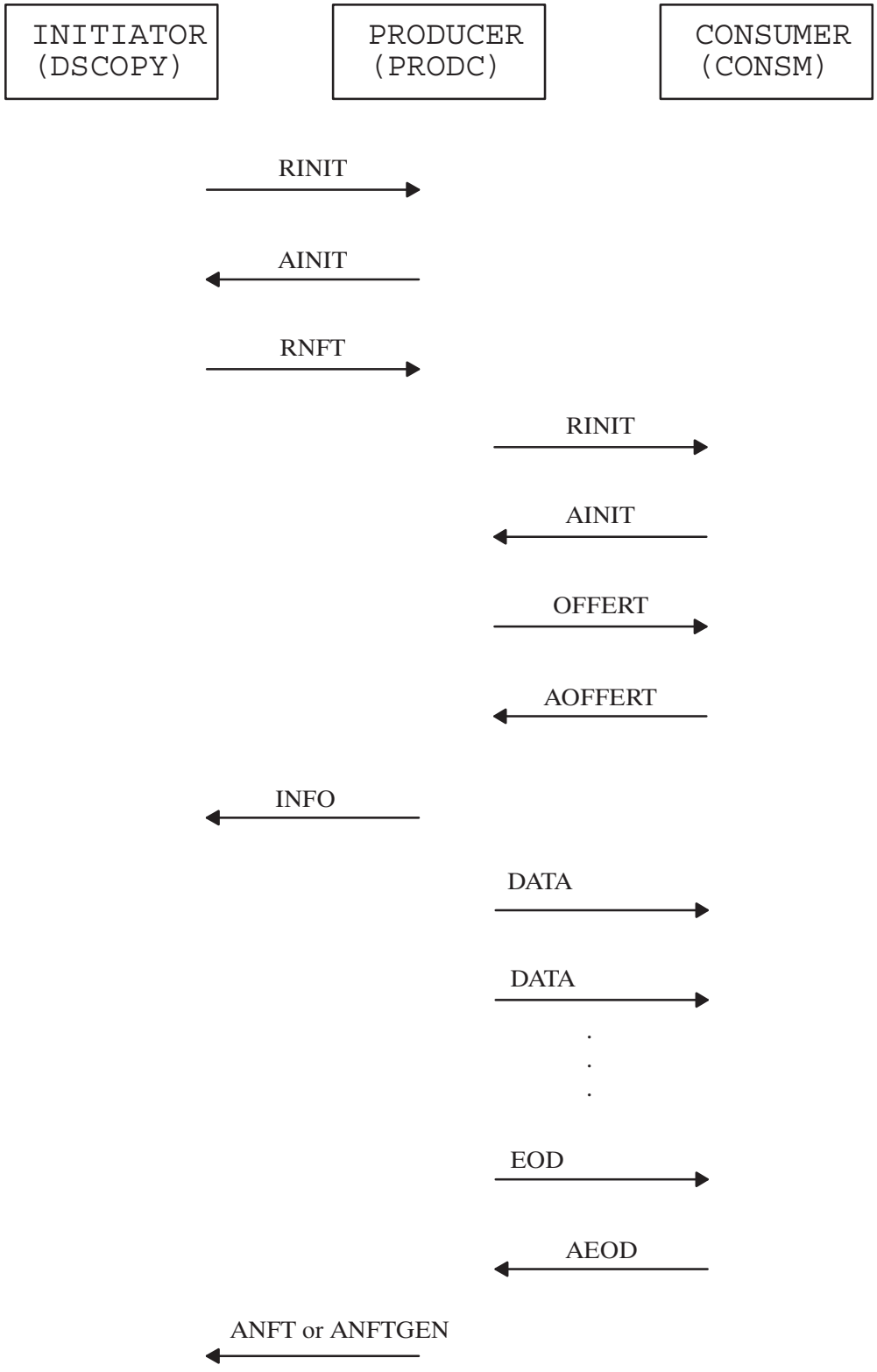


Table 9-5. NFT Messages

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). Send 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.
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.)

Message	Purpose
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.

DSCOPY

DSCOPY is scheduled at the user's request. Its main function is to parse the user's commands and interact with PRODC.

If the user makes a file transfer request, DSCOPY will attempt to establish an NFT connection with the Producer node. If this connection already exists, DSCOPY will send an RNFT message to PRODC.

If the user exits DSCOPY, DSCOPY will shut down its VC connection to PRODC and terminate.

NFTMN

If you specify NFT in the NSINIT dialogue, NSINIT will schedule NFTMN. NFTMN will then remain scheduled, waiting for requests to start NFT connections. NFTMN creates any necessary sessions and schedules the required server (PRODC or CONSM). When NFTMN is first scheduled, it creates a call socket and binds it to a well-known address, making it a well-known call socket. NFTMN then waits on this call socket for VC connection requests. When it receives an IPCConnect request, it creates a VC socket and uses the node's NFT Buffer Size (configured via NSINIT) for the VC receive buffer size.

After the VC connection is established, NFTMN performs the following tasks:

- Waits for an RINIT message on the VC socket.
- When NFTMN receives the RINIT message, it previews the message (using IPCRecv with the PREVIEW flag set), and schedules a clone of the requested server (PRODC or CONSM).
- Creates a NetIPC user record for the server and links it to NFTMN's user record.
- Writes the VC socket descriptor of the VC socket that received the RINIT to to the server's user record (this passes the VC socket to the server, and is similar to using IPCGive/IPCGet).
- In multi-user systems, attempts to log on, using the logon string passed in RINIT, or attaches to an existing session. (Refer to "Session Management" for more information.)
- Schedules the server, passing the session ID to which the server is to attach.
- Removes the link to the user record.

PRODC

PRODC produces the source file, and must continually communicate with DSCOPY and CONSM. One reason that PRODC must keep the communication channel to DSCOPY open after the file transfer begins is so that NFT can respond to user requests to cancel or abort file transfers.

Main Control Loop

Once PRODC is scheduled and has established an NFT connection with CONSM, it remains in the following control loop:

- Call `IPCSelect` in non-blocking mode on its VC sockets to DSCOPY and to CONSM to see if any messages have arrived from these processes.
- If a message is queued on either VC socket then read it and process it. If both sockets have messages queued on them, PRODC will read the message from DSCOPY first.
- If no messages have arrived, and PRODC is in the middle of a file transfer, PRODC will send CONSM the next buffer of file data in a `DATA` message. (Refer to "Sending File Data" below for more information.)

  If PRODC is not in the middle of a file transfer, PRODC checks if it is ready to start a file transfer or request CONSM to create a directory. (For more information on group transfers, refer to "PRDC1" below.) If so, PRODC starts a new file transfer by sending an `OFFER` message for a transparent mode file transfer or `OFFERI` message for an interchange mode file transfer; or, PRODC requests CONSM to create a file directory with a `DIRECTORY` message.

  If PRODC has no more files or directories to transfer, it calls IPCSelect in blocking mode and waits for a message to arrive from CONSM or DSCOPY. If a message arrives, PRODC will process it.

PRODC executes this loop until DSCOPY sends it an `ABORT` or `CANCEL`, or DSCOPY closes the VC connection. PRODC's termination handling is described in the subsection "Abort Handling."

Sending File Data

To send file data, PRODC uses two buffers, a send buffer and a disk file buffer. These buffers are stored in NFT code space. PRODC uses the send buffer for the `IPCSend` data buffer, and uses the disk file buffer for reading file data from the disk.

The size of the data message in the NetIPC send/receive buffers is determined by the NFT buffer size (configured with NSINIT) on the Producer and Consumer node. For the VC connection, the nodes use the NFT buffer sizes specified in NetIPC `IPCConnect` and `IPCRecvCn` parameters. When the NFT connection is being initialized, the NFT modules negotiate the size of buffers to send by exchanging each node's configured NFT buffer size in the `RINIT` and `AINIT` messages. If the buffer sizes are not the same, the lower buffer size is used. However, the amount of file data that PRODC can actually send in a buffer is the negotiated buffer size minus two words, since PRODC must prepend a two-word `DATA` message header to the file data before sending it.

For transparent mode file transfers, PRODC ignores record markers. PRODC opens the source file with the `FmpOpen F` option (unbuffered access), and reads large sections of raw file data into the disk file buffer. PRODC then writes the file data to the send buffer without modifying it. PRODC writes the file data from the disk file buffer to the send buffer in blocks that are the size of the negotiated buffer minus two words. PRODC then adds the `DATA` header and calls `IPCSend` to send the data to CONSM.

For interchange mode file transfers, PRODC may have to alter the record format. If the record size is less than or equal to the size of the negotiated buffer size minus two words, PRODC reads the source file into the disk file buffer, one record at a time. PRODC then writes the record into the send buffer, and uses the space in the send buffer to do any record format changes. PRODC will buffer as many records as will fit in the send buffer before sending them. If the record size is greater than the negotiated buffer size minus two words, PRODC reserves a partition in the disk file buffer that is the size of the longest record in the file. PRODC will use this partition to format the records, and use the remainder of the disk file buffer to read data from the disk.

Interchange Format Records. Record characteristics for interchange format transfers are dependent on source file characteristics, source and target node types, and any user-specified options. The record characteristics are described in *NS-ARPA/1000 User/Programmer Reference Manual*.

For variable length interchange records, PRODC prepends a 2-word header to each record that specifies the record length in bytes.

For sparse source files (files that are missing intermediate records), PRODC prepends a 2-word header to each record that specifies the record's logical record number.

Directories

If the user specifies a file mask for the source file and any directory matches that mask, NFT will transfer all the files in that directory. If the Consumer node has a hierarchical file system, PRODC will send CONSM a `DIRECTORY` message, specifying the directory, before transferring the files. PRODC will not resume execution until it receives an `ADIRECTORY` message from CONSM. If the directory already exists at the consumer node, CONSM will report an error in the `ADIRECTORY` message. PRODC will report this to DSCOPY in an `ANFT` or `ANFTGEN` message.

PRDC1

A PRDC1 clone is scheduled by PRODC when the user specifies a file mask or directory type extension for the source file. DSCOPY sends the file specification to PRODC. PRODC schedules PRDC1, passing the file specification in the scheduling parameters. PRDC1 will produce file and directory names from the file specification and write them to a scratch file. When PRDC1 is done producing file and directory names, it closes the scratch file and terminates, passing the scratch file name, number of file names and number of directory names in its return parameters to PRODC. PRODC opens the scratch file and processes the file and directory names sequentially, decrementing the file name and directory name counts each time it processes an entry.

Error Handling

If PRODC encounters an error while trying to produce a file or directory, it will send a CANCEL to CONSM. CONSM will respond with an AABORT, and PRODC will then send an ANFT or ANFTGEN to DSCOPY, reporting the error. Similarly, if CONSM reports an error to PRODC in an ADATA or ADIRECTORY message, PRODC will report the error to DSCOPY in an ANFT or ANFTGEN message.

Termination Handling

Normally, NFT shuts down when DSCOPY closes the VC connection to PRODC (as previously described in "General Flow"). Therefore, if PRODC detects that DSCOPY has closed the VC connection while PRODC was waiting for a command from DSCOPY (PRODC was not in the middle of a file transfer), PRODC follows its normal termination procedure; it will close its VC connection to CONSM and terminate.

However, if DSCOPY closes the VC connection while PRODC is in the middle of processing a user's command, PRODC will continue to process the user's command. In the case of a group transfer, PRODC will continue until it has attempted to transfer all the files indicated by the file mask and requested CONSM to create any required directories.

If PRODC's VC connection to CONSM goes down, PRODC makes note of it. If PRODC was in the middle of a file transfer or waiting for the result of a directory creation, PRODC will notify DSCOPY that its connection to CONSM went down. PRODC will then abort the user's command. In the case of a group transfer, PRODC will not try to process any more file or directory names. PRODC will remain scheduled; if it receives another file transfer request from DSCOPY, it will attempt to open a new connection to a Consumer node.

CONSM

CONSM is scheduled by NFTMN at the node where the destination file will be created.

Main Control Loop. Once CONSM is scheduled it executes its main control loop. In the main control loop, CONSM steps through the following functions:

- Block, waiting for a message to arrive.
- If an OFFERI or OFFERT message arrives, attempt to open the file specified. For OFFERI messages, check to see which file format to use. Send back an AOFFERI or AOFFERT. Return to top of loop.
- If a DATA message arrives, write the file data to disk. Return to top of loop.
- If a DIRECTORY message arrives, attempt to create the directory. Return the result to PRODC in an ADIRECTORY message.
- If an EOD message arrives, close the file and send an AEOD. Return to top of loop.

CONSM executes this loop until it encounters an error while writing file data to the disk or while creating a directory; or until PRODC sends it an ABORT or CANCEL; or PRODC closes the VC

connection. If CONSM encounters an error while writing file data to the disk or while creating a directory, it will return an error to PRODC in an ADATA or ADIRECTORY message. If PRODC sends CONSM an ABORT or CANCEL, CONSM will respond with an AABORT and return to the top of its main control loop. If PRODC closes the VC connection, CONSM cleans up and terminates, as described in "General Flow."

Session Management

For multi-user systems, DSCOPY passes logon strings or session keys to PRODC and CONSM for the sessions used in file transfers. If the Producer node is the same as the Initiator node and the user does not specify a source logon string, or if the Consumer node is the same as the Initiator node and the user does not specify a target logon string, DSCOPY will pass a session key to the particular server. (A session key consists of the session ID of an existing session and a number used for security.) In all other cases, DSCOPY will pass a logon string.

DSCOPY passes the logon strings or sessions keys as follows: for the source logon string or session key, DSCOPY passes it to NFTMN (at the Producer node) in the RINIT message; for the target logon string or session key, DSCOPY passes it to PRODC in the RNFT message, and PRODC forwards it to CONSM in the RINIT message.

If NFTMN receives a logon string, NFTMN creates a local session, passes the session key for that session in the server's (PRODC or CONSM) scheduling parameters, and records the session key in the server's (PRODC or CONSM) user record. The server is now the owner of that session, and when the server process terminates, UPLIN will log off that session.

If NFTMN receives a session key (a session already exists for the logon string), NFTMN passes the session key for that session in the server's (PRODC or CONSM) scheduling parameters. The server attaches to that session, but the process that created the session still owns it.

Session Termination

All sessions created by NFT modules are logged off by UPLIN after the servers terminate.

Error Handling

If DSCOPY is run interactively, NFT will return error messages from DSAM if NSINIT was able to store the contents of NSERRS.MSG in DSAM at initialization time. Otherwise, NFT will return an error code.

Abort Handling

If DSCOPY, PRODC, or CONSM abort, UPLIN cleans up the resources as it does for all NS-ARPA user processes. UPLIN will shut down any VC connections and log off any sessions to which the process was attached. When UPLIN shuts down the VC connections, the other modules involved in the file transfer will note this and terminate, as described in the previous subsections.

The RTE file system will purge any files (scratch files or target files) left open by the aborted module because all files are opened with the FmpOpen T (Temporary) option.

Index

Symbols

#GET, 9-38
#GRPM, 9-18
#MAST, 9-17, 9-18, 9-23
#MSSM, 9-41
#NAT, 9-41
#NRVS, 9-17
#RMSM, 9-40
#SEND, 9-28
#SLAV, 9-37

Numbers

802 LI, statistics, 2-20

A

ACEPT, processing, 9-43
address
  IEEE 802.3 station, 2-5
  IP, 2-5, 2-18
  multicast, 2-5
  Router/1000, 2-5, 2-18, 2-21
  troubleshooting, 1-6, 1-8
ANH, 9-5, 9-9
APLDR, 2-28
appropriate next hop, 9-5, 9-9
architecture, 9-1

B

Berkeley sockets, 9-2
Bisync, 9-48
  board statistics, 2-13
  I/O requests, 9-52
  ID sequences, 8-9
  link troubleshooting, 1-5, 1-17
  message records, 7-1
  protocol, 9-50
BREVL, 4-1
  runstring, 4-4
BRTRC, 5-1, 5-5
  runstring, 5-5
BSD IPC, 9-2, 9-5
  troubleshooting, 1-10, 1-11
buffer area, 9-62
buffer size, NFT, 2-9
BYE, subroutines, 9-54

C

cancellation (MA) statistic, 2-17
checksum, 2-9
class I/O rethreads, 9-19
class number, 2-24, 2-29
  IFP, 2-9
  INPRO, 2-9
  OUTPRO, 2-9
CNSLM, 9-60
configured resources, statistics, 2-7
connect site path report, 9-7
CONSM, 9-72
control buffer, tracing, 5-23, 5-26
cost matrix, 9-26
CSR, 9-8
CXL66, 9-51

D

D3KMS, 7-1, 9-56
data flow, 9-12
debugging, 5-1, 5-2
device type, 2-10
DEXEC, 9-45
  troubleshooting, 1-10
distributed EXEC, 9-45
distributed LOGON/LOGOFF (DLGON), 9-40
DLGON, 9-40
down reference, 2-20
DS/1000-IV
  compatible services communication (RTE-RTE), 9-48
  compatible services internals, 9-14
  compatible services parameter modification, 8-1
  compatible services statistics, 2-28
  compatible services tracing, 5-20, 5-24
  compatible services troubleshooting, 1-11
  local appendage, 5-26
  software, 1-8
DS/3000 messages, 9-49
DSAM, 9-61
  buffer area, 9-62
  global area, 9-61
  tables area, 9-62
DSCOPY, 9-69
DSLIN, processing, 9-47
DSMOD, 8-1
  /A command, 8-5
  /E command, 8-8
  /I command, 8-9
  /L command, 8-10
  /N command, 8-11
  /S command, 8-13
  /T command, 8-14
  CN command, 8-6
  commands, 8-2
  DI command, 8-7

runstring, 8-1
DSQ, 9-19
DSRTR, 1-8
DSTEST, 1-18
dynamic rerouting, 9-26
processing, 9-27

E

event logging, 4-1
log mask, 4-2, 4-3, 4-5
output, 4-7
statistics, 2-27
event message, 9-4
EVMON, 4-1
output, 4-7
runstring, 4-2
statistics, 2-27

F

file transfer protocol, troubleshooting, 1-12
FINIS, processing, 9-44
FMTRC, 5-1, 5-6
dialogue, 5-8
formatting, 5-6
runstring, 5-6
trace files, 5-15
VMA size, 5-7
FTP
errors, 1-6
troubleshooting, 1-12

G

gateway table, 9-9
See also GT
GET processing, 9-43
global area, 9-61
global socket descriptor, 2-19, 5-19
GRPM, 9-25
GSD, 2-19, 5-19
GT, 2-6, 9-9

H

hardware
failures, 1-13
troubleshooting, 1-5
HDLC
board statistics, 2-11
link connect processing, 9-45
link down processing, 9-47
link failure, 9-19, 9-20
link troubleshooting, 1-5, 1-16
message processing, 9-14, 9-20
read and write processing, 9-47
retry processing, 9-46
header, 9-3
HELLO, subroutines, 9-54
hop count, modifying, 8-14
HP 3000
DSLIN, 9-47
ID sequences, 8-9
LU, 2-28
message classes, 7-20
message streams, 7-20
message tracing, 7-1

I

ID*66, 9-20, 9-45, 9-48, 9-51
idle session timeout, modifying, 8-14
IEEE 802.3
address, 2-6, 2-21
board statistics, 2-15
message processing (DS/1000-IV), 9-15
IFP, class number, 2-9
IFPM, 9-15
Image/1000, troubleshooting, 1-8
information utility, 2-1
initialization, troubleshooting, 1-4
INPRO, 9-1, 9-15
class number, 2-9
internet protocol. *See* IP
IP, 9-2, 9-5
address, 1-6, 2-5, 2-18
statistics, 2-8, 2-20
IPCConnect, 9-10
IPCCreate, 9-8
IPCLookUp, 9-9
IPCName, 9-8
IPCRecvCn, 9-10

L

LAN
board statistics, 2-15
link troubleshooting, 1-5, 1-14
message processing (DS/1000-IV), 9-15
station address, 1-6
statistics, 2-21
trace records, 5-21
level, upgrade, 2-18, 2-28
LI, 9-2
statistics, 2-20
line down count, modifying, 8-14
linedown statistic, 2-16
link
disable, 8-7
enable, 8-10
troubleshooting, 1-13
link interface, 9-2
link vector, 9-27
local appendage, 5-26
log mask, 2-27, 4-2, 4-3, 4-5
LOG3K, 7-1
?? command, 7-4
/E command, 7-5

commands, 7-3
EN command, 7-5
EX command, 7-5
LU command, 7-6
NO command, 7-5
runstring, 7-2
TY command, 7-7
UP command, 7-8
LOGCHG, 4-1
runstring, 4-5
logging, 4-1
output, 4-7
statistics, 2-27
LU, 2-5, 2-10, 2-11, 2-21, 2-22, 2-28

M

MA, 9-17, 9-30
master
request, 9-17
subroutines (to MPE), 9-54
TCB, 2-25
timeout, 2-28
master program session module, 9-41
master timeout, modifying, 8-14
Mbuf, 9-62
memory buffer, 9-62
memory manager, 9-61
message, 9-3, 9-12
DS/3000, 9-49
message accounting, 9-17, 9-30
statistics, 2-16
message header, Router/1000, 9-41
message size, 9-13
message tracing, 7-1
module, protocol, 9-1
monitor, scheduling, 8-13, 9-39
multicast address, 2-5

N

name, 9-10
name records, statistics, 2-8
NetIPC, 2-19, 9-2, 9-5
troubleshooting, 1-10, 1-11
VC connection establishment, 9-8
network account table, 9-41
network file transfer
See also NFT
troubleshooting, 1-12
NFT, 9-62
abort handling, 9-73
buffer size, 2-9
checksum, 2-9
data structures, 9-63
directories, 9-71
error handling, 9-73
file buffers, 9-70
general flow, 9-63
interchange format, 9-71
message flow, 9-67
modules, 9-62
session management, 9-73
termination handling, 9-72
tracing, 5-18
troubleshooting, 1-12
NFTMN, 8-13, 9-69
scheduling, 8-13
NICE formatted records, 5-17
examples, 5-17
NICE mode, 5-12
NICE option, 5-4
nodal path report, 6-2, 9-7, 9-10, 9-11
nodal registry, 9-11
list utility (NRLIST), 6-1
nodal routing vector, 2-18, 8-11, 9-27
modifying, 8-6
node name, 2-5
NPR, 6-2, 9-7, 9-11
NRINIT, 6-2
NRLIST, 6-1
dump mode, 6-3
raw mode, 6-3
runstring, 6-1
NRV, 2-18, 8-6, 8-11, 9-27
NS-ARPA program, statistics, 2-7
NSERR.MSG, 1-6
NSINF, 2-1
? command, 2-2
+ command, 2-2
A command, 2-2, 2-5
abort display command, 2-2
C command, 2-7
command summary, 2-3
continue display command, 2-2
exit command, 2-2
exiting, 2-2
help command, 2-2
I command, 2-10
L command, 2-11
M command, 2-16
main menu, 2-4
N command, 2-18
P command, 2-19
Q command, 2-2
R command, 2-22
runstring, 2-2
S command, 2-23
suspend command, 2-2
T command, 2-24
U command, 2-27
V command, 2-28
W command, 2-30
Z command, 2-2
NSINIT, 1-4
NSTRC, 5-1
message posting, 9-12, 9-13
runstring, 5-3
statistics, 2-27
VMA file, 5-3

O

OUTPRO, 9-1, 9-15
  class number, 2-9

P

packet exchange protocol. *See* PXP
parameter modification, 8-1
path, 9-3, 9-6
path flow, 9-1
path record, 9-4
PCB, 9-5
PCLOS, processing, 9-43
PCONT, processing, 9-42
performance, 5-1
PING, 3-1, 3-2
  simultaneous sessions, 3-2
PNL, 2-26
PNRPY, processing, 9-43
POPEN, processing, 9-42
PRCNM, 9-57
PRDC1, 9-71
PREAD processing, 9-42
probe, 9-10
  statistics, 2-8
process information, 2-19
process number list, 2-26
PRODC, 9-70
PROGL, 2-28
program information, 2-19
program-to-program communication, 9-42
protocol, modules, 9-1
protocol stack, 9-3
pseudo LU, 2-30, 2-31
pseudo terminal LU, 2-30, 2-31
PSI, links, 9-51
PTOPM, 9-42
PWRIT, processing, 9-42
PXP, 9-2, 9-5
  statistics, 2-20

Q

QUEUE, 9-36, 9-49
QUEX, 7-1, 9-52
QUEZ, 9-52

R

REJCT, processing, 9-43
remote, session, 9-17, 9-40
remote busy retry, 2-28
  modifying, 8-14
remote file access, statistics, 2-28
remote file access monitor, 9-44
remote quiet wait, 2-28
  modifying, 8-14
remote session, statistics, 2-23
rerouting, 9-19, 9-26
  processing, 9-27
resource, clean up, 9-39, 9-60
resource number, 2-28
retry
  limit, 9-46
  timeout, 2-28
RFA, statistics, 2-28
RFAM, 9-44
Router/1000
  address, 2-5, 2-18, 2-21
  header, 5-25
  message header, 9-41
  statistics, 2-22
  trace records, 5-24
RPCNV, 9-49, 9-59
RQCNV, 9-49, 9-58, 9-59
RTE
  to MPE master side communication, 9-54
  to MPE slave side communication, 9-58
RTR LI, statistics, 2-21

S

SBUF, 9-61
segment size, 9-13
  TCP, 2-9
select code, 2-10
session, remote, 9-17, 9-40
SL, processing, 9-44
slave, TCB, 2-25
slave list, processing, 9-44
slave off, processing, 9-44
slave timeout, 2-28
  modifying, 8-14
SO, processing, 9-44
socket
  record, 2-19
  statistics, 2-8
  trace records, 5-19
  types, 2-20
socket buffer, 9-61
socket registry, 9-11
  tracing, 5-19
software, troubleshooting, 1-5
software revisions, troubleshooting, 1-8
station address, 2-5
store-and-forward traffic, 9-25
subnet mask, 1-6
system resources, troubleshooting, 1-7

T

table sizes, troubleshooting, 1-7
tables area, 9-62
TCB, 2-24, 9-18
  allocation, 9-17
TCP, 9-2, 9-5
  segment size, 2-9
  statistics, 2-20
  troubleshooting, 1-11

TELNET
errors, 1-6
number of server programs, 2-7, 2-9, 2-30, 2-31
number of user programs, 2-7, 2-9, 2-30
pseudo terminal LU, 2-30, 2-31
troubleshooting, 1-12
timeout, retry limit, 9-46
timeout (MA) statistic, 2-17
timeout value, 2-28
timeouts, troubleshooting, 1-11
timing values, modifying, 8-14
trace file, 5-15
header, 5-15
tracing, 5-1, 7-1
control buffer, 5-23, 5-26
DS/1000-IV compatible services, 5-20
message posting, 9-12, 9-13
NFT, 5-18
NICE formatted records, 5-17
Router/1000 trace records, 5-24
socket registry, 5-19
socket trace records, 5-19
statistics, 2-27
terminating, 5-5
transaction control block, 2-24, 9-18
allocation, 9-17
transmission control protocol. *See* TCP
transport, 9-2
TRC3K, 7-1, 7-10
?? command, 7-12
commands, 7-11
EXIT command, 7-13
FORMAT command, 7-14
LIST command, 7-15
PRINT command, 7-16
runstring, 7-10
sample output, 7-18
SET command, 7-17
TRFAS, 1-8
troubleshooting
addresses, 1-6, 1-8
BSD IPC, 1-10, 1-11
DS/1000-IV compatible services, 1-11
guidelines, 1-1
hardware, 1-5
hardware failures, 1-13
initialization, 1-4
link, 1-13
NetIPC, 1-10, 1-11
RTE-A resources, 1-7
software, 1-5
software revisions, 1-8
table sizes, 1-7
TCP, 1-11
tracing, 5-1
user applications, 1-10

U

upgrade level, 2-18, 2-28
UPLIN, 9-18, 9-39, 9-60
user record ID, 2-20

X

X.25, 2-28, 9-51
link troubleshooting, 1-5
links, 9-53
message processing, 9-23
message records, 7-1
MPE, 9-48