DECnet/E
System Manager's Guide

Order No. AA-H505B-TC

January, 1982

This manual describes the network related tasks that a DECnet/E System Manager performs along with the software capabilities available to manage the network.

OPERATING SYSTEM AND VERSION:	RSTS/E	V7.1
SOFTWARE VERSION:	DECnet/E	V2.0

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

digital equipment corporation · maynard, massachusetts

First Printing, January 1982

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

Copyright © 1982 by Digital Equipment Corporation

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

The following are trademarks of Digital Equipment Corporation:

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

Distributed and Mid-Range Systems Publications typeset this manual using DIGITAL's TMS-11 Text Management System.

310ALL

Contents

Chapter 3 Controlling and Monitoring the Network

Chapter 4 DECnet/E Network Control Program

Figures

Tables

Preface

Manual Objective

This manual describes what a system manager should know to manage the Phase III DECnet product, DECnet/E V2.0.

Intended Audience

In practice, many people may be responsible for carrying out the task of managing a network. For the sake of simplicity, however, these people are referred to as "system manager" or "network operator." In this manual, these terms refer to any DECnet/E user, privileged or nonprivileged, or any network operator from another DECnet node who wants to communicate with DECnet/E using a network management program.

DECnet/E operates on the RSTS/E (Resource Sharing Time Sharing/Extended) system. Before reading this manual you should be familiar with the networking concepts presented in the *Introduction to DECnet*, and you should have a working knowledge of the RSTS/E system.

NOTE

It is recommended that you read this manual before reading the other DECnet/E manuals so you become familiar with general network operations. Once you begin controlling your network on a daily basis you should know the procedures described in the *DECnet/E Network Installation Guide.*

- Required reading for the effective use of this manual:

  DECnet/E Release Notes
  Order Number: AA-M269A-TC

  Introduction to DECnet
  Order Number: AA-J055B-TK

  RSTS/E
  System Generation Manual
  Order Number: AA-2669E-TC (original manual)
  Order Number: AD-2669E-T1 (update)

  RSTS/E
  System Manager's Guide
  Order Number: AA-2762D-TC

- Related references include the following manuals:

  DECnet/E
  Network Installation Guide
  Order Number: AA-K714A-TC

DECnet/E
Guide To User Utilities
Order Number: AA-H504B-TC

- General references include the following manuals:

  DECnet/E
  Network Programming in
  BASIC-PLUS and BASIC-PLUS-2
  Order Number: AA-H501B-TC

  DECnet/E
  Network Programming in
  COBOL
  Order Number: AA-H503B-TC

  DECnet/E
  Network Programming in
  FORTRAN
  Order Number: AA-L266A-TC

  DECnet/E
  Network Programming in
  MACRO-11
  Order Number: AA-L265A-TC

- An overview of the DIGITAL Network Architecture (DNA) is provided in the following manual:

  DIGITAL Network Architecture
  General Description
  Order Number: AA-K179A-TK

- Compliance with the DNA Network Management Functional Specification and the Network Management Minimal Subset ensures compatibility with other DECnet Phase III systems. The following functional specifications provide detailed descriptions of Phase III DNA:

  DNA Data Access Protocol (DAP) Functional Specification
  Order Number: AA-K177A-TK

  DNA Digital Data Communications Message Protocol (DDCMP)
  Functional Specification
  Order Number: AA-K175A-TK

  DNA Maintenance Operations Protocol (MOP) Functional Specification
  Order Number: AA-K178A-TK

  DNA Network Management Functional Specification
  Order Number: AA-K181A-TK

DNA Network Services (NSP) Functional Specification
Order Number: AA-K176A-TK

DNA Session Control Functional Specification
Order Number: AA-K182A-TK

DNA Transport Functional Specification
Order Number: AA-K180A-TK

Structure of This Manual

The DECnet/E System Manager's Guide is both a tutorial and reference manual. The tutorial portions of text emphasize the changes needed to update DECnet/E to a Phase III product. The reference portions of text contain the major revisions of the NCP (Network Control Program) commands and show how to run NCP.

Because Phase III is unlike the previous release, the tutorial portions of text are for both new and previous users of DECnet/E. Previous components have been removed (NCU no longer exists), and new capabilities have been added. Although the network management utility is still called the Network Control Program (NCP), the command syntax is different from the previous release and new commands have been added.

How to Use This Manual

The manual is divided into two parts. Part I of this manual provides a general description of the features and capabilities needed to manage your network and includes the following chapters:

- Chapter 1 Introduction
- Chapter 2 DECnet/E Concepts and Entities
- Chapter 3 Using Network Management Capabilities

Part II of this manual shows how to run NCP and provides a reference for NCP commands:

- Chapter 4 DECnet/E Network Control Program

If you have never used DECnet/E, read the first three chapters before using the NCP commands.

If you have used previous versions of DECnet/E NCP, read Chapter 2, which describes several features that are new with Phase III DECnet. If you have any questions after you read Chapter 2, read Chapter 3 for further information on how to use network management capabilities.

If you are familiar with Phase III DECnet and the RSTS/E operating system and you have used other DECnet NCPs, you may need to use only sections of

Chapter 2 for reference. These include:

Routing	(Section 2.1)
Nodes	(Section 2.2)
Circuit and Line Glossary	(Section 2.3)
Multipoint	(Section 2.4)
Lines	(Section 2.5)
Circuits	(Section 2.6)
Objects	(Section 2.7)
Event Logging	(Section 2.8)

Until you know NCP commands well, use Chapter 4 for reference. Afterwards, you can use the command summary in Appendix A as a reference guide for NCP. Refer to the Glossary for a description of any unfamiliar terms.

Documentation Conventions

UPPERCASE	Uppercase letters indicate keywords you must type as shown. (You can abbreviate keywords to the first three letters of each keyword.)
lowercase italic	Lowercase italic letters represent items you must replace according to the accompanying description. When an item needs a description of more than one word, the words are hyphenated, for example, file-spec.
[ ]	Brackets enclose optional parameters that can be omitted. (Parameters not enclosed in brackets must be included.) In general, brackets are logical symbols only and should not be included in the input. However, brackets surrounding a project-programmer number (ppn) must be included.
red	User input in examples is printed in red to distinguish it from system output.
CTRL/	The character string "CTRL/" followed by a letter indicates a control character (for example, CTRL/Z). Control characters are generated by depressing the control key and the appropriate letter key at the same time and are echoed on the console display as a caret (^) followed by the letter (for example, ^Z).

Unless shown otherwise, spaces or tabs must separate items in a command string. More than one space or tab between items is treated by the system as a single space. Also, although it is not shown in this manual, you must follow each command string with a carriage return.

Chapter 1
Introduction

This chapter provides a general description of DECnet networks and the network management utility used to control them. It also describes the major tasks a system manager must perform.

1.1 DECnet Networks

DECnet refers to the hardware and software that connects different DIGITAL systems to form networks. All DECnet implementations are based on a design structure called the DIGITAL Network Architecture (DNA). The structure ensures that implementations providing fewer features will always work with implementations providing more.

This manual describes the DECnet implementation for DIGITAL's RSTS/E operating system. DECnet/E provides all the capabilities of a normal RSTS/E system plus the networking capabilities described in this manual and the companion DECnet/E manuals listed in the Preface.

If your network contains operating systems other than RSTS, you should check the DECnet documentation for those systems to make sure the network capabilities you need are common to those systems. (Refer to the *Introduction to DECnet* for a brief description of the capabilities for all DECnet systems.)

1.2 A Few Terms You Should Know

Network — Refers to the family of software modules, parameter files, hardware components, and facilities that tie different DIGITAL systems together. The software modules and parameter files make up the DECnet software (see Section 1.3).

Node — Refers to a DIGITAL computer system that uses DECnet software to link the system with other DIGITAL computer systems in a network. The nodes in the network must be running either

Phase II or Phase III DECnet. The following terms are used to describe nodes in the context of network management:

Executor node	The node that executes commands.
Local node	The node your terminal is plugged into.
Remote node	Any node in the network other than the local node. A remote node could be in the same room as the local node.
Adjacent node	A remote node that is one hop away from the local node (see Section 2.1).

Consider the sample network shown in Figure 1–1.

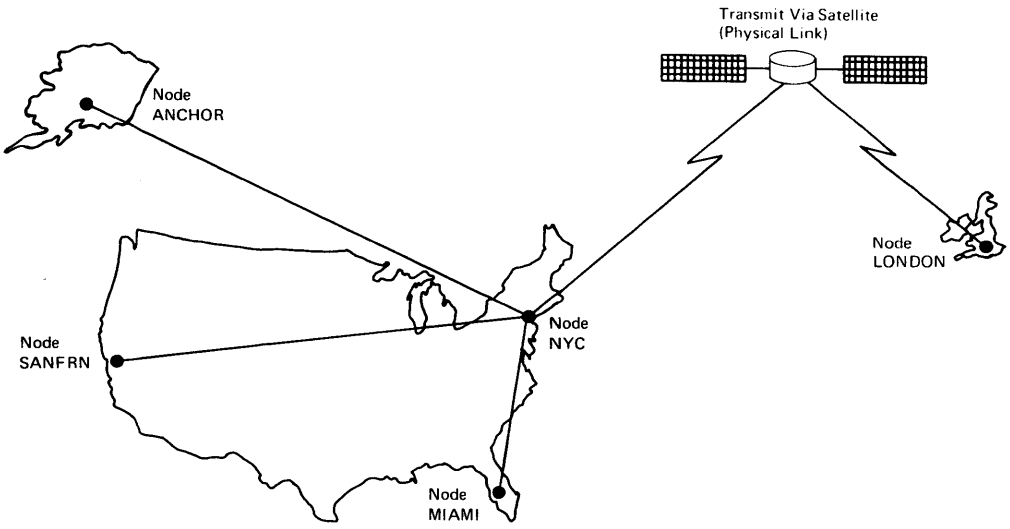


Figure 1–1: Computer Network Composed of Five Nodes

In this example, node NYC is adjacent to nodes SANFRN and MIAMI. If node NYC is the local node, then all other nodes in the network are remote. Unless specified otherwise, (see Section 2.9), you enter and execute commands at the local node.

1.3 The DIGITAL Network Architecture

The design of all DECnet systems is based on the DIGITAL Network Architecture (DNA). DNA is a software structure that consists of layers. Each layer defines a set of network capabilities and a set of rules, called protocols, to carry out these capabilities.

DECnet/E is implemented according to a set of protocols defined by DNA: DDCMP, Transport, NSP, NICE, and DAP. Refer to the *DIGITAL Network*

Architecture General Description for more information.

- DDCMP, Transport, and NSP are used for general network operation.

  DDCMP (Digital Data Communications Message Protocol) provides physical link control in all DECnet systems by making sure that data flows sequentially and error-free. DDCMP uses the Cyclic Redundancy Check (CRC) to detect errors, retransmits messages to recover from errors, and numbers data segments to ensure sequential transmission of data.

  Transport defines the rules that determine the physical path or route along which data travels to its destination.

  NSP (Network Services Protocol) defines the rules for communication between programs in a network over paths called logical links. Logical links allow many different data streams to travel to the same receiving node. These streams of data are then separated at the receiving node and delivered to the appropriate program. (Refer to the DECnet/E programming manuals for more information on logical links.)

- NICE (Network Information and Control Exchange) is used for network management. It defines mechanisms for exchanging network, node, and configuration data, and for servicing requests from software modules residing in the Network Management layer.

- DAP (Data Access Protocol) is used for file access. It defines mechanisms for performing remote file access and transfer on behalf of software modules residing in the User layers.

1.4 Overview of DECnet/E Structure

DECnet/E features for network communication are part of the RSTS/E system monitor. The manner in which these features are accessed differs with the type of user.

User programs interface with DECnet/E through NSP (Network Services Protocol). Although it is not a separate entity or program in DECnet/E, the term NSP is used throughout this manual to refer to the software that allows a user program to establish and exchange data over logical links. The user interface to this software is also known as "session control," where a "session" is the process of creating, using, and disconnecting a logical link. The RSTS/E programmer accesses NSP's session control services by coding calls to subroutines that access monitor system routines. Note that some session control functions are not accessible by BASIC-PLUS programs.

Terminal users access certain network capabilities through various DECnet/E utility programs. The utilities TLK (Talk) and LSN (Listen) allow two terminal users to type messages to each other. The utilities NFT (Network File Transfer) and FAL (File Access Listener) work together to allow net-

work file manipulation. NFT and FAL are implemented according to the Data Access Protocol (DAP) so that files can be exchanged with other DECnet systems having the same capabilities. The NET (Network Command Terminal) utility allows the terminal user to log into and use a remote RSTS/E system as if the local terminal were directly connected to the remote system.

Some elements of DECnet/E are not directly accessible to any user. These lower level elements of DECnet/E include the network Transport module (TRN) that implements the Transport protocol, several hardware devices (the DMC11, DMR11, DMV11, and DMP11), and their associated software device drivers.

The device drivers are responsible for handling messages received from and sent over the physical communication lines. NSP gives a message to be sent to Transport, which selects an appropriate data path based on the destination of the message. Transport then passes the message to the proper device driver for actual transmission. The drivers manipulate the device registers to cause message transmission to occur.

To handle incoming messages, the drivers include buffer management functions that allow messages to be received from the physical links. When a message is received, it is passed to Transport, which checks the destination of the message. If the message is for the local node, Transport passes it to NSP for further processing. If the message is destined for another node in the network, Transport selects an outgoing data path and passes the message to one of the drivers for transmission.

The hardware devices are intelligent communication controllers that connect the PDP-11 to physical communication lines. The DMC11 and DMR11 devices control lines that connect to only one other system in the network (point-to-point lines), and the DMV11 and DMP11 devices control lines that can connect to more than one other system (multipoint lines). The physical lines can include cables, modems, telephone circuits, and satellite transmission facilities. Each controller contains a microprocessor, its own memory, and microcode (firmware) that implements the DDCMP protocol. Using DDCMP in firmware considerably reduces the software overhead for the DECnet/E system.

1.5 Your Role As System Manager

As a system manager, you use the features provided by NCP (Network Control Program) to monitor and control the various entities (circuits, events, lines, nodes, and objects) in your network.

NCP is the highest level in the Network Management layer in the DIGITAL Network Architecture. With NCP, you can configure, test, and control a network containing DECnet Phase III nodes and perform limited testing with Phase II nodes.

NCP accepts command input from a user terminal and then creates, modifies, and queries the permanent parameter file $NETPRM.SYS and the vol-

atile parameter file [0,1]NSP0.SYS. You create the permanent parameter file with the DEFINE commands and the volatile parameter file with the SET SYSTEM command. The SET SYSTEM command loads the values established in the permanent parameter file into the volatile parameter file. Permanent values remain constant until cleared or reset and are used only as initial values for the volatile parameters. Volatile values reflect up to the minute network status (see Section 2.10).

In addition, NCP displays status information on network operation with the SHOW/LIST commands.

As a system manager, you can also monitor and control your network at any node (not only the local node) by establishing a remote node as the executor node. Under these circumstances, the Network Management Listener (NML) is invoked on the remote node. (NML is installed as a permanent network program on all Phase III nodes and is started automatically as the result of a request from NCP on another node.) Further commands entered at the local user terminal are formatted by NCP at the local node according to the NICE protocol and transmitted to NML for execution on the remote node. Refer to Section 2.9 for more information on remote command execution.

In addition to the operations you perform to manage a RSTS/E system, to manage a DECnet/E node you must perform the following operations:

- Set up the volatile and permanent parameter files
- Start up your DECnet/E node
- Monitor your DECnet/E node
- Monitor remote DECnet/E nodes

See the *DECnet/E Network Installation Guide* for information on setting up the parameter files and starting up your nodes. See Chapter 3 of this manual for information on monitoring your nodes.

Chapter 2
DECnet/E Concepts and Entities

This chapter describes DECnet/E concepts and entities. The information will help you use the Network Control Program (NCP) utility described in Part II of this manual.

To understand DECnet/E network management, you should familiarize yourself with the following concepts and entities:

Routing	Section 2.1 summarizes the network capability that determines the physical path along which data travels to its destination.
Node Identification	Section 2.2 specifies the formats for identifying nodes and describes the use of node parameters.
Multipoint	Section 2.4 describes the capability for controlling network activity on multipoint lines.
Lines	Section 2.5 specifies the format for identifying lines and describes the use of line parameters.
Circuits	Section 2.6 specifies the formats for identifying circuits and describes the use of circuit parameters.
Objects	Section 2.7 specifies the format for identifying objects and describes the use of object parameters.
Event Logger	Section 2.8 describes the DECnet/E event logger and the use of event logging parameters.
Remote Executor Nodes	Section 2.9 describes the techniques used to enter commands at a "local" command node and cause them to be executed at a specified "remote" executor node.

Access Control	Section 2.9.3 describes requirements, formats, and techniques for controlling user access to nodes.
Network Parameters	Section 2.10 describes the permanent and volatile parameter files.

2.1 Routing

Routing is the network capability that selects the physical path or route along which data packets travel to their destinations. This section provides a brief explanation of routing and the routing parameters you control. For a more detailed explanation of the concepts behind routing see the *Introduction to DECnet.* See the *DNA Transport Functional Specification* for a detailed description of the routing architecture.

2.1.1 Routing and Nonrouting Nodes

In terms of routing, DECnet/E recognizes two types of nodes:

- Routing nodes (Phase III nodes only)
- Nonrouting nodes (Phase III end nodes and all Phase II nodes)

Routing Nodes

A routing node is a Phase III node that can receive packets intended for another node and forward them to a neighbor (adjacent node) in a logical communications path. *Forwarding* occurs when a routing node accepts a packet to pass on to the destination node. In Figure 2–1, NYC, a routing node, can route messages from SANFRN to MIAMI and from MIAMI to SANFRN. Although NYC acts as the routing node through which the other two nodes communicate, SANFRN and MIAMI are not aware of NYC's involvement in the communication process. In a network consisting of only Phase III nodes, any node can send packets to any other node in the network.

The following terms describe Phase III DECnet routing:

Hop	The distance from one node to a neighbor (adjacent node).
Path	The route a packet takes from source node to destination node.
Circuit	The logical communications path between two nodes along which a packet is forwarded. In multipoint configurations a single line, or physical path, supports one or more circuits. In point-to-point configurations a single line supports a single circuit.
Path length	The number of hops in a logical communications path between two communicating nodes. Path length is the

number of circuits a packet must go through to reach its destination.

Cost — An arbitrary integer value (1-25) you assign to a circuit. Each circuit has a separate cost. Nodes on either end of a circuit can assign different costs to the same circuit.

Path cost — The sum of the circuit costs along a path between two nodes. Packets are forwarded on paths with the least cost.

Reachable node — A destination node for which the DECnet/E routing module has determined a usable path exists.

Nonrouting Nodes

There are two types of nonrouting nodes. All Phase II nodes are nonrouting nodes, and Phase III nodes with a single communications circuit (that is, Phase III nodes that are end nodes) are nonrouting nodes. End nodes are aware only of their neighbor (adjacent node). If adjacent to a Phase III routing node, an end node can communicate with other nodes in the network by using the forwarding capability of the routing node. In Figure 2–1, SANFRN, a nonrouting node, can send packets to nodes NYC and MIAMI and receive packets from these nodes, but cannot route packets from NYC to MIAMI.

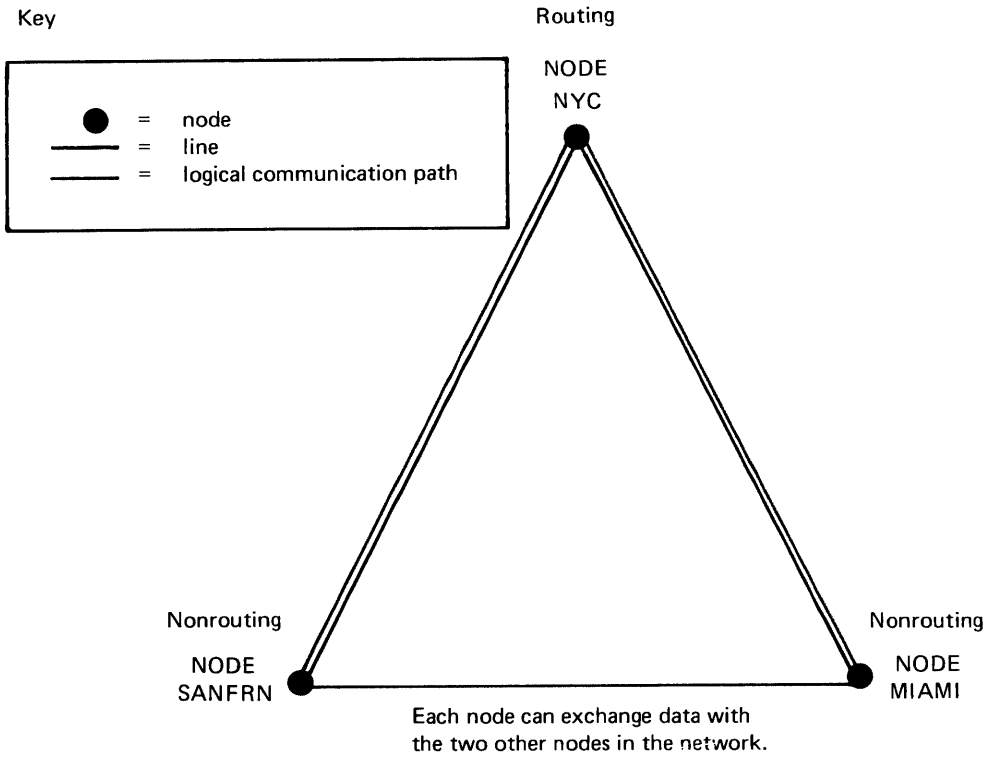


Figure 2–1: Routing and Nonrouting Nodes

Phase III nodes with one circuit are automatically initialized by DECnet/E as nonrouting nodes at network startup. (See the *DECnet/E Network Installation Guide* for details on network startup.)

Phase II nodes can communicate only with adjacent Phase II or Phase III nodes. Therefore, Phase II nodes should be placed in the network carefully. For example, because Phase II nodes recognize only adjacent nodes, a Phase III node cannot forward packets through another Phase III node to communicate with a Phase II node. Likewise, a nonrouting node connected to a Phase II node can communicate only with that Phase II node.

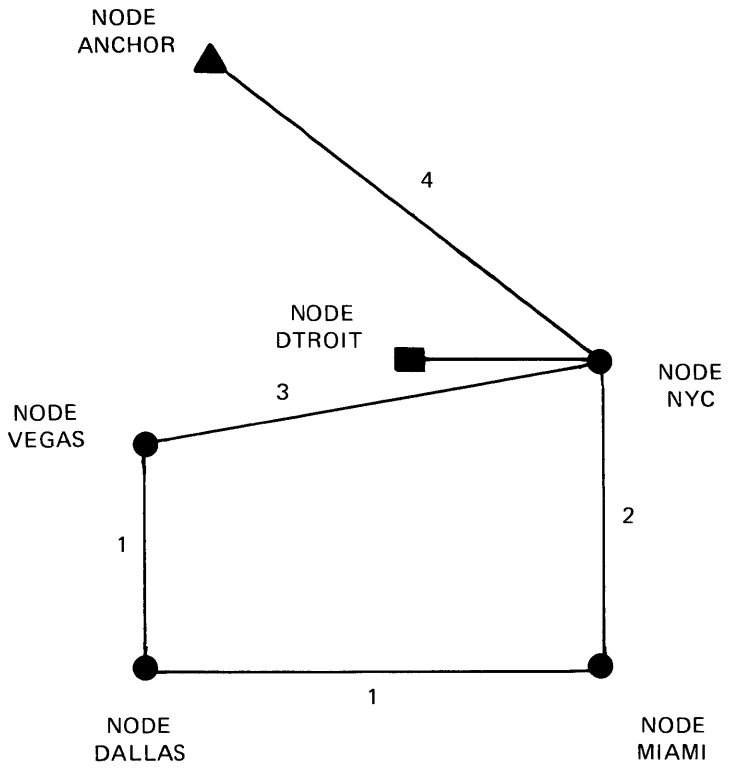
2.1.2 Transport's Role in Routing

The Transport module in each routing node controls routing. Routing parameters are stored in the routing data base located in the extended data buffer (XBUF).

Based on the routing parameters in XBUF, Transport calculates the path length and path cost from the local node to each possible destination node in the network. It performs this calculation for each circuit that is available for logical link traffic. Transport then finds the circuit with the least costly path. The routing data base at each routing node contains the results of these calculations.

Whenever Transport receives a packet to be forwarded, it examines the routing data base and sends the packet on the path of least cost. If the destination is unreachable, Transport discards the packet. Every time a value that affects routing (a routing parameter) changes, all routing data bases are updated with routing messages sent throughout the network. In this way, routing data bases at each node reflect the most current information such as which lines are up, the current line costs, and which nodes are reachable. Routing always directs traffic on the path of least cost.

In Figure 2–2, there are two possible paths Transport can take to route packets from node ANCHOR to node DALLAS. Here, the path of least cost is through nodes NYC and MIAMI (a cost of 7). As long as the NYC to MIAMI circuit remains up, this is the preferred path because the path through NYC and VEGAS has a higher cost (a cost of 8). However, if the NYC to MIAMI circuit goes down, packets will be routed through nodes NYC and VEGAS.



Line costs are defined at local Node NYC in this figure.

Key

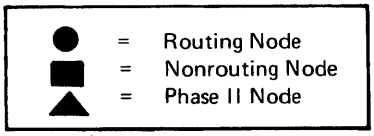


Figure 2–2: Network Line Costs and Node Types

2.1.3 Routing Parameters

As a system manager, you assign the routing parameters listed below. You can set or change the following parameters with the NCP commands SET/DEFINE EXECUTOR and SET/DEFINE CIRCUIT. See the *DECnet/E Network Installation Guide* for information on node generation guidelines.

Parameter	Description
Cost	Cost is an arbitrary integer value assigned to a circuit between two adjacent nodes. Cost affects the traffic flow over a logical connection (circuit) between a source node and a destination node. Path cost is the sum of all circuit costs over which the packet is transmitted.
Maximum Address	Maximum address specifies the highest node address value used in the network. The maximum circuits parameter and the maximum address parameter determine the size of the routing data base.
Maximum Circuits	Maximum circuits specifies the number of circuits in the routing data base for the executor node. No more than the maximum number of circuits can be ON at any given time, although more than the maximum number may be configured on the system. The parameter files can handle up to the maximum number of circuits that are configured in the hardware; this value can be greater than the maximum number of circuits in the routing data base. If maximum circuits is 1, the node is configured as an end node (a nonrouting node).
Maximum Cost	Maximum cost is the maximum path cost allowed from the executor node to any other network node. A remote node is considered *unreachable* if the cost of every possible path to that node exceeds the value established for this parameter. However, a "live" remote node should not be unreachable if the cost is set correctly. A node can be unreachable for other reasons, for example, when a circuit is down or a node is OFF.
Maximum Hops	The value for maximum hops is the network *diameter*. A remote node cannot be reached over a path if the number of hops needed to get to the node exceeds the value established for this parameter.

2.2 Node Identification, States, and Counters

To use the Network Control Program (NCP) utility you need to identify nodes and to understand node states.

2.2.1 Node Identification

Many of the parameters you control need the identification of a specific node. You identify a node with a unique node name or number. The form of node identifiers depend on the command keyword you use. The NCP SHOW commands display the volatile parameters and the LIST commands display the permanent parameters.

To satisfy routing requirements, each node in the network must have a unique address. Because it is often easier to remember a name rather than an address, a name can be associated with that address at any time. The addresses are known to the entire network. The names are known to the local system only. In the list of node names maintained on local nodes, each node name must be unique. To avoid confusion, it is recommended that you consider the names as "network" names and set up the same name-address correspondence on all nodes in the network.

NOTE

DECnet/E requires that you assign names to all nodes with which you want to establish logical links. All DECnet utilities and network access functions documented in the DECnet/E programming manuals require that you reference nodes by name. Only NCP allows you to reference nodes by address.

Node Identification Parameters:

node address — A node address is a unique decimal integer (in the range 1 to MAXIMUM ADDRESS) assigned to a node at network generation. The maximum upper limit for node addresses in DECnet/E is 255.

node name — A node name is a one to six character, alphanumeric string that contains at least one alphabetic character.

node-identifier — A *node-id* is either a node address or a node name.

You can identify a node by typing NODE followed by the node's name or address. The executor node can also be identified by typing EXECUTOR.

When you want a command to apply to more than one node, you can use a plural node identification. KNOWN NODES are all nodes that either have a

name or address or are reachable. DECnet Transport on a given executor assigns two states to remote nodes, *reachable* and *unreachable*. The executor perceives nodes as either reachable or unreachable by reading the routing data base. ACTIVE NODES are all known nodes the executor perceives as reachable. On an end node, ACTIVE NODES is the executor node, the adjacent node, and all nodes that currently have links open to that end node.

A loop node is not really a node, but is a way of referring to a circuit (the loopback target). A loop node is used to force traffic to the specified circuit for testing purposes. A pseudo-node name is defined to refer to the loopback target, so the syntax is similar to that of other loop commands. Loop nodes are always considered reachable because they refer to the executor node, which is always reachable. Section 3.4 provides a more extensive discussion of the different kinds of loopback commands.

Alias Node Names

As a privileged user, you can use NCP to assign alias node names to destination nodes. Only one alias can be assigned to each node. This simplifies remote node identification. Alias node names are known only to the local node. The network maps a user request containing an alias node name to the network destination node name and address and then establishes the logical link.

Assigning and Changing Alias Node Names

You can assign and change alias node names with the SET/CLEAR NODE *node-id* ALIAS command.

The following example shows how you might use the alias capability: you have a program called PAYROL that accesses a node named LISBON. LISBON is down, and you must run the PAYROL program. However, you have another node named PHILLY that can also support the PAYROL program.

To enable PAYROL to *talk* to the other node, you can assign the name PHILLY as an *alias* for the node LISBON. By using the alias capability in this way, you do not have to change the program. However, you must first clear the primary node name PHILLY so you can use the name as an alias (an alias node name must be unique).

The commands to accomplish this are:

```
NCP> CLEAR NODE PHILLY NAME
NCP> SET NODE LISBON ALIAS PHILLY
```

If necessary, you could change the alias PHILLY to another alias. You can, for example, use the following command:

```
NCP> SET NODE PHILLY ALIAS ROME
```

ROME is now an alias for LISBON.

2.2.2 Node States

The executor node can be in one of four states:

ON	Allows logical links.
OFF	Does not allow new logical links. Packet forwarding and any data flow in progress is immediately terminated. All circuits are set OFF, and all existing logical links are aborted.
SHUT	Does not allow new logical links, but existing logical links remain in effect until they stop normally. Forwarding through this node continues normally. When all logical links that begin or end in the executor node are gone, the node goes OFF.
RESTRICTED	Does not affect forwarding through this node, yet no new incoming logical links from other nodes are allowed. Outgoing logical links are allowed.

You can change executor node states with the SET EXECUTOR STATE command.

Before issuing any SET command, the SET SYSTEM command must first be issued to load the values from the permanent parameter file $NETPRM.SYS into the volatile parameter file [0,1]NSP0.SYS.

1. SET EXECUTOR STATE ON causes NSP to start.
2. SET EXECUTOR STATE OFF stops network operation immediately. Because you can lose important data in the process, this command should be used only when it is vital to stop network operation immediately.
3. SET EXECUTOR STATE SHUT causes a gradual network shutdown and should be used under normal circumstances because all logical links are completed before the network shuts down.

The NETOFF utility provides a timed shutdown procedure similar to the RSTS/E SHUTUP utility. (Refer to Section 3.6 for more information.)

2.2.3 Node Counters

DECnet software automatically maintains certain statistics for nodes in the network. These statistics are known as node counters. Such information includes the number of connect requests sent and received, and the number of messages sent and received. This information can be used alone or with logging information to evaluate the performance of a given node.

Executor node counters are always available. They represent data counts for local logical links (those between two programs on the executor node) and error counts for the NSP and Transport modules. Remote node counters are kept while links are active to a node. Remote node counters can be deleted when no links are active to release system resources. Whenever a node counter block is deleted, the counter values are logged, so the data is not lost. Appendix B provides a complete list of node counters and their descriptions. See Section 2.8 for a discussion on logging.

Setting the Logging Frequency

You can use NCP to establish the frequency with which counters are logged and set to zero. At any point while the network is running, you can also display node counter statistics using the SHOW NODE *node-id* COUNTERS command.

To set a timer whose expiration automatically causes the node counters to be logged at the logging sink (location) and then set to zero, use either the SET EXECUTOR or the SET NODE command with the COUNTER TIMER parameter. For example:

```
NCP>SET EXECUTOR COUNTER TIMER 60
```

The preceding example causes a node counter logging event to take place every 60 seconds for the local node.

The following example specifies that counters for remote node VEGAS are to be logged at the local node every 60 seconds.

```
NCP>SET NODE VEGAS COUNTER TIMER 60
```

Note that these counters are maintained on the executor node. To clear the COUNTER TIMER parameter, enter the CLEAR EXECUTOR or CLEAR NODE command and specify the COUNTER TIMER parameter.

Setting Node Counters to Zero

At any point when the network is running, you can set node counters to zero for a given remote node, the executor (local) node, or all known nodes. Use any of the following commands to set node counters to zero:

```
NCP>ZERO NODE BOSTON COUNTERS
NCP>ZERO EXECUTOR COUNTERS
NCP>ZERO KNOWN NODES COUNTERS
```

2.3 Circuit and Line Glossary

Circuits and lines should not be confused. A circuit is a logical point-to-point connection. A line is a physical link or wire over which circuits can be established. A single circuit between two nodes corresponds to a single line only when those two nodes are connected by a point-to-point line. On a multipoint line, where there can be more than two nodes on a single line (see Fig-

ure 2–3), several circuits can be established over the single line – one circuit between each slave and the master. You can turn ON as many circuits as necessary, up to the maximum number of circuits allowed by the routing data base for the executor node. The routing data base has an upper limit of 16 circuits.

A multipoint configuration is one where one physical line allows multiple logical circuits by connecting a master (control station) with slaves (tributaries). On lines controlled by the DMP or DMV, the master controls the slaves by polling them. Section 2.4 describes the multipoint configuration in more detail.

The following terms are used to describe the multipoint process:

Station	The station describes the type of device. For example, you can operate either a multipoint control, multipoint tributary, or point-to-point station.
Controller	A controller is a hardware device that controls activity on a physical line. It resides on the system bus and its operations are directed by a device driver. A single DMP or DMV is a hardware controller.
Master	A master is the station controlling the multipoint line. A master controls the slaves.
Control station	A control station is a multipoint circuit that is equivalent to the master.
Tributary	A tributary is a station on a multipoint line that acts like a slave rather than a master.
Slave	A slave is the same as a tributary. Tributaries act as *slave* stations because they can send and receive data only to and from the master, and only when polled by the master.
Polling	A master continually polls its slaves by sending request messages to each tributary address in the polling list.
Tributary address	A master uses a numeric address to poll the tributary. You can set and change these addresses during network operation with the commands SET/DEFINE CIRCUIT TRIBUTARY *address.*This address is interpreted by the peripheral equipment on the multipoint line to determine which hardware interface should respond. Note that the tributary address is different from the node address and circuit number.
Circuit number	A circuit number is a decimal number that is part of the identification of a circuit on a multipoint line. The number is used internally to identify the circuit. For example, DMP-1.3 is a circuit-id that refers to circuit

number 3 on DMP-1. The node to which the executor is connected by this circuit can be named BOSTON and have a node address of 27. However, the tributary address can be 123 which is the number recognized by the hardware that controls the connection. Note that the circuit number, node address, and tributary address are unrelated and can all be different.

2.4 A Multipoint Configuration

A multipoint configuration can be thought of as a "comb" where the teeth of a comb represent the many communication links or circuits the control station can have with its tributaries (see Figure 2–3).

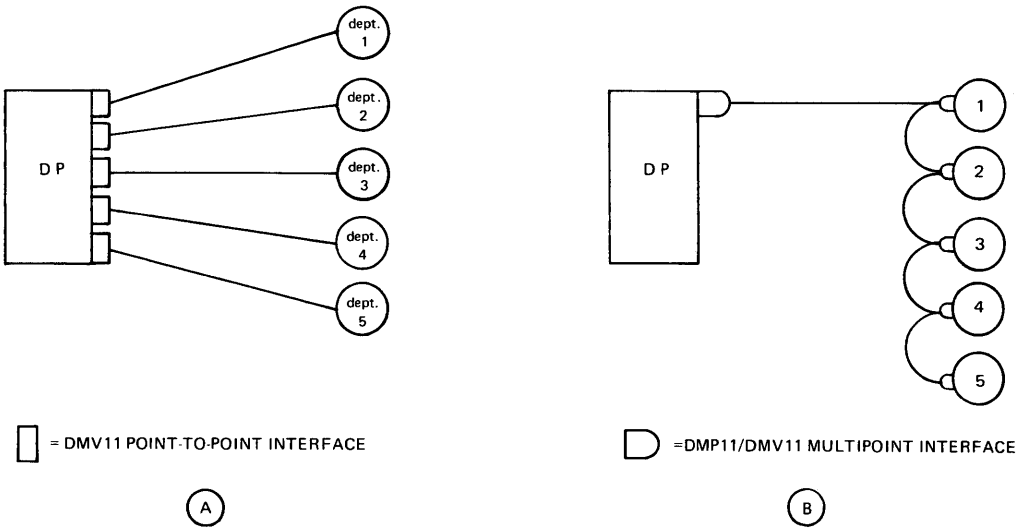


Figure 2–3: A Multipoint Configuration

The master manages all communication between itself and the tributaries. Each tributary knows only its own address and can communicate only with the master when it is polled.

2.4.1 The Multipoint Process

In DECnet/E, DMP and DMV devices control multipoint operations. These devices can also be used in point-to-point mode. In point-to-point mode, the DMP device is equivalent to a DMC device. If you want a line to be used exclusively in point-to-point mode, a DMP will do, but a DMC is probably a more cost-effective choice. Because the DMV is the only communication device for the LSI-11 bus (Q-bus), the DMV is the only choice for both point-to-point and multipoint line types.

A multipoint line has one master and up to 16 (12 if the master is a DMV) active tributaries. The master periodically polls each tributary to determine whether the tributary has any data to transmit.

The master transmits any data it may have for a tributary at the time it polls the tributary. A tributary is always given the opportunity to transmit data back to the master at that time.

A polled tributary with no data to send automatically returns an ACK (positive acknowledgment) to inform the master that it is still *alive*. When tributaries stop responding to the poll, they are polled less frequently.

Each time the DMP/DMV driver receives information from a tributary (the driver knows each tributary-address), the driver checks the internal list of circuit numbers to find out which circuit corresponds to the tributary from which the driver received data.

NOTE

To save time, you can assign the lowest *circuit-ids* to the most active multipoint circuits. In this way, the driver does not spend unnecessary time checking a lengthy list of circuit numbers.

You should be aware that to the DECnet Transport layer above the driver level, a multipoint line looks exactly like a set of independent point-to-point lines. Transport deals only with circuits and performs the same processing whether the circuits are combined on a single multipoint line or are all on separate point-to-point lines. The driver then determines which line the circuit corresponds to and what operations are needed for the data to be transmitted.

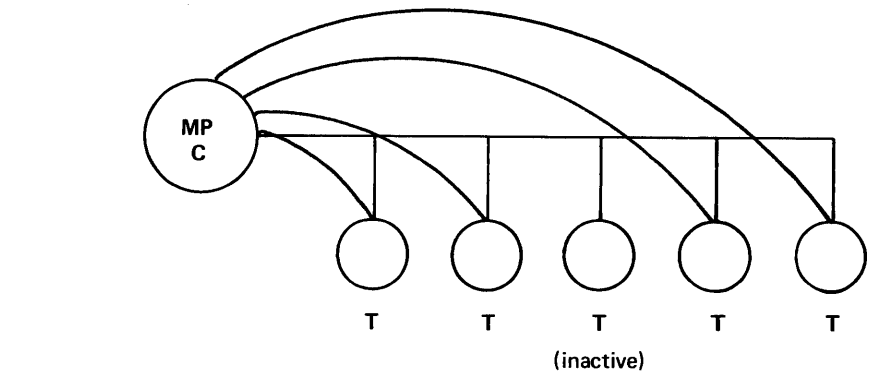
A multipoint tributary can be either a routing node or an end node (see Section 2.1). A multipoint master with two or more tributaries must be a routing node. If a program at one tributary wishes to communicate with a program at another tributary, it sends a connect initiate message (see the DECnet/E programming manuals) containing the other tributary's node-id. This message, and all other data flowing between the two tributaries, is forwarded by Transport at the control station to the two nodes involved.

2.4.2 A Multipoint Application

Multipoint is useful when a number of systems need to be connected, and the amount and frequency of data transfers are relatively low. In such cases, the use of multipoint allows substantial savings in hardware and cabling costs.

Consider a situation in which a company has a number of small departmental computers and one large central system in the data processing department. The small systems are mostly autonomous but periodically transfer

their data to the central data processing system for further processing. In this situation, two configurations are possible:



Key

MPC = Multipoint Controller
T = Tributary
—— = Line
—— = Circuit

Figure 2–4: A Multipoint Application

In configuration B, the large system needs only a single line controller rather than five. Also, less cable is needed if the small machines are in close proximity.

On the other hand, B has a lower throughput because the five small systems share a single line. Also, a failure of the DMP device on the large system affects all five small machines rather than only one (though one more DMP device as a backup would remedy that).

2.5 Lines

DECnet/E supports point-to-point and multipoint line devices.

- A point-to-point line is one line connecting two adjacent nodes.
- A multipoint line connects a master node (control station) with slave nodes (tributaries).

For data exchange, a multipoint line acts as a set of logical lines or circuits between the master and each slave. A point-to-point line is treated as a single circuit (the one between the two nodes on the line). In a multipoint configuration both the master and slaves are part of the same physical line. A node can have both point-to-point and multipoint lines connected to it at the same time. However, on any one multipoint line, a node can serve as either the control station or tributary station, but not both (see Section 2.4).

2.5.1 Line Identification

As a system manager, you can control and monitor many aspects of line activity described in this manual. The commands you use to perform these operations must include a line identification.

You set up parameters for lines with the SET/DEFINE LINE commands. A line identifier pertains to a physical line. DECnet/E uses DMC/DMR device drivers to communicate in point-to-point mode and DMP/DMV device drivers to communicate in multipoint mode. It is also possible to run DMP/DMV in point-to-point mode (see Section 2.4).

You identify a line by typing the keyword LINE followed by the *line-id*. A *line-id* is a unique character string [or name] and consists of 1 to 16 alphanumeric characters. The general format is:

dev-c[*-u*]

where:

dev	The type of communication line, for example, DMC or DMP.
c	A decimal number (0 or a positive integer) designating the line's controller.
u	A decimal unit or line number (0 or a positive integer) included if the controller supports multiple communication lines. DECnet/E does not support any device multiplexers, so this field is not used. However, you might encounter this field if communicating with a different system.
wildcard	An asterisk (*) that replaces a controller, unit number, or tributary. The asterisk specifies all known lines in the range indicated by its position in the line identification. You can use more than one wildcard in a line identification.

Examples of valid line identifications:

DMC-0	DMC11, controller 0
DMP-*	DMP11, all controllers
DMV-3	DMV11, controller 3

Examples of invalid line identifications:

*	This is an attempt to identify all devices.
DMC-0.1	This is a *circuit-id* and not a *line-id*.
XM1:	XM1: is a valid device name for the RSTS monitor, but is not valid for DECnet programs.

2.5.2 Line Counters

DECnet software automatically maintains line counters. Line counters report the amount of time since the counters were last set to zero. All line counters previously defined in DECnet/E V1.1 for DMC/DMR devices are now known as circuit counters. Only DMP/DMV devices now have line counters. The line counters keep track of errors that are specific to the device and cannot be attributed to any of the circuits on the line. The information line and circuit counters report can be used to measure the performance for a given line. Section 2.6 discusses circuit counters in more detail. Appendix B provides a complete list of line and circuit counters and their descriptions.

At any point while the network is running, you can display line counter statistics using the SHOW LINE line-id COUNTERS command.

At any point when the network is running, you can zero line counters for a given line or for all known lines. Use the following commands to zero line counters:

```
NCP>ZERO LINE DMP-0 COUNTERS
NCP>ZERO KNOWN LINES COUNTERS
```

2.6 Circuits

You establish circuit ownership when you invoke the SET/DEFINE CIRCUIT OWNER EXECUTOR commands. When you turn a circuit ON, DECnet tries to assign itself circuit ownership if you have not already done so. However, if a non-DECnet user previously assigned the circuit for non-DECnet use, you would not be able to obtain ownership of the circuit. Therefore, it is recommended that you invoke the circuit ownership commands when you set up the parameter file.

NOTE

A line does not have an independent state. A line goes on when the first circuit on the line is set to the ON state. A line goes off when the last circuit on the line is set to the OFF state.

The following parameters determine circuit states and substates:

STATE Establishes the circuit's operational state at the executor node. The possible states you may set are as follows:

- ON The circuit is available for normal network operations.
- OFF The circuit is not available for use by any network or network-related software.

The two substates for the ON state are as follows:

ON

STARTING — Starting indicates "trying to establish contact." If the Starting mode takes more than ten seconds, there is a problem. For example, you could have a verification mismatch, or the circuit might not be set to the ON state at the remote node.

RUNNING — Running indicates normal operating mode. In this mode, if you send packets across the physical line they will reach their destination. This substate is not displayed with the SHOW command.

2.6.1 Circuit Identification

As a system manager, you can control and monitor many aspects of circuit activity described in this manual. The commands you use to perform these operations must include a circuit identification.

You set up parameters for circuits with the SET/DEFINE CIRCUIT commands. A circuit identifier pertains to a logical line.

You identify a circuit by typing the keyword CIRCUIT followed by the *circuit-id*. A *circuit-id* is a unique character string (or name) consisting of 1–16 ASCII characters. The general format is:

dev-c[*-u*][*.t*]

where:

dev — The type of communication circuit, for example, DMP or DMV.

c — A decimal number designating the circuit's controller.

u — A decimal unit number (0 or a positive integer) included if the controller supports multiple communication lines. DECnet/E does not support any device multiplexers, so this field is unused. However, you might encounter this field if you are talking to a different system.

t — A decimal number identifying a tributary on a multipoint line. This is a logical circuit number, not to be confused with the tributary address. (In a multipoint configuration, the tributary address is the identification a tributary responds to when polled by the controller.)

wildcard — An asterisk (*) that replaces a controller, unit number, or tributary. The asterisk specifies all known circuits in the range

indicated by its position in the circuit identification. You can use more than one wildcard in a circuit identification.

Examples of valid circuit identification:

DMC-0	DMC11, controller 0
DMP-1.5	DMP11, controller 1, tributary 5

Examples of invalid circuit identification:

DMP-1	No tributary is specified
DMP-1.45	Tributary number exceeds the range
DMP-*.1	Wildcards must be rightmost
DMP-2.*	DMP11, controller 2, all tributaries
DMP-*.*	DMP11, all controllers, all tributaries

2.6.2 Circuit Counters

DECnet software automatically maintains circuit counters. Information provided by circuit counters includes: the number of data packets sent, received, and lost over the circuit; the number of data and buffer errors; the number of timeouts; and the amount of time since the counters were last set to zero. This information can be used to measure the performance for a given circuit. Appendix B provides a complete list of circuit counters and their descriptions.

At any point while the network is running, you can display circuit counter statistics using the SHOW CIRCUIT circuit-id COUNTERS command.

Example:

```
NCP
NCP> SHOW CIRCUIT DMC-0 COUNTERS
Circuit Volatile Counters as of 17-Nov-81 14:23:57

Circuit = DMC-0

       >65534  Seconds Since Last Zeroed
         4846  Terminating Packets Received
         4810  Originating Packets Sent
            0  Terminating Congestion Loss
        49758  Transit Packets Received
        57260  Transit Packets Sent
            0  Transit Congestion Loss
            0  Circuit Down
            0  Initialization Failures
      3557925  Bytes Received
      3539337  Bytes Sent
        62831  Data Blocks Received
```

```
        68705  Data Blocks Sent
            0  Data Errors Inbound
            0  Data Errors Outbound
            0  Remote Reply Timeouts
            0  Local Reply Timeouts
            2  Local Buffer Errors, including:
                Naks Sent, Buffer Unavailable

NCP>
```

2.7 Objects

When one task or network job communicates with another, DECnet identifies the jobs or tasks as two "objects" communicating. Objects are programs on the network that are identified by name or number. The network software links objects using the NSP protocol during logical link communication.

As a system manager, you supply information for two general types of DECnet/E objects when setting up the network:

- Objects with a zero object type. These are usually user-defined programs for special purpose applications. They are named when a connection is requested. It is recommended that you keep a list of the names of these objects to prevent duplication. The object type number for all of these objects is 0. You can define an object of type zero in the DECnet/E parameter files, but it is recommended that you do not because an object of type zero cannot be started automatically.
- Objects with a nonzero object type. These are programs that provide specific network services. For example, FAL, the File Access Listener (described in the *DECnet/E Guide to User Utilities*), and NML, the Network Management Listener (described in Chapter 1 of this manual), fall in this category. The object type number serves as a standard addressing mechanism that identifies programs throughout the network by purpose or function rather than by name. Object type numbers for all nonzero objects range from 1-255.

You can control four parameters related to objects:

- Object name
- Object type
- P1 (optional)
- P2 (optional)

2.7.1 Object Names and Object Types

You identify objects with a name (object name), a number (object type), or both.

An object name is a one to six character descriptor for a program, for example, ASH, KASEY, and HARPO. This name corresponds to the receiver-id for the program (see the Declare Receiver call description in any of the DECnet/E programming manuals). The name can include alphanumeric characters and the special characters "$", ".", and "_" (dollar, dot, and underline). Object names should be unique within the local node.

An object type is a decimal integer in the range 1-255. Object types 1-127 are reserved for use by DIGITAL. Object types 128-255 are available for user applications programs. Appendix D provides a list of standard DECnet object numbers and the names associated with them. An object must be defined for each program that is to be automatically started by NSP. Refer to the DECnet/E programming manuals for a description of how one program can form a network connection to another program by object type. Because all of the standard DECnet/E utilities use this method of forming a connection, you must define the standard objects for NSP.

Object types simplify the identification of programs performing identical functions throughout the network in various nodes without going through the cumbersome process of arriving at, and enforcing, standard naming conventions. Use the object type parameter to specify a unique object number for nonzero objects as shown in the following example.

```
NCP>SET OBJECT 17 NAME FAL FILE $FAL.TSK
```

The example above identifies the FAL object because the number is 17. Note that the object name is not required and might not be the same across the network. Thus, you could assign a name other than FAL to this object although doing so would be potentially confusing.

```
NCP>SET OBJECT 17 NAME NETWRK
```

The example above identifies the standard FAL object type with the name NETWRK. For consistency, it is recommended that you use standard object names because they are normally referenced network-wide.

Use the SET/DEFINE OBJECT commands to define and modify these parameters. Use the CLEAR/PURGE OBJECT commands to delete any or all parameters from the volatile and permanent parameter files.

2.7.2 P1 and P2 Parameters

The optional parameters, P1 and P2, specify parameters needed for a program invoked by NSP.

For objects written in BASIC-PLUS and BASIC-PLUS-2, P1 is the line number where program execution begins. (See the description of the Concise

Command Language (CCL) in the *RSTS/E BASIC-PLUS Programming Manual.*) For objects written in the other languages, P1 is either passed to the program or unused.

For all languages, P2 is a general-purpose parameter passed to the program and either used for program-specific information or unused.

If you supply the parameters keyword, you must supply values for both P1 and P2. If P1 is used and not P2, a value of 0 must be specified for P2, and vice versa.

2.8 EVENT LOGGER

The network software logs certain events that occur during network operation. An event is defined as a network- or system- specific occurrence for which the logging component maintains a record. A partial list of significant events includes:

- Node counter activity (when they are reset)
- Circuit and node states (when they change)
- Error conditions associated with routing (for example, unreachable nodes or oversized packets)
- Data transmission errors (for example, "partial routing update loss" occurs when you receive a routing message from a neighbor [adjacent node] that specifies a reachable node whose node number exceeds the range specified by your MAXIMUM ADDRESS parameter. This condition occurs when your neighbor's MAXIMUM ADDRESS parameter is larger than yours).

Figure 2–5 shows conceptually how the event logger operates.

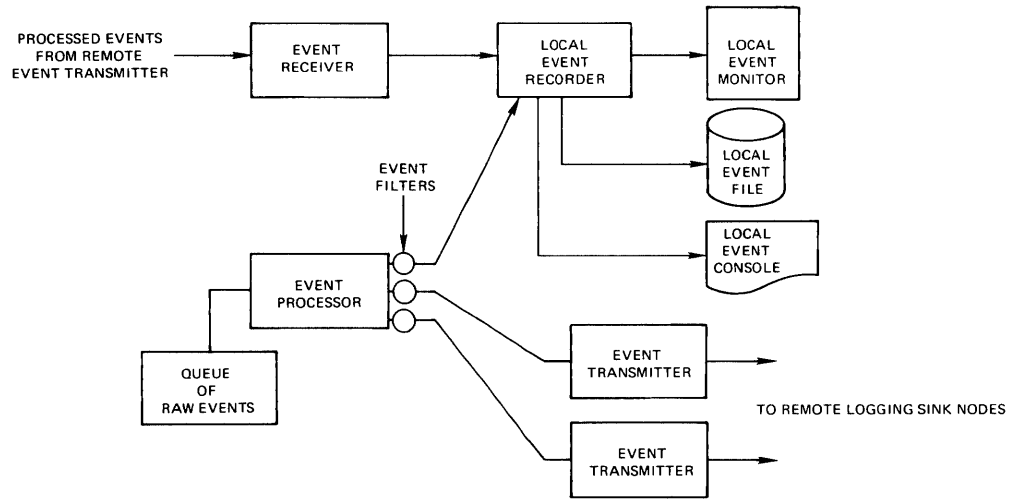


Figure 2–5: Event Logger Model

Events are stored in event queues at each DNA layer. As the queues fill, the events are passed to the Event Processor.

The Event Processor filters events according to assigned class and type (see Section 2.8.2). The term *filter* refers to both the physical data structures you create to record events and the process performed to screen the events. The filtering process eliminates logging for some events you consider to be of no interest, so that events do not clutter up the queue. As system manager, you define the filters you wish to apply for each separate logging target or "sink."

The Event Processor filters events to the appropriate logging sink nodes. A logging sink node is a node where event information is sent to be recorded and stored. A sink is a storage area, on a logging sink node, for recording events. You can log an event to three types of sinks: a console, a file, and a monitor. You can log to sinks on your own node and to sinks on remote nodes. Other remote nodes can send events to your event receiver to be logged on your local sinks. The following list briefly decribes the three types of logging sinks. Refer to the *DNA Network Management Functional Specification* for more information.

1. A logging console is a terminal that records events on a logging node. The default logging console is KB0:. Appendix F describes the format of event messages for the logging console component.

2. A logging file is a user-specified file on a logging node. The DECnet/E logging file receives events in the standard NICE protocol format. The default file is $EVTLOG.LOG.

3. A user-supplied logging monitor program, specified by receiver name (see Section 2.7), receives and processes events in NICE protocol format. This program can receive event data and possibly take some action based on this information. The default monitor name is EVTMON.

DECnet keeps events in separate event class queues before they are passed to the Event Processor for filtering. Events are time-stamped when they occur. Because the Event Processor does not necessarily receive events in the order they occur, it is important for you to consider the time stamp for each event when you analyze output from the event logger.

The Event Processor Receiver, EVTLOG, runs as a detached job under RSTS/E and is started up using NCP. However, you can also run the same program with a RUN $EVTLOG command. Entered in this manner, the program acts as a utility that accepts a logging sink file generated by the Event Processor, containing events in NICE protocol format. Using this file, $EVTLOG produces a human-readable file in the format used for console output. See Section 2.8.8 for information on dumping the event file.

2.8.1 Setting Logging Parameters

As system manager, you are responsible for controlling certain aspects of event logging. In particular, you can control source-related parameters (actual events to be logged and the location at which these events are logged) and sink-related parameters (the name of the logging component at the local node and its operational state).

Use the SET/DEFINE LOGGING commands to establish your logging parameters at network startup. Continue to use these commands when you want to redefine or modify the parameters during network operation.

Use the CLEAR LOGGING command to remove any or all parameters from the volatile parameter file, and use the PURGE LOGGING command to remove any or all parameters from the permanent parameter file.

To identify the name of the logging device on the local node, use the NAME parameter. For example, if the device is a logging file, the following command creates the file NET.LOG for event logging:

```
NCP>SET LOGGING FILE NAME [6,254]NET.LOG
```

Regardless of the logging component you use, parameter selection is the same. If you want to modify parameters for all logging sinks on the executor node, use the KNOWN LOGGING entity identifier when issuing the LOGGING commands as:

```
NCP>SET KNOWN LOGGING EVENTS 4.2
```

2.8.2 Identifying Events

Events are defined by class and type. Appendix E lists all possible events by class and type. Events are recorded from the various DNA layers. You can specify the types of events to be recorded by using the following event list format:

class.type

where:

class Identifies the DNA layer to which the event pertains

type Identifies a form of event, unique within an event class

For example, to specify an event in the Transport layer, you would use class 4. The event types for this class range from 0 through 14. Event type 0 indicates aged packet loss, event type 1 indicates unreachable node packet loss, and so on.

Use the EVENTS entity identifier for the SET/DEFINE LOGGING commands to specify those events to be logged. If you want to log all event classes and types recognized by the executor use the KNOWN EVENTS entity identifier. When defining the logging entity, you must specify events to be logged.

When providing an event list for the EVENTS entity identifier, you can specify only one class. However, several formats exist for defining event types for a class. You can specify a single event type, a range of types, or a combination of the two. Note that types must be specified in ascending order within the range specified. The following examples illustrate these formats:

Event list	Meaning
4.4	Identifies event class 4, type 4
4.5-7	Identifies event class 4, types 5 through 7
4.5,7-9,11	Identifies event class 4, types 5, 7 through 9, and 11

The following two examples illustrate invalid event lists:

```
NCP>SET KNOWN LOGGING EVENTS 4.4,5.1

NCP>CLEAR KNOWN LOGGING EVENTS 4.7,4-2,1
```

The first example defines more than one event class. The second example defines event types in numerical descending order rather than ascending order.

A wildcard character (*) can be specified for the event type. The following example illustrates the use of wildcards:

Event list	Meaning
2.*	Identifies all event types known for class 2 events

The following examples illustrate invalid uses of the wildcard character:

```
NCP>DEFINE LOGGING FILE EVENTS *.2-5

NCP>PURGE LOGGING FILE EVENTS 3.2-*
```

The first example specifies event types for all classes. Unless you use the KNOWN EVENTS parameter, you can specify only event type information for a single class. The second example uses a wildcard to specify a partial range of event types. The wildcard character indicates the entire range of event types for a given class.

2.8.3 Logging Events

A single event can be logged to three nodes, the local node and two remote nodes. This gives you the option of logging an event to nine possible locations (sinks): the console, file, and monitor on the local node and on each remote node. All sinks are independent. Thus, you can log events to a

remote logging sink regardless of whether you are logging them locally. However, if the local sinks are not used, you are still limited to two remote nodes (six sinks).

Use the SINK parameter to specify the node at which you want to log an event. If you do not specify a node, the executor node is the default. The following example routes all event information to the logging monitor program running on node DENVER.

```
NCP>SET LOGGING MONITOR SINK NODE DENVER
```

Only locally generated events are sent to a remote sink. Remotely generated events that you might receive go only to local sinks. Therefore, the sender of the event determines which sinks are used. The only control the receiving event recorder has is to turn a sink OFF as stated in the following command:

```
SET LOGGING CONSOLE STATE OFF
```

2.8.4 Controlling the Operational State of Logging

You can control the operational state of logging sinks for the local node only. There are two possible logging states:

ON — This state indicates the logging sink is available for receiving event information. This is the normal operational state.

OFF — This state indicates the logging sink is unavailable for receiving event information. In this state, events are not logged for that sink. Any events arriving from a remote node directed to a sink that is OFF are discarded without warning.

Use the STATE parameter to turn ON or OFF individually all defined logging sinks that have a name at the local node. The following example causes the event logger program to be started. If an event logger is already running at this node, the newly started version notifies the system console and exits. Note that this control over logging does not affect the operational state of the node.

```
NCP>SET ALL LOGGING STATE ON
```

2.8.5 Losing Events

Events can get lost (fail to be logged) for a variety of reasons. Conditions that would cause this to happen include the following:

- The routing path to the sink node exceeds the maximum path length.
- The sink to which you want to log an event is in the OFF state.
- The sink to which you want to log an event is not found.
- A circuit is down and there is no other path to the sink node.

In the above cases, lost events messages are not generated.

Events can also get lost when the input queue for an event type is filled. In the monitor, each class of events has a separate queue that holds a maximum of ten events at a time. The following procedure describes how input queue overflow at the logging source is treated:

If an event occurs and the input queue is filled, a lost events message is generated. This message is logged as a single event. It indicates the time when the first event loss occurred. If more events are lost before the lost events message can be reported to the Event Processor, this fact is not recorded. Therefore, each lost events message indicates only that one or more events were lost. Lost events messages received after the initial message is sent are considered new lost events messages.

For example, you receive your first lost events message. It reports that the first event was lost at 3:00 PM. You then receive another lost events message indicating that the first event was lost at 3:30 PM. Therefore, each lost events message is received individually.

There is also an output queue for each logging sink holding filtered events of several classes and types. If this queue overflows a lost events message is also generated.

2.8.6 Starting the Event Logger

If the logging STATE in the permanent parameter file is specified as ON by a DEFINE ALL LOGGING STATE ON command, use the SET SYSTEM command to enable event logging. For example, the following command, by activating the event logger, enables events to be logged.

```
NCP>SET SYSTEM
```

Note that the SET SYSTEM command initiates all network activity.

If you need to restart event logging during network operation, use the commands SET ALL LOGGING STATE ON and SET ALL LOGGING STATE OFF.

2.8.7 Troubleshooting with the Event Logger

If you have a problem entity (that is, a circuit, line, or node) in your network, you can use this special type of logging for troubleshooting.

Because events can get lost when event queues reach their thresholds, you can eliminate unnecessary event logging by specifying the *source qualifier* option in the NCP command, SET LOGGING. This way, you can analyze only those events that occur at the source of the problem entity.

For example, if you have a problem with circuit DMC-4, and you want to make sure events on circuit DMC-4 do not get lost, you can enter the following command:

```
NCP>SET LOGGING  sink-type  EVENTS 0.* CIRCUIT DMC-4  sink node
```

By entering this command, only events of class 0 occurring on circuit DMC-4 are reported.

Restrictions for Source Qualifier Logging

Because the purpose of this type of logging is for troubleshooting only, there are restrictions imposed.

- You can specify only one source qualifier per event class. However, by specifying an event list (see Section 2.8.2) you can apply the source qualifier to multiple types within that class. For example, you can isolate a range of event types within event class 0 for line DMC-1 with the following command:

  ```
  NCP>SET LOGGING  sink-type  EVENTS 0.1-3 LINE DMC-1  sink node
  ```

- If you are going to SET qualified event types within a class, you must first CLEAR those unqualified event types within that class. The same event cannot be logged as qualified and as unqualified. In other words, if you specify a source qualifier for a range of event types within an event class, that range of event types within that class cannot be logged elsewhere. In the example above, if you use the source qualifier option to isolate event types 2-5 in class 0 for line DMC-1, you cannot log event types 2-5 to any other sink.

- You can use the source qualifier option to log events to multiple sinks, but only on one sink node. Therefore, you can log events to three sinks (the console, file, and monitor) on a sink node.

The display for events specified with the source qualifier option is the same format as the display for unqualified events. See the SHOW LOGGING command for details on the type of information displayed.

2.8.8 Dumping the Event File

When the logging sink is a file, the event logger stores events in the NICE protocol format specified for event messages in the *DIGITAL Network Architecture, Network Management Functional Specification, Version 3.0.0*. The structure of this file, with sample records, can be found in Appendix F. When run by a privileged user, the event logger prints one of these files in human-readable form, the same format used by the Event Processor to display events on the console sink.

Example:

```
$ RUN $EVTLOG

Enter logfile name <.LOG>: $EVTLOG

Event type 4.14, Node reachability change
Occurred 15-Nov-81 16:16:07.7 on node 135 (ROME)
Node address 91 (LONDON)
Reachable


Event type 480.0
Occurred 15-Nov-81 16:16:17.3 on node 135 (ROME)
Node address 143 (PARIS)
Parameter  2130 = 23456
```

In this mode, the program does not act as the event logger and does not interfere with the "live" event logger on the system. In this mode, the program only reformats the event file.

It is recommended that you periodically close the event logger's current event file and open a new file by issuing a SET LOGGING FILE NAME command that specifies the name of the new file. Then, $EVTLOG can be used to produce a human-readable copy of the file, or you can write a program to report on selected information from the file. This procedure is necessary because a file being updated by the event logger should not be used to produce this formatted dump.

2.9 Remote Executor Nodes

The SET EXECUTOR NODE command and the TELL prefix allow you to control or monitor several remote nodes. They allow you to enter certain NCP commands at the local node for execution at a remote node. The only commands that cannot be executed remotely are CLEAR EXECUTOR NODE, SET EXECUTOR NODE, SET SYSTEM, CLEAR SYSTEM, EXIT, and the TELL prefix. Depending on the remote system you are communicating with, access control information might be required.

The SET EXECUTOR command, the TELL prefix, and access control information are discussed in the subsections that follow.

2.9.1 The SET EXECUTOR Command

When NCP is invoked, the default executor is always the local node. To execute commands remotely, you issue the SET EXECUTOR NODE command and then specify the remote node you want to act as executor. It is possible for several users at a single node to set their NCP to a different executor.

You may have to supply access control information when you set a remote node as the executor. Section 2.9.3 describes this process. Clearing the executor resets the access control to the local node.

The level of privilege allowed at the remote executor node depends on the access control information it requires. The level of privilege when the command node is the executor depends on the account you are logged in under.

Once you set a remote node as the executor, any additional commands that you type (with the exception of those named) are executed from that node. NCP forms a logical link connection with NML, the Network Management Listener, at the remote node. NML logs in with the access control information you supply or, if you do not supply access control information, to a default account. NCP translates any commands you type into the NICE protocol format (see the *DNA Network Management Functional Specification*) and sends them to the remote NML for execution. (User programs can also communicate with a remote NML using this protocol.) When the executor is a remote node, DECnet/E NCP prompts with "executor::NCP>" to indicate which node is executing the commands. For example, the following prompt indicates that SANFRN is the executor node:

```
SANFRN::NCP>
```

The executor node interprets all NCP commands with its own network management programs and gets any required default parameters from its own data base. The format of the output is determined by the designated executor node. For example, if remote node SANFRN is named the executor node, any commands executed there are treated as if SANFRN were the local node.

To reset the executor back to the local node, type CLEAR EXECUTOR NODE. This also causes NCP to revert to its usual prompt, NCP>. When you restart NCP, the executor automatically reverts to the local node.

The executor also automatically reverts to the local node when you exit NCP.

As shown in Figure 2–6, you can set the executor node using the following NCP command:

```
NCP>SET EXECUTOR NODE LONDON [access control information]
```

The NCP commands that you subsequently enter at your local node are then executed at the remote executor node. The executor node interprets each command through the Network Management Listener (NML) and then performs the requested function. Each command must be stated as if it were entered to NCP directly at the remote executor node. Any information output as a result of the execution of a command is displayed at the local node. Refer to the appropriate DECnet documentation if the remote node is running a DECnet implementation other than DECnet/E.

To reset the executor to the local node, type the following command:

```
NCP>CLEAR EXECUTOR NODE
```

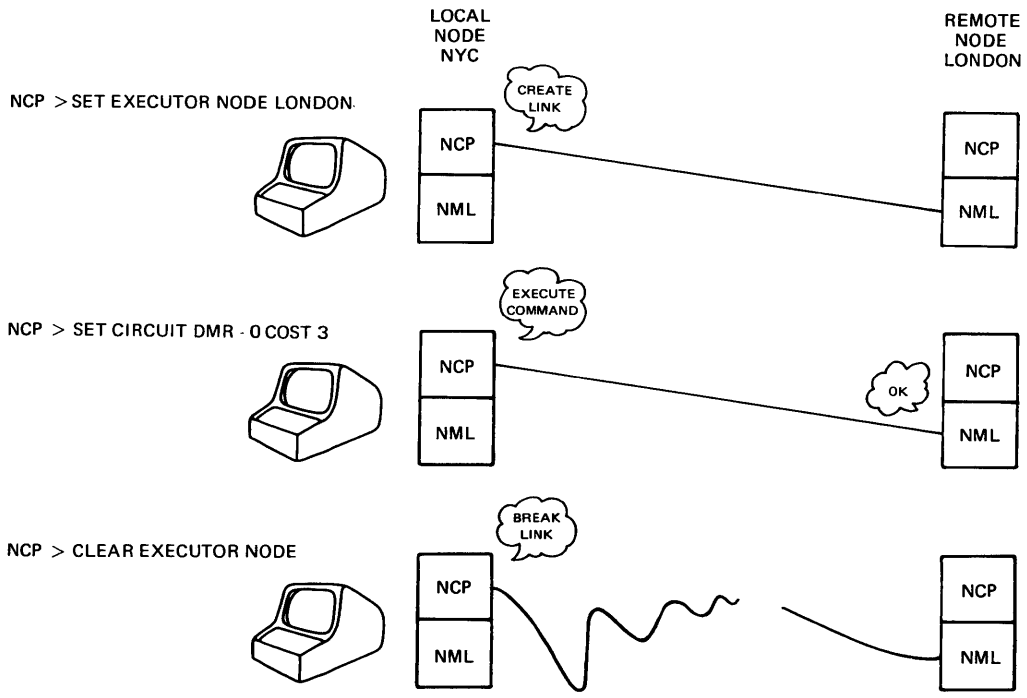


Figure 2–6: Remote Command Execution

NOTE

The local NCP is always responsible for creating and breaking the link with the remote node.

2.9.2 The TELL Prefix

TELL is a prefix and not a command. It is meaningful only when placed before a command. To execute only a few commands at a remote node or to override the currently set executor, issue a command with the TELL prefix. For example:

```
TELL TAHOE [access control information] SHOW ACTIVE LOGGING
```

Remote execution applies only to the one command entered with the TELL prefix. The TELL prefix is useful if you want to execute only a few commands from a remote node because control automatically returns to the currently set executor after the commands are invoked. However, if you want to enter several commands for remote execution, it is easier and more efficient to use the SET EXECUTOR NODE command instead of repeatedly using the TELL prefix.

As with the SET EXECUTOR NODE command, you may need to supply access control information each time the TELL prefix is used.

2.9.3 Access Control

Access control information identifies you to a program on a remote node. If you want to communicate with a program on a remote node, some access control information might be required to create a link with that program. Although the DECnet/E system does not require use of these fields, some DECnet programs do require access control information. For example, the NFT program (see the *DECnet/E Guide to User Utilities*) prompts for the following information:

```
NFT>FILE.MAC=ROME::FILE.MAC

User:  [2,172]

Password:  (not echoed)

Account: B72058
```

FAL logs into this account on the remote system to access the remote file. Each remote system's rules about file and directory protection determine which files can be accessed by FAL running under the account supplied.

NFT prompts for this information, but for NCP access control information should be included in the command line. For example:

```
NCP>SET EXECUTOR NODE ROME USER [1,100] PASSWORD PASTA

NCP>TELL ROME USER [1,100] PASSWORD PASTA SET CIRCUIT DMC-1 STATE ON
```

Supplying an account on ROME allows NML, the Network Management Listener, to log into the remote system. A privileged account is required to allow NML to perform privileged network management operations on the remote node, such as turning circuits ON and OFF.

SHOW and LIST are nonprivileged network management commands on most DECnet systems. DECnet/E NCP can define a local unprivileged account that NML logs into when no other account is supplied with a command, so unprivileged commands can be executed. Any command that changes the system state is privileged.

The format for access control information is:

[USER *user-id*] [PASSWORD *password*] [ACCOUNT *account number*]

Information that you specify is used in the following ways:

1. The *user-id* identifies the person or program requesting the logical link. In RSTS/E terms, the user-id is the *project-programmer number* used at log-in.
2. The *password* is often used to verify access to a system under a user identification. In RSTS/E, this password is used with the *project-programmer number* at log-in. Here, you supply a password acceptable to the remote node or program, if required.
3. The *account number* is often used in addition to the user identification and password as a billing account number. In DECnet/E, this field is normally unused.

Access control fields not required should be left blank.

2.9.4 Failure Conditions

DECnet/E V2.0 has a new type of failure mode to report problems involving corruption of the dynamic routing data base. This failure mode can be described as a "controlled network hang" and is intended to allow crash-type data to be gathered without crashing the entire RSTS/E system.

Example:

```
File corrupted
?Bad directory for device (error text)
```

This error message indicates a corruption of the routing data base.

The routing data base is accessed dynamically by monitor-level components of DECnet/E, so its corruption means that safe network operation cannot proceed. When the monitor-level components of DECnet/E detect the corruption of this data base, they stop processing network traffic and return a single error code, "BADDIR", to all network function calls EXCEPT for the NCP command SET EXECUTOR STATE OFF, and log an event of type 36.0 (Routing data base corruption).

At this point, use the RSTS/E V7.1 SNAP command of the UTILTY program to produce the equivalent of a crash dump that includes all dynamic DECnet/E structures. Copy the CRASH.SYS file and the [0,1]NSP0.SYS file to some secondary area so they can be returned with the SPR. Shut the network down with the SET EXECUTOR STATE OFF command, issue the CLEAR SYSTEM command, and restart the network in the usual way.

2.10 Network Parameters

The circuit, line, logging, node, and object parameters that define your network are relatively permanent because changing them changes the way a node functions in the network.

Some examples of changing parameters are:

- Setting a circuit state to ON
- Changing a node name associated with a node address
- Setting the routing cost for a circuit
- Setting a node to receive certain logged events

Parameters may be set either as temporary (volatile) values or permanent values.

2.10.1 Permanent Parameter File

You establish the permanent parameter file ($NETPRM.SYS) with the DEFINE commands at network installation. Use the DEFINE and PURGE commands to change the permanent parameter file during network operation. Use the LIST commands to display the contents of this file. (See Chapter 4 and the *DECnet/E Network Installation Guide* for more information.) Permanent parameters take effect each time the SET SYSTEM command is issued.

2.10.2 Volatile Parameter File

At network startup, you establish the volatile parameter file [0,1]NSP0.SYS by loading parameters from $NETPRM.SYS with the SET SYSTEM command. [0,1]NSP0.SYS is a RSTS-installed file similar to a swap file. It is accessed whenever circuit states are changed, network objects are spawned, logging parameters are changed, and connections (inbound or outbound) are initiated by network programs. [0,1]NSP0.SYS is also accessed when SHOW commands are executed to display its contents.

NCP allows you to change the file name and the device where the volatile parameter file will be installed. For instance, you may want to install it on a high-speed swapping disk to optimize network performance.

Use the SET/CLEAR commands to change the volatile parameter file during network operation. Use the SHOW commands to display network status during network operation. Volatile parameters take effect immediately.

See the *RSTS/E System Manager's Guide* for information on listing the volatile parameter file (also called the DECnet/E system file) and the *RSTS/E Programming Manual* for information on adding, removing, and listing the volatile parameter file.

2.10.3 Parameter Restrictions

The DEFINE/PURGE commands and the SET/CLEAR commands perform complementary operations. However, certain line and node parameters can-

not be cleared or purged individually. These restrictions are noted in Chapter 4 in the command descriptions.

Some changes cannot be made on an active entity. For example, a line cannot be changed from full-duplex to half-duplex while its circuit state is ON. Some volatile parameters (the local node name, for example) can be changed only when the network is shut down. These restrictions are also noted in Chapter 4 in the command descriptions. In addition, the SET/CLEAR EXECUTOR NODE and SET/CLEAR SYSTEM commands do not have a DEFINE/PURGE counterpart.

Chapter 3
Controlling and Monitoring the Network

Chapters 1 and 2 describe the concepts necessary to understand network management and to set up the permanent parameter file. Given this background, this chapter describes network management from the point of view of running the network. With this information, you should be able to start your DECnet/E node, monitor its network activity, and shut it down in an orderly way.

3.1 Starting Your DECnet/E Node

Refer to the *DECnet/E Network Installation Guide* for information on how to complete the network installation. The network installation procedure includes setting up the permanent parameter file to define the configuration of your network. To do this, you must first invoke NCP by typing RUN $NCP or the CCL command, NCP. Then set up the permanent parameter file ($NETPRM.SYS) with NCP's DEFINE commands.

After you set up the permanent parameter file, you should perform the following operations whenever you want to start the network on your node.

1. Type RUN $NCP or the CCL command, NCP, to invoke NCP.
2. Type the command "SET SYSTEM".
3. Type the command "SET EXECUTOR STATE ON".

It is recommended that you include these commands in the system startup file, so your network is enabled automatically whenever you start your RSTS/E system.

The SET SYSTEM command automatically establishes the volatile parameter file [0,1](NSP0.SYS) by loading everything contained in $NETPRM.SYS into volatile storage. The SET SYSTEM command can also automatically enable event logging. If you need to restart or stop event logging during network operation, use the commands SET ALL LOGGING STATE ON and SET ALL LOGGING STATE OFF.

If you want to make temporary changes to your network characteristics, without entering them in the permanent parameter file, you should use the SET commands after issuing SET SYSTEM and before issuing SET EXECUTOR STATE ON.

Once you issue the SET EXECUTOR STATE ON command, your node is ON. You can then use NCP to control the operational states of network components (for example, circuit states), both local and remote. In effect, you can dynamically reconfigure your network to control the use of resources. Use the SET/CLEAR commands to reconfigure the volatile parameter file during network operation. Use the DEFINE/PURGE commands to reset the permanent parameter file, at any time, during network operation or after network shutdown.

3.2 Monitoring a Running Network

You can monitor network activity in two ways: by means of NCP's SHOW/LIST commands and by means of the event logging facility. This section discusses the use of the SHOW/LIST commands. See Section 2.8 for a discussion of events and event logging.

3.2.1 Displaying Information with NCP

Network parameters are described either as "status" or "characteristics." Parameters that indicate status contain dynamic information (such as circuit state) that changes automatically while various activities are taking place in the network. Parameters that indicate characteristics (such as logging parameters at a specified sink node) are static in the sense that once set, either at network startup or during network operation, they normally remain constant until cleared or reset.

NCP provides two commands for displaying information about network parameters: SHOW and LIST. The SHOW command displays information from the volatile parameter file about the status of the running network. The LIST command performs a similar function except it lets you display and verify information from the permanent parameter file.

The SHOW command allows you to monitor network operation in terms of the various dynamic changes that take place. For example, whenever the state of a circuit changes, this may change the configuration of the running network in terms of reachable and unreachable nodes.

When issuing the SHOW and LIST commands, you can select various keywords to display specific types of information. You can choose between several display options, depending on the information you want. The display option determines the format and type of information that NCP displays.

Display options are as follows:

ALL	This display option includes all characteristics, counters, and status information. Depending on the entity identifier, this may include all line characteristics or all logging events.
CHARACTERISTICS	This display option includes static information usually specified in the permanent or volatile parameter file. Depending on the entity identifier, this may include the identification of the local node and relevant routing parameters or the names and numbers of known network objects.
COUNTERS	This display option provides counter information for circuits, lines, and nodes, including the local node. Because they are inherently volatile, counters can be displayed only with the SHOW command. Appendix B discusses circuit, line, and node counters.
EVENTS	This display option includes information about events to be logged and is valid only for the SHOW LOGGING and LIST LOGGING commands.
STATUS	This display option includes dynamic information that usually reflects network operations for the running network. Depending on the entity identifier, this may include the local node and its operational state, reachable and unreachable nodes and their operational states, or circuits and their operational states. This display option is valid only with the SHOW command.
SUMMARY	This display option includes an abbreviated list that contains CHARACTERISTICS and STATUS information. For example, SHOW ACTIVE NODES SUMMARY includes the node name (characteristics) and whether a remote node is reachable or unreachable (status).

Examples of these display options and their formats are provided in Chapter 4 under the descriptions of the SHOW and LIST commands. If you do not specify a display type when issuing a SHOW or LIST command, SUMMARY information is displayed by default.

When you issue a command to display information about network entities, you can use either the singular or plural form of the entity identifier as shown in the following examples:

```
NCP>SHOW NODE BOSTON CHARACTERISTICS
                  .
                  .
                  .

NCP>SHOW KNOWN NODES CHARACTERISTICS
                  .
                  .
                  .
```

In addition, a second form of the plural – the ACTIVE keyword – can be used with CIRCUIT, LINE, LINK, LOGGING, and NODE entity identifiers. Whereas the KNOWN keyword displays information for all instances of the entity designated by the entity identifier available to the local node, the ACTIVE keyword displays information for all instances of active entity identifiers (that is, those whose state is other than OFF). The following example shows how to display the characteristics for all active nodes in the network.

```
NCP>SHOW ACTIVE NODES CHARACTERISTICS
                  .
                  .
                  .
```

All NCP display commands optionally allow you to direct the information displayed to a user-specified output file with the TO file-spec parameter. The following example creates the file NET.LOG in which summary information of all event logging for the network is stored.

```
NCP>SHOW KNOWN LOGGING SUMMARY TO NET.LOG
                  .
                  .
                  .
```

3.3 Monitoring a Remote Node

To monitor a remote node you can use the TELL prefix or the SET EXECUTOR NODE command. (Refer to Section 2.9 for more information.)

Once you have established a communication link with the remote node you want to monitor, use the SHOW/LIST commands as described in Section 3.2.

The following example displays the line characteristics, circuit counters, and line status at the remote node MILAN:

```
NCP>SET EXECUTOR NODE MILAN

MILAN::NCP>SHOW KNOWN LINES CHARACTERISTICS

MILAN::NCP>SHOW CIRCUIT circuit-id COUNTERS

MILAN::NCP>SHOW LINE line-id STATUS
```

The following example records all event logging information, such as sink settings and filters (not the events themselves), from remote node MILAN to the local node.

```
NCP>TELL MILAN [access control information] SHOW KNOWN LOGGING KNOWN EVENTS
```

3.4 Testing the Network

NCP provides tests for checking out network software and hardware. The tests send data through various network components and then return the data to its source. After you have started your DECnet/E software, you may want to run some of these tests to determine whether network software and hardware are functioning properly. (See the *DECnet/E Network Installation Guide* for more information.)

These tests should be used in the following situations:

1. The first time you build a new version of DECnet/E
2. After communications hardware changes, new installations, or repairs
3. For troubleshooting when network problems occur

The problems you encounter in network operation can be due either to hardware problems or to specifying the wrong parameters. The latter problem can be easily fixed using NCP.

Different variations of these tests are available to exercise the various layers of the network. These tests may be initiated by user-written programs or by DECnet/E-supplied programs. This section describes those variations that relate to DECnet/E loopback capabilities and the LOOP command.

There are five levels of loopback testing possible with DECnet/E. These five levels are illustrated in Figure 3–1 and are listed below:

1. Local logical link loopback
2. Controller loopback
3. Hardware loopback
4. Remote Transport loopback (Phase III only)
5. Remote logical link loopback (Phase II and Phase III)

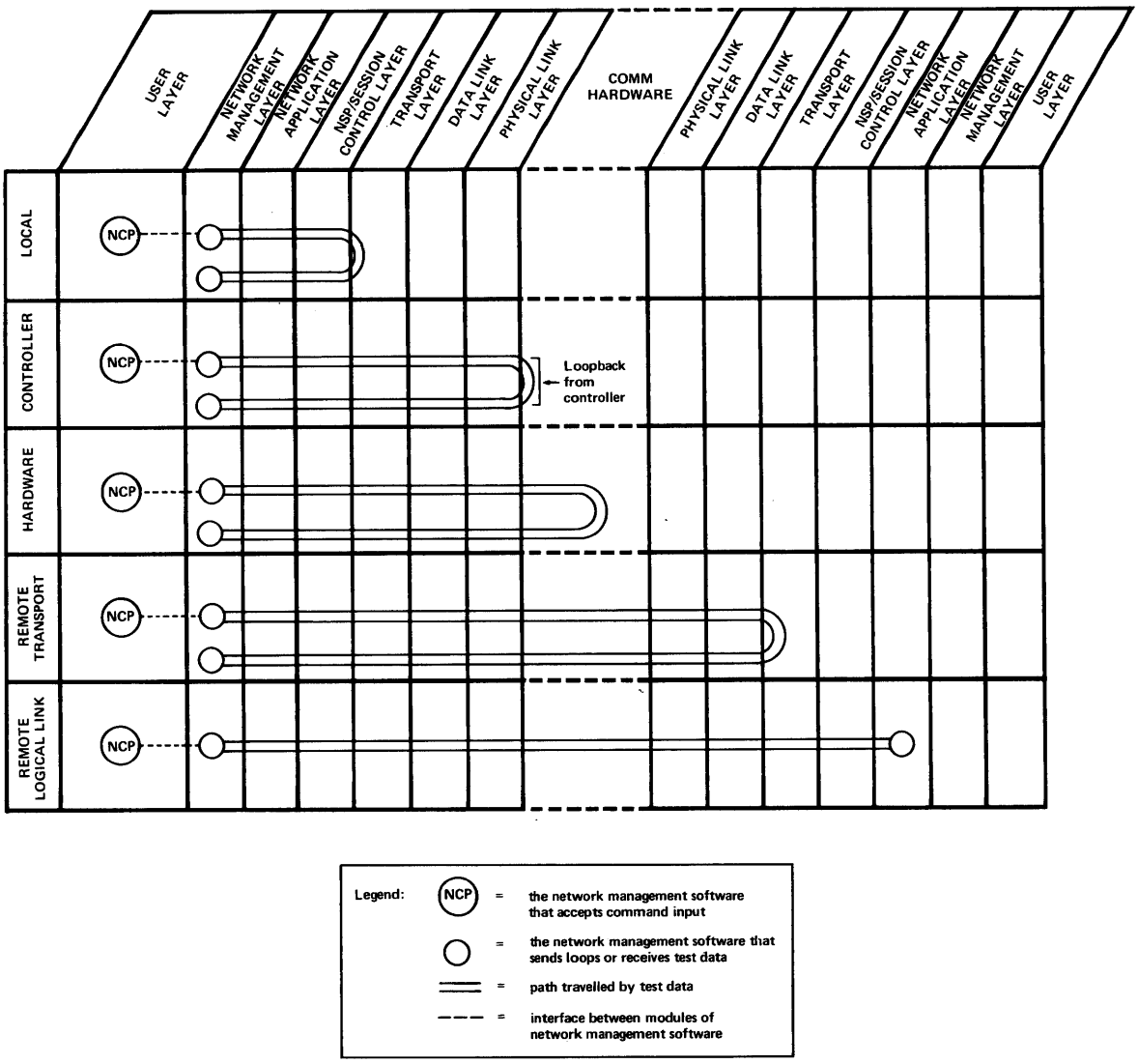


Figure 3–1: Levels of Loopback Testing

Each level tests for accurate data transmission one step "farther out" from the local node. If your node has too many data transmission errors, you can isolate the problem by proceeding through these tests. For example, if the local logical link loopback test shows no errors, but the controller loopback test does, the problem is probably in the communications device.

NCP's LOOP command performs a loopback test using a specified node or circuit. If, for example, you want to test a logical link on circuit DMC-0 that connects the local, executor node (NYC) with node SANFRN, you can make up a loop node name (for example, TSTDMC) and enter NCP's SET NODE command. The following command is an example of Remote Transport Loopback (see Section 3.4.4):

```
SET NODE TSTDMC CIRCUIT DMC-0
```

Run the test by entering NCP's LOOP command:

```
LOOP NODE TSTDMC
```

The test messages then travel on the circuit you specified. The SET NODE node-id CIRCUIT circuit-id command is necessary because, under normal operation, the routing module decides which circuit to use. The messages can return on a circuit other than the one you specified because the remote routing module chooses the circuit of least cost.

Transport treats a loop node as a destination equal to the executor node, but "tied" to a particular circuit. When NSP gives Transport a message with the destination equal to the executor node, Transport gives the message back, without passing it to a driver for transmission over any circuit.

When NSP gives Transport a message with the destination equal to a remote node, Transport chooses a circuit which has the minimum cost to the remote node and passes the message to the driver for the specified circuit for transmission. When NSP gives Transport a message with the destination equal to a loop node, Transport fills in the executor node number as the destination for that message and passes the message to the driver for the circuit identified with the specified loop node name. The local driver tries to transmit the message with the destination equal to the executor node. Somewhere the message is looped back, either by hardware or by the adjacent node's Transport software. Then the message is received by the local driver and passed to Transport. Transport delivers the message to the local NSP because its destination is the executor node, like messages which originate at remote nodes.

To remove the association of the loop node name with the circuit, enter NCP's CLEAR NODE command:

```
CLEAR NODE TSTDMC CIRCUIT
```

3.4.1 Local Logical Link Loopback

Local logical link loopback occurs when a DECnet/E program establishes a logical link with itself or with another program at the local node. Logical link loopback tests the user program and NSP, as shown in Figure 3–2. Physical communication lines are not used. Logical link loopback tests do not affect normal network operations and can be done at the same time as other network operations.

The MIRROR program provides the logical link loopback capability for Phase III nodes only. MIRROR does nothing more than return received messages to the sending node. MIRROR can be used locally in your node or remotely to test the path between your node and a remote node (see Section 3.4.5). MIRROR is installed as a permanent network object and is invoked with NCP's LOOP command.

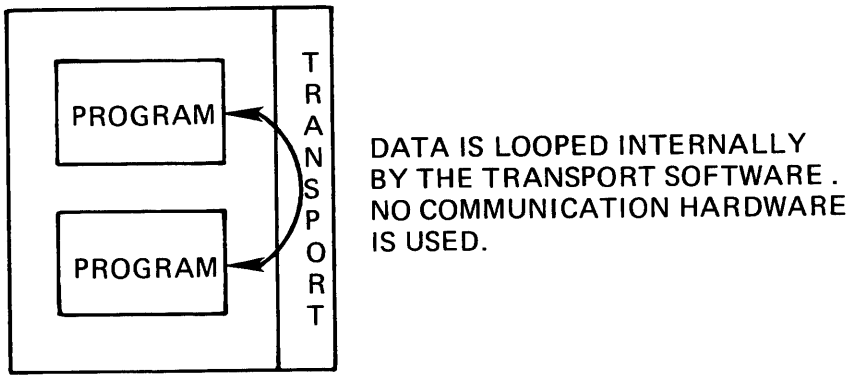


Figure 3–2: Local Logical Link Loopback

3.4.2 Controller Loopback

Controller loopback uses a maintenance feature of the communications device hardware (DMC11s, DMR11s, DMP11s, or DMV11s) to loop data within the device, without involving the cable, modem, or communications circuit (Figure 3–3). Loopback connectors are not required and cables or modems do not have to be disconnected. The controller loopback mode can be used to test some of the communications device hardware, in addition to NSP, Transport, and the local programs.

The controller loopback test operates with a single communications device. To successfully loop the data, both the transmitter and receiver must operate at the same time. Half-duplex operation implies that the receiver for the communications device is disabled when the transmitter is operating. Full-duplex operation implies that the transmitter and receiver operate simultaneously and independently. Therefore, these loopback arrangements require the communications device to operate in full-duplex mode.

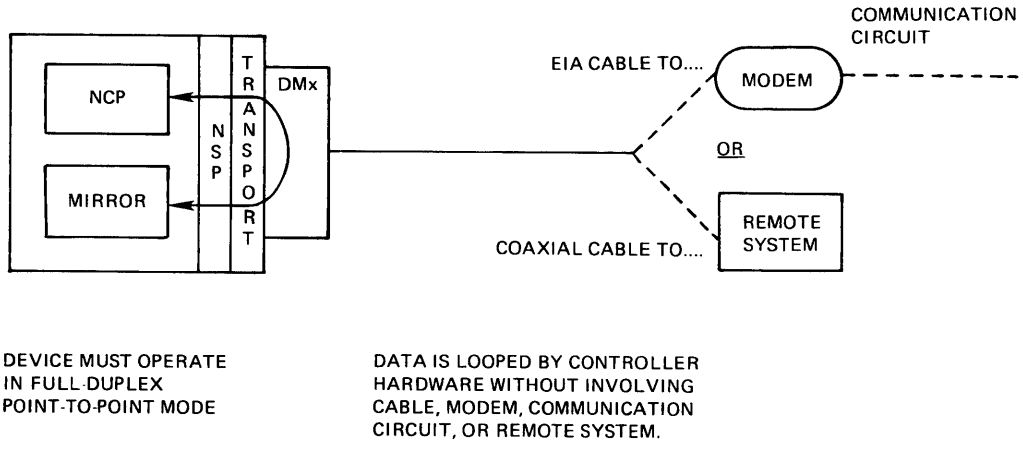


Figure 3–3: Controller Loopback

The SET LINE *line-id* CONTROLLER LOOPBACK command conditions the line for controller loopback (that is, it establishes the line as a loopback circuit). For each circuit to be set in the loopback state, you also need to define a loop node name. The loop node name is a pseudonym for the looped-back circuit. After the node initialization sequence completes, the loop node acts as the remote node on that circuit.

Controller loopback tests have no effect on network activity on other lines and can be done at the same time as normal network operations. In addition, controller loopback tests can be run on multiple lines at the same time.

Connections directed to a target program on the loop node (which is actually the local node) use the looped data path within the communications device. For example, you can use the LOOP NODE command (using the loop node name established in the SET NODE TSTDMC CIRCUIT DMC-0 command) to send a test pattern through the communications device and back to the local node. Or, you could use the NFT or TLK utilities. (Refer to the *DECnet/E Guide To User Utilities* for details on how to run these utilities.) The LOOP NODE command is more convenient because it needs only one operator and one terminal and does all necessary data checking automatically.

3.4.3 Hardware Loopback

Hardware loopback is used to test components of the physical communications path other than the communication devices, NSP, and the local programs. All of the hardware loopback tests operate by installing or enabling a loopback device somewhere in the communications path between the local communications device and the remote system. In the variations described below, and shown in Figure 3–4, the loopback point moves progressively farther away from the local system. The various tests can be used to isolate a problem to a component on the communications path.

To perform a hardware loopback test, you must set up the hardware for loopback testing on one or more lines (see Appendix C of the *DECnet/E Network Installation Guide*).

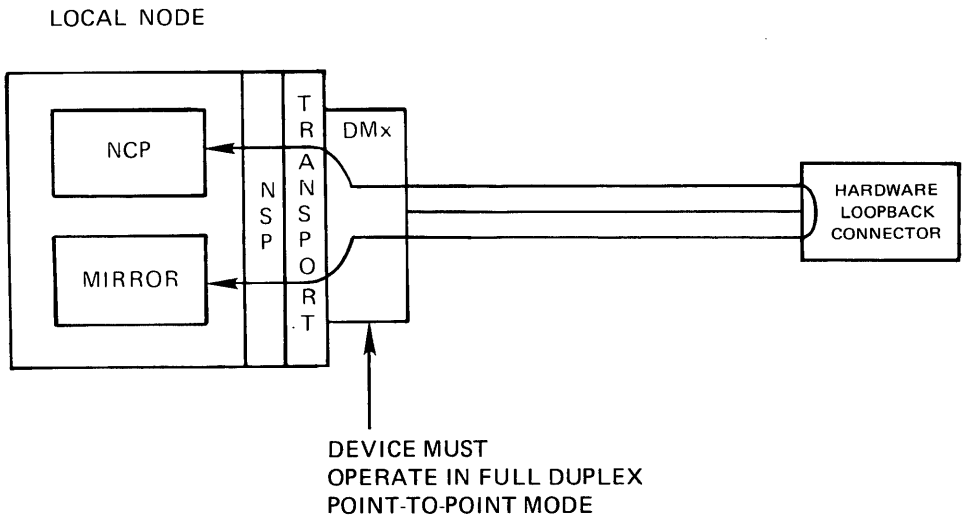


Figure 3–4: Hardware Loopback

3.4.4 Remote Transport Loopback (Phase III only)

There is a remote node loopback test for Phase III nodes that tests the node up to the Transport level. This low-level loopback test is useful because you can isolate network problems at each level of the software. For example, if there is a problem in the node, but you are able to loop through the Transport level, you know the problem lies somewhere at the NSP level. To perform this test, the remote node must be a Phase III routing node (not an end node). Figure 3–5 describes this type of loopback.

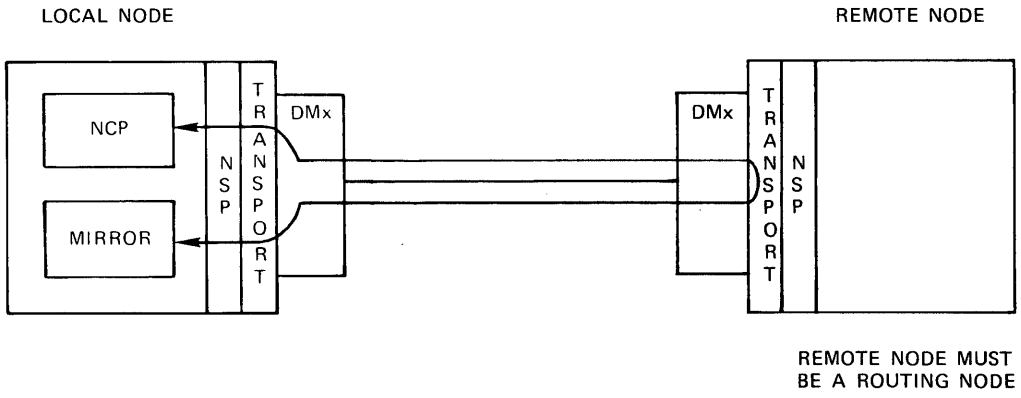


Figure 3–5: Remote Transport Loopback (Phase III only)

3.4.5 Remote Node Loopback (Phase II and Phase III)

The remote node loopback test for Phase II and III nodes tests the physical line between two nodes, line interfaces on both ends of the physical line, NSP, and Transport at both nodes. This test, shown in Figure 3–6, is used to

test point-to-point connections. This type of remote node loopback tests the node up to the NSP level.

The MIRROR program is used in Phase III nodes. The local NSP establishes a logical link with the MIRROR program at the remote node.

NCU (Network Control Utility) is used in DECnet/E V1.1 to perform remote node loopback for Phase II nodes. Other Phase II systems might call this loopback utility something different.

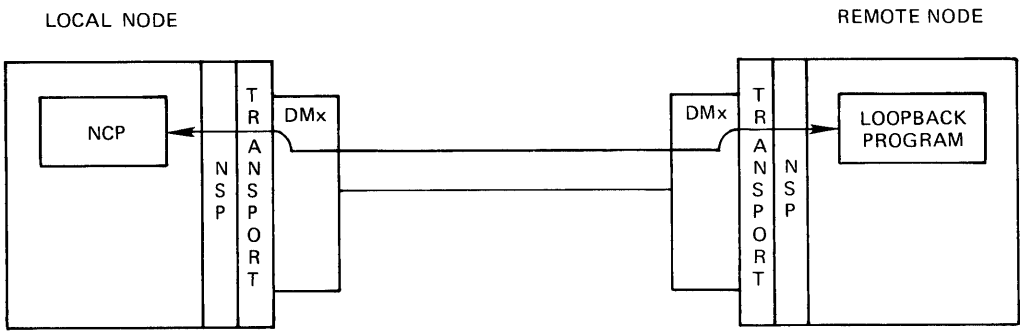


Figure 3–6 Remote Node Loopback (Phase II and Phase III Nodes)

3.5 Displaying Error Counters

The SHOW CIRCUIT *circuit-id* COUNTERS command displays data transmit and receive errors that occur at the DDCMP level. The SHOW CIRCUIT *circuit-id* COUNTERS command displays the contents of error counters kept by the device itself and by Transport.

Example:

```
NCP> SHOW CIRCUIT DMC-1 COUNTERS
Circuit Volatile Counters as of 18-Nov-81 11:12:01

Circuit = DMC-1
     12856 Seconds Since Last Zeroed
           0 Terminating Packets Received
           0 Originating Packets Sent
           0 Terminating Congestion Loss
           0 Transit Packets Received
           0 Transit Packets Sent
           0 Transit Congestion Loss
           0 Circuit Down
           0 Initialization Failures
           0 Bytes Received
           0 Bytes Sent
           0 Data Blocks Received
           0 Data Blocks Sent
           0 Data Errors Inbound
           0 Data Errors Outbound
           0 Remote Reply Timeouts
           0 Local Reply Timeouts
           0 Local Buffer Errors

NCP>
```

3.5.1 Local Buffer Errors

If a message is received and a buffer is not available for storing the message, the communications device will negatively acknowledge (NAK) the message, thereby forcing the remote system to retransmit. Each NAK shows up in the COUNTER display. If the same message is negatively acknowledged eight times, the communications device interrupts the PDP-11 and informs the driver a nonfatal overrun error has occurred.

Thus, a large number of "Buffer Unavailable" errors returned by a SHOW CIRCUIT command indicates the receive buffer quota for the line is too small. If a NAK is sent because the receive buffer was too small, you need to make sure the same buffer size is specified for both your node and the adjacent node. Increase the buffer quota by one or two buffers using the SET LINE *line-id* RECEIVE BUFFERS *number* command. You may also have to increase the size of the extended buffer pool (XBUF) to stay within the guidelines specified. See the *DECnet/E Network Installation Guide* for more information.

Buffer Unavailable errors happen more often if a fast processor (for example, a PDP-11/70) with a high-speed line is connected to a slower processor (for example, a PDP-11/34) with a high speed line (for example, a 1-Megabaud link). The Buffer Unavailable errors are reported by the slower processor.

3.5.2 Data Errors Inbound

DDCMP uses a 16-bit, CRC-16, Cyclic Redundancy Check polynomial for error detection. When the communications device detects an error in either the DDCMP header portion of a DDCMP message or the data portion of a message, it increments the appropriate counter and negatively acknowledges the message, causing the remote system to retransmit the message. If the same message is negatively acknowledged eight times for either of these reasons, the device interrupts the PDP-11 to inform the driver that a nonfatal data check error has occurred.

Errors of this type are common on switched or leased telephone lines. They should not occur often on coaxial cables. Large numbers in the counter display indicate some type of degradation in the communications facilities. If the circuit is a dial-up (switched) telephone line, the condition may clear up simply by hanging up and redialing the remote system. Consult the carrier if problems occur with a leased line. For cable connections, check for damage to cables, and check the routing of cables to avoid electrically "noisy" environments.

The communications device also runs a timer on each message transmitted and sends a REP (Reply to Message Number) message if no response (ACK or NAK) is received during the timeout interval. The REP message forces the transmitter and receiver to resynchronize the message sequence and start retransmissions if necessary. The PDP-11 is interrupted if eight con-

secutive REPs are sent with no response from the remote system. The driver logs an error and increments the counter.

3.5.3 Data Errors Outbound

If the remote system detects a CRC error or a Buffer Unavailable condition as described previously, it will negatively acknowledge the message and force the local system to retransmit. Every NAK received is counted by the device and appears in the counter display. No interrupt occurs, however, and the device continues to retransmit until the message is acknowledged, or until a transmit timeout in the driver expires (this timer is equal to the listen timer + 2 seconds). The listen timer is a multiple of the hello timer. The hello timer can be modified by the NCP command SET CIRCUIT circuit-id HELLO TIMER number.

If the counters indicate a large number of NAKs received, check the error counters at the remote site. If Buffer Unavailable errors are causing the NAKs, increase the buffering on the remote system. If the NAKs are caused by CRC errors, check the modems and communication circuits.

The RSTS/E device driver runs a timer on the last message transmitted to prevent any NAK/retransmission sequence from continuing forever. Assuming that a local RSTS/E system is trying to transmit a message which is repeatedly negatively acknowledged or not acknowledged at all by the remote communications device, the local driver times out on transmission, logs a timeout error, and takes the circuit down. Timeouts of this type mean that no message has gotten across the line within the time interval set from transmit timeout. This implies that the circuit has been disconnected, a modem has failed, or the remote system has crashed. (Note that it is actually possible for a device to continue operating and negatively acknowledge messages even though the host system is down.) If the circuit is at fault and both systems are DECnet/E systems, they both time out on transmission and take the line down. If the circuit is idle (that is, there are no normal messages to time) the DECnet/E Transport sends Hello messages to generate artificial traffic – only to detect failures of this type.

In the reverse direction, a REP received is a request from the remote system to resynchronize the message sequence. The local device responds with an ACK or NAK to indicate the sequence number of the last message received correctly. Each REP received is counted by the device and appears in the counter display, but no interrupt occurs.

A large number of REPs Received is usually caused by degradation in the communications circuit.

3.5.4 Counter Wraparound

Some of the error counters (displayed by SHOW CIRCUIT *circuit-id* COUNTERS command) for the DMC11 and DMR11 are 8-bit counters that

wrap around (reset to zero) after 255 errors. The error counters can wrap around quickly on a high-speed line that is prone to errors.

For example, on a 56-kilobaud line using 100-character data messages, the NSP and Transport overhead is 15 bytes and the DDCMP overhead is 10 bytes, for a total message length of 125 bytes or 1000 bits. The message transmission time is calculated as follows:

$$\frac{1000 \text{ bits}}{56000 \text{ bits/second}} = 18 \text{ milliseconds}$$

This translates to 56 messages per second. If messages were arriving at this rate and all messages were NAKs or CRC errors, the data CRC error counter kept by the DMC11 would overflow in about five seconds. With the device, retransmissions can actually occur at the rated line speed because no system, NSP, or user overhead is involved. The error counters often overflow under these conditions.

The ZERO CIRCUIT *circuit-id* COUNTERS command can be used to reset the counters to zero after they have been displayed.

3.6 Shutting Down Your Node with NETOFF

You can shut down the network in an orderly way by using the NETOFF utility. NETOFF allows you to shut down the network without shutting down the entire operating system; it causes a gradual shutdown rather than an immediate disconnect.

Although NETOFF performs a similar operation to the SET EXECUTOR STATE SHUT command (both allow existing links to complete), only NETOFF issues a message to other users telling them when the network is about to close down. This gives network users an opportunity to respond to the message that the network is shutting down – they can finish up their network processing or request that network operation be extended.

To invoke NETOFF, type:

```
RUN $NETOFF
```

The following sequence of operations then occurs:

1. NETOFF prompts you for information to be supplied: how many minutes until network shutdown and how many minutes until new network links should be disabled.
2. NETOFF broadcasts the following message to all system terminals: THE NETWORK IS SHUTTING DOWN IN [*n*] MINUTES. (If you enter a CTRL/C and then enter the command SET EXECUTOR STATE ON to NSP before the time period expires, the shutdown operation is terminated and network operations are resumed.)

3. When the time period for disabling new network links expires, NETOFF sends the SET EXECUTOR STATE SHUT command to NSP. If no logical links are open, NSP stops network operation completely, and the NETOFF program is complete. Otherwise, NSP prohibits the forming of any new logical links but allows normal operation to continue on existing links.

4. When the time period for network shutdown expires, NETOFF sends the SET EXECUTOR STATE OFF command to NSP. At this point, any logical links that exist are broken by NSP. "Network Abort" status is sent to local programs with links open. NSP then shuts down Transport, thus ending network operation.

5. NETOFF sends a STOP command to the Event Logger program, performs a CLEAR SYSTEM and returns a SHUTDOWN COMPLETE message to the operator.

3.7 Analyzing Crash Dumps with ANALYS

When a RSTS/E system crash occurs, time-sharing operations are halted. If crash dumps were enabled by the RSTS/E system manager at system startup, the critical contents of memory are written into the file [0,1]CRASH.SYS.

The ANALYS system program analyzes information stored in the CRASH.SYS file and produces a report of the contents in memory, including XBUF (high memory), at the time of the crash.

The output of ANALYS supplies hardware and software information that can be used by a software specialist to ascertain the possible causes of a system crash. The ANALYS output includes a report similar to SYSTAT in addition to a memory dump of the critical contents of memory. The report also provides information on network activity and the network event logging printouts. The ANALYS output is supplemented by the error logging printouts. Refer to the *RSTS/E System Manager's Guide* for details on how to run ANALYS.

ANALYS should be run after you restart the RSTS/E system. This is necessary because a subsequent system crash would cause the CRASH.SYS file to be overwritten with a new crash dump, thereby destroying the information recorded from the previous crash.

For some critical RSTS/E SPRs (Software Performance Reports), it might be necessary to supply a copy of the CRASH.SYS file in machine-readable form, together with other system information. For DECnet/E, such cases should also include a machine-readable copy of the volatile parameter file, [0,1]NSP0.SYS.

Chapter 4
DECnet/E Network Control Program

The Network Control Program (NCP) is a DECnet/E utility program that accepts terminal commands to configure, control, monitor, and test a DECnet network. This chapter shows how to use NCP. Appendix A provides a command reference summary.

4.1 Invoking NCP

To invoke NCP, type the following command:

```
RUN $NCP
```

NCP will return the following prompt:

```
NCP>
```

Once you receive this prompt, you can type NCP commands.

4.2 Exiting NCP

To exit NCP, type EXIT or CTRL/Z after the NCP prompt (NCP>).

4.3 Issuing NCP Commands

You enter NCP commands as entity identifiers and keyword parameters separated by spaces or tabs. Use the standard continuation line convention — a hyphen as the last character in the line — to continue a long command to the next line. The NCP prompt (NCP>) is not printed on continuation lines. Instead, continuation lines prompt with MORE>. For example:

```
NCP>SET EXECUTOR -
MORE>ADDRESS 11 -
MORE>INCOMING TIMER 30 -
MORE>STATE ON
NCP>
```

If the first character of a line is an exclamation point (!), the line is ignored by NCP as a comment line. Hyphens within and at the end of a comment line are ignored. Lines beginning with an exclamation point, however, are not ignored if they follow a command line ending with a hyphen.

When entering an NCP command, you can abbreviate any command, command qualifier, or optional keyword parameter name to its first three letters. The following examples perform the same function:

```
NCP>SET EXE ADD 11 STA ON

NCP>SET EXECUTOR ADDRESS 11 STATE ON
```

For clarity, all examples in this section show the full command syntax.

Some NCP commands permit certain keywords to be omitted when the command is entered. These are "helping" words that provide syntactic clarity. The same command can be specified either way as shown in the following examples:

```
NCP>ZERO CIRCUIT DMV-0.1 COUNTERS

NCP>ZERO CIRCUIT DMV-0.1
```

4.4 Command Syntax

An NCP command has three parts: a command name, an entity identifier, and one or more keyword parameters. For example:

Command name	Entity Identifier	Keyword parameters
SHOW LIST	ACTIVE LINES KNOWN LINES LINE *line-id*	CHARACTERISTICS TO *file-spec* COUNTERS STATUS SUMMARY

You must supply an entity identifier for each command, and, in most cases, you can optionally select one or more parameters from the keyword parameter list. Italic lettering is used to indicate user-supplied variables. Keywords can be abbreviated to their first three characters.

NCP has two sets of commands: one for accessing the volatile parameter file and one for accessing the permanent parameter file. The mapping of the commands to the various operations to be performed is shown in the following table:

Function	Volatile	Permanent
Creating/Modifying parameters	SET	DEFINE
Displaying parameters	SHOW	LIST
Zeroing/Resetting parameters	CLEAR	PURGE

For convenience, commands having the same entity identifiers and keyword parameters are listed together. Command categories are referenced in the following order: SET/DEFINE, SHOW/LIST, and CLEAR/PURGE. With the exception of the first two commands listed in the SET/DEFINE category, commands are referenced alphabetically within each command category.

There are four special NCP commands: ABORT LINK, EXIT, LOOP, and ZERO. These commands are described in alphabetical order in Section 4.8.

The TELL prefix, used to specify a remote node as the executor node for the one command it prefixes, is described in Section 4.9.

In some cases, the entity identifiers are not meaningful to both the permanent and volatile parameter files. For example, the entity identifier ACTIVE LINES is meaningful only to the SHOW command because information of this type is dynamic in nature. Similarly, displaying COUNTERS is meaningful only to the SHOW command.

Many parameters and a few entity identifiers require user-supplied values. A number of user-supplied values that appear in several NCP commands are defined in the table that follows. Whether they are specified with entity identifiers or keyword parameters, the values have the meanings shown here unless otherwise noted.

Variable	**Meaning**
access control information	Information that identifies you to a program on a remote node (Section 2.9.3). The format for access control information is as follows:
[USER *user-id*] [PASSWORD *password*] [ACCOUNT *account*]	
circuit-id	A string in the form *dev-c*[-*u*].*t* (Section 2.6).
file-spec	A file specification is the syntax used to name a file. For example, SY:[200,1] REB.RNO.
line-id	A string in the form *dev-c*[-*u*] (Section 2.5).
node address	A numeric value in the range of 1-255 (Section 2.2).
node-id	Either a node name or a node address (Section 2.2).
node name	A string of up to six alphanumeric characters containing at least one alphabetic character (Section 2.2).

4.5 SET/DEFINE Commands

Use the SET commands to change the volatile parameter file during network operation. Use the DEFINE commands to establish the permanent parameter file at network installation and to change the permanent parameter file during network operation.

Use the DEFINE PERMANENT PARAMETER FILE command to establish the default permanent parameter file ($NETPRM.SYS) on the executor node.

Command Format:

DEFINE PERMANENT PARAMETER FILE

Example:

The following command creates the default permanent parameter file ($NETPRM.SYS) on the executor node. Once this command is entered, you can establish all other entities and values for keyword parameters in the permanent parameter file on the executor node. Note that if a permanent parameter file currently exists, it will not be deleted by this command.

```
NCP>DEFINE PERMANENT PARAMETER FILE
```

SET SYSTEM

Use the SET SYSTEM command after the RSTS/E system is started or after the CLEAR SYSTEM command is entered.

Command Format:

SET SYSTEM

Example:

The following command installs the volatile parameter file and loads it from the permanent parameter file. If the logging state is defined as ON, SET SYSTEM starts the event logging program $EVTLOG.

```
NCP>SET SYSTEM
```

Use the SET CIRCUIT command to create or modify circuit parameters in the volatile parameter file on the executor node. Use the DEFINE CIRCUIT command to create or modify circuit parameters in the permanent parameter file on the executor node.

Command Format:

```
SET       CIRCUIT circuit-id    ALL
DEFINE    KNOWN CIRCUITS        AUTORESTART
                                        OFF
                                        ON
                                COST cost
                                HELLO TIMER seconds
                                OWNER EXECUTOR
                                STATE
                                        OFF
                                        ON
                                TRIBUTARY trib-address
```

NOTE

Wildcards are not permitted in the *circuit-id* when using the ALL keyword parameter. The keyword parameter ALL cannot be specified with the DEFINE command or the KNOWN CIRCUITS entity identifier. The keyword parameter STATE is used only with the SET command.

Entity Identifiers:

CIRCUIT *circuit-id*

When specified with the DEFINE command, identifies a circuit in the permanent parameter file ($NETPRM.SYS).

When specified with the SET command, identifies a circuit in the volatile parameter file [0,1](NSP0.SYS).

KNOWN CIRCUITS

When specified with the DEFINE command, refers to all circuits that have a name in the permanent parameter file.

When specified with the SET command, refers to all circuits that have a name in the volatile parameter file.

Keyword Parameters:

ALL

Copies all parameters for the circuit from the permanent parameter file into the volatile parameter file [0,1](NSP0.SYS).

AUTORESTART

Refers to when you are unable to send or receive data on a circuit.

OFF — Transport declares the circuit as permanently down.

ON — Transport restarts the circuit after it goes down.

COST *cost*

Specifies the routing cost of the circuit. The value range is 1-25. Transport uses this value to route messages along the path of least cost from the source node to the destination node.

HELLO TIMER *seconds*

Specifies how often (at what rate)

Transport sends a message to the adjacent node on the circuit. This message is sent if there has been no data activity on the circuit for that period of time. The recommended time period is 10-15 seconds for high speed circuits (56 k baud or more) and 30-60 seconds for low speed circuits (9600 baud or less). This value should be the same on both sides of the circuit.

The LISTEN TIMER is a multiple of the HELLO TIMER. The LISTEN TIMER determines the period of time allowed to elapse before Transport receives a message from the adjacent node on the circuit. If the LISTEN TIMER expires, the circuit goes down.

OWNER EXECUTOR

Specifies that the circuit is reserved for DECnet use exclusively.

STATE

Specifies the operational state of the circuit. There are two possible states: OFF and ON.

OFF — The circuit is not operating and cannot send or receive data.

ON — The circuit is operating and can send and receive data.

TRIBUTARY *trib-address*

Specifies the tributary address of the circuit. The value range is 1-255.

Example:

The following command turns circuit DMC-1 to the ON state.

```
NCP>SET CIRCUIT DMC-1 STATE ON
```

SET EXECUTOR
DEFINE NODE

Use the SET EXECUTOR command to create or modify parameters in the volatile parameter file that controls the network on the executor node. Use the DEFINE EXECUTOR command to create or modify parameters in the permanent parameter file that controls the network on the executor node.

When the executor has been defined to have a particular name or address, that name or address can be used as a *node-id* to refer to the executor. Thus, SET EXECUTOR is equivalent to SET NODE *node-id* when *node-id* is the name or address of the executor.

The commands SET/DEFINE EXECUTOR modify the volatile and permanent parameter files for the executor node. The executor node can be either the local node or a remote node.

You can specify SET/DEFINE EXECUTOR *node-id* in three ways. Note that the keyword parameter STATE is used only with the SET command.

1. By specifying the ALL keyword parameter only.
2. By specifying the STATE substate keyword parameter only.
3. By specifying any combination of keyword parameters other than ALL or STATE.

To change the executor address (or any node address) you must issue the following commands:

1. Issue the command CLEAR/PURGE EXECUTOR ALL to make sure the new address is undefined.
2. Issue the command SET/DEFINE EXECUTOR ADDRESS to specify the new node address. Note that the old address is undefined once you issue this command. Therefore, you must redefine the old address if you want it to exist.

The executor name can be changed at any time as long as the new name does not duplicate an existing node name.

Command Format:

SET DEFINE	EXECUTOR NODE node-id	ADDRESS *node address* ALIAS *name* ALL BUFFER SIZE *number* COUNTER TIMER *seconds* DATA TRANSMIT QUEUE MAXIMUM *number* DEFAULT ACCOUNT *user-id* DELAY FACTOR *number* DELAY WEIGHT *number* IDENTIFICATION STRING INACTIVITY TIMER *seconds* INCOMING TIMER *seconds* INTERRUPT TRANSMIT QUEUE MAXIMUM *number* MAXIMUM ADDRESS *number* MAXIMUM BUFFERS *number* MAXIMUM CIRCUITS *number* MAXIMUM COST *number* MAXIMUM HOPS *number* MAXIMUM LINKS *number* MAXIMUM VISITS *number* NAME *node name* OUTGOING TIMER *seconds* RETRANSMIT FACTOR *number* ROUTING TIMER *seconds* STATE OFF ON RESTRICTED SHUT VOLATILE PARAMETER FILE NAME *file-spec*

NOTE

The keyword parameter ADDRESS must be defined before any other executor parameters can be entered. The keyword parameter ALL is used only with the SET command. The keyword parameter VOLATILE PARAMETER FILE NAME *file-spec* is used only with the DEFINE command. The keyword parameter MAXIMUM BUFFERS *number* can be set only when the EXECUTOR STATE is OFF.

Entity Identifiers:

EXECUTOR	Specifies the executor node.
NODE *node-id*	Specifies the node that represents the executor.

Keyword Parameters:

ADDRESS *number*	Specifies the address for the executor node. The valid range is 1-255.
ALIAS *name*	Specifies an alternate name to designate a node.
ALL	Copies all parameters for the executor node from the permanent parameter file ($NETPRM.SYS) to the volatile parameter file.
BUFFER SIZE *number*	Specifies the size of the line buffers and thereby controls the maximum segment size of all messages received. The buffer size should be the same for all lines. The value range is 246-568.
COUNTER TIMER *seconds*	Sets a timer whose expiration causes a node counter logging event. A value of 0 disables the timer. The value range is 1-65535.
DATA TRANSMIT QUEUE MAXIMUM *number*	Specifies the number of unacknowledged Network Data messages, per logical link, that NSP holds in a transmit queue after the messages are sent to a remote node. When a remote node acknowledges receipt of a data message, NSP removes it from the transmit queue. The number can be set from one to eight unacknowledged messages. See the "Transmit Queue Management" section in any of the DECnet/E programming manuals for more information.
DEFAULT ACCOUNT *user-id*	Specifies an account you assign to network programs such as MIRROR, FAL, and NML. It is recommended that this account is nonprivileged.

DELAY FACTOR *number*

Specifies the value of a timeout on an acknowledgment of a message sent to a remote node. The delay factor is specified in units of 16 (where 16 = 1 second). The delay factor is derived with the following algorithm:

$$\text{timeout} = \frac{\text{delay factor} * \text{delay estimate}}{16}$$

The recommended value is 32-48 (2-3 seconds times the node delay). The valid range is 1-255.

DELAY WEIGHT *number*

Specifies a "weighted average" that affects the way in which the delay estimate for a node is computed. The delay weight is derived with the following algorithm:

$$\text{delay estimate} = \frac{\text{message delay} + \text{weight} * \text{delay estimate}}{\text{weight} + 1}$$

The recommended value is 2. The valid range is 1-255.

IDENTIFICATION *string*

Specifies a text string that describes the executor node, for example, "Research Lab". The string is 32 characters of ASCII text. If the string contains blanks or tabs, it must be enclosed in quotation marks when entered in NCP. A quotation mark within a quoted string is indicated by two adjacent quotation marks ("").

INACTIVITY TIMER *seconds*

Specifies the period of inactivity (no data flowing in either direction) on a logical link before the node checks to see if the logical link still works. If no activity occurs within the specified number of seconds, NSP generates artificial traffic to test the link. The value range is 1-65535.

INCOMING TIMER *seconds*

Specifies a timeout value (in seconds) for the duration between the time a connect request is received for an object at the executor node and the time that connect request is accepted or rejected. If you do not accept or reject the connect request within the number of seconds specified, the request is rejected. The value range is 1-65535. The recommended value is 30.

INTERRUPT TRANSMIT QUEUE MAXIMUM *number*	Specifies the number of unacknowledged Interrupt messages and Link Service messages held in a transmit queue by NSP. Like the Data transmit queues, there is a separate Interrupt/Link Service transmit queue for each logical link. The number can be set from one to eight unacknowledged messages.
MAXIMUM ADDRESS *number*	Specifies the largest node address and consequently the greatest number of nodes the executor node can recognize. The value range is 1-255.
MAXIMUM BUFFERS *number*	Specifies the maximum number of transmit buffers the Transport buffer pool can use. If this number is set too low, the transit congestion loss circuit counter goes up. The recommended value is 5-10 times the number of specified circuits. If you have considerable traffic or lines of differing speeds, you might need to set maximum buffers to a larger number (as high as 30 times the number of specified circuits). The valid range is 3-65535.
MAXIMUM CIRCUITS *number*	Specifies the maximum number of circuits the executor node can use. If this value is 1, the executor is configured as an end node rather than a routing node. The valid range is 1-16.
MAXIMUM COST *number*	Specifies the maximum path cost allowed from the executor node to any node. The path cost is the sum of the line costs along a path between two nodes. Use a number in the range of 1 to 1022. A node is considered UNREACHABLE by the Transport module if the path cost is greater than this value. The value specified for maximum path cost must be greater than the value specified for maximum hops.
MAXIMUM HOPS *number*	Specifies the maximum number of routing hops from the executor node to any other reachable node. Use a number in the range of 1 to 30. A node is considered UNREACHABLE by the Transport module if the number of hops in the path is

greater than this value. The valid range is 1-255. The value specified must be less than that specified for maximum path cost.

MAXIMUM LINKS *number*

Specifies the maximum logical link count for the executor node. Use a number in the range of 1-127.

MAXIMUM VISITS *number*

Specifies the maximum number of nodes a message can visit prior to being received by the destination node. The DECnet routing software increments a *visit count* in each packet each time the packet is forwarded by a node. When the visit count exceeds this value, the packet is discarded. This prevents erroneous packets from traveling around the network forever. Use a number in the range 1-255. This number must be greater than the number specified for MAXIMUM HOPS. The recommended value is 1 or 2 greater than MAXIMUM HOPS.

NAME *node name*

Specifies the name designated to the executor node.

OUTGOING TIMER *seconds*

Specifies a timeout value (in seconds) for the duration between the time a connection is requested and the time that connection is acknowledged by the destination node. The value range is 1-65535. The recommended value is 30.

RETRANSMIT FACTOR *number*

Specifies the maximum number of times any given message (except a Connect Initiate Message) will be transmitted if no acknowledgment is received from the remote node before the logical link is disconnected. The value range is 1-255. The recommended value is 8.

ROUTING TIMER *seconds*

Sets the timer (in seconds) that controls the *periodic* sending of the current routing information. That is, routing messages are sent at this rate if *nothing* changes. The value range is 1-65535. The recommended value is 300.

STATE

Specifies the operational state of the local node. Four possible states are:

OFF	Allows no new logical links, stops existing links, and stops route through traffic.
ON	Allows logical links and forwarding through the node.
RESTRICTED	Allows no new inbound links from other nodes, but allows outbound links to other nodes, and does not affect routed traffic in progress. Unlike SHUT, this is a permanent state.
SHUT	Allows no new logical links, but does not destroy existing links. Forwarding through the node is allowed until the node goes to the OFF state. SHUT is a temporary state because it goes to the OFF state when all existing links are gone.

VOLATILE PARAMETER FILE NAME *file-spec*

Specifies the volatile parameter file (default = SY0:[0,1]NSP0.SYS) in the executor node. If you have a private disk, you can improve performance for most volatile NCP operations by assigning the VOLATILE PARAMETER FILE NAME *file-spec* to the disk.

NOTE

The following commands are valid only while the executor is OFF.

Examples:

The following command loads all permanent parameter file entries stored for the local node into the volatile parameter file.

```
NCP>SET EXECUTOR ALL
```

The following command sets the local node's address to 11 and defines the identity of the local node. When the state is set to ON, the node address set here is used to identify the node.

```
NCP>SET EXECUTOR ADDRESS 11
```

The following command sets the local node's name to NYC and is used with the SET EXECUTOR ADDRESS command to establish the node's identity.

```
NCP>SET EXECUTOR NAME NYC
```

The following command turns the local node ON.

```
NCP>SET EXECUTOR STATE ON
```

SET EXECUTOR NODE

Use the SET EXECUTOR NODE command to establish a remote node as the executor.

The SET EXECUTOR NODE node-id command means that all commands during this invocation of NCP are sent to the Network Management Listener (NML) at the specified remote node and executed there. You can change a node's executor status by:

1. Exiting NCP
2. Specifying CLEAR EXECUTOR NODE
3. Specifying SET EXECUTOR NODE *node-id* to a different node-id

This operation is meaningful only to the SET command, and with the exception of access control information, no parameters other than those indicated above can be specified.

Command Format:

SET EXECUTOR NODE *node-id* [*access control information*]

NOTE

The NODE keyword must precede the node-id. (The access control information can appear in any order after the node-id. Access control information is optional (see Section 2.9.3).

Entity Identifiers:

EXECUTOR	Specifies the executor node.
NODE *node-id*	Specifies a node address or name to identify a node.

Examples:

The following command sets the executor to node WASHDC

```
NCP>SET EXECUTOR NODE WASHDC USER [1,217] PASSWORD BERYT
```

The following command displays counters for circuit DMC-1 on node WASHDC.

```
NCP>SHOW CIRCUIT DMC-1 COUNTERS
```

The following command resets the executor to the local node.

```
NCP>CLEAR EXECUTOR NODE
```

The following command displays counters for circuit DMC-1 on remote node WASHDC. Note that the effect is the same as stated in the previous commands.

```
NCP>TELL WASHDC SHOW CIRCUIT DMC-1 COUNTERS
```

Use the SET EXECUTOR NODE *node-id* command when you want to invoke a series of commands on a specified remote node. Use the TELL prefix when you have a single command to invoke on a specified remote node.

SET LINE
DEFINE

Use the SET LINE command to create or modify line parameters in the volatile parameter file on the executor node. Use the DEFINE LINE command to create or modify line parameters in the permanent parameter file on the executor node.

Command Format:

```
SET         KNOWN LINES   ALL
DEFINE      LINE line-id  CONTROLLER
                                  LOOPBACK
                                  NORMAL
                          DEAD TIMER milliseconds
                          DELAY TIMER milliseconds
                          DUPLEX
                                  FULL
                                  HALF
                          PROTOCOL
                                  DDCMP CONTROL
                                  DDCMP DMC
                                  DDCMP POINT
                                  DDCMP TRIBUTARY
                          RECEIVE BUFFERS number
                          RETRANSMIT TIMER milliseconds
                          VERIFICATION
                                  ANSWER
                                  OFF
                                  ORIGINATE
```

NOTE

Wildcards are not permitted in the line-id when using the ALL keyword parameter. ALL cannot be specified with the DEFINE command or the KNOWN LINES entity identifier.

Entity Identifiers:

KNOWN LINES	When specified with the SET command, identifies all lines that are named in the volatile parameter file [0,1](NSP0.SYS).

When specified with the DEFINE command, identifies all lines that are named in the permanent parameter file ($NETPRM.SYS).

LINE *line-id*

Specifies the line identifier.

Keyword Parameters:

ALL

Copies all parameters for the line from the permanent parameter file into the volatile parameter file [0,1](NSP0.SYS).

CONTROLLER

Specifies the controller mode for the line. There are two possible modes:

LOOPBACK — The internal device controller loopback operating mode, used only for testing.

NORMAL — The normal device operating mode.

DEAD TIMER *milliseconds*

Controls the interval at which dead tributaries are polled. This option applies only to lines of type DDCMP CONTROL. The value range is 1-65535. If no value is specified, the default is 10000 (10 seconds).

DELAY TIMER *milliseconds*

Controls the minimum time to delay between tributary polls. This option applies only to lines of type DDCMP CONTROL. The value range is 1-65535. If no value is specified, there is no delay.

DUPLEX

Specifies the hardware mode of the line. There are two possible modes:

HALF — half-duplex

FULL — full-duplex

PROTOCOL

Specifies the type of DDCMP protocol that is used on the line. The protocol name values are as follows:

CONTROL — Specifies a multipoint control station (master). This protocol type applies only to the DMP/DMV and is used on lines connected to one or more tributary stations.

DMC — Specifies a variation of the POINT type protocol. DMC type is used to communicate with the DMC11 or other devices (for example, DMR11 operating in DMC mode) that use DDCMP V3.1.

POINT — Specifies a point-to-point line to a device other than a DMC. For example, a DMR, DMP, or DMV operating in DDCMP V4.0 mode. This protocol type is used on lines where the other end is also running in point-to-point mode. Note that RSTS/E supports only DMRs in DMC mode. However, you could be connected to a non-RSTS system with a DMR in V4.0 mode.

TRIBUTARY — Specifies a multipoint tributary (slave). This protocol type applies only to the DMP/DMV and is used on lines connected to a control station.

RECEIVE BUFFERS *number* — Specifies the maximum number of receive data buffers that the line is allowed to use from system resources. Although the value range is 3-255, values above 10 are not normally useful.

RETRANSMIT TIMER *milliseconds* — Applicable with the DMP/DMV devices. Specifies the amount of time before an unacknowledged block of data on the line is retransmitted. The value range is 1-65535 milliseconds. The default is 3000 (3 seconds). For the DMC, this parameter is fixed at 3 seconds and cannot be changed.

VERIFICATION — Indicates whether password verification is required to start up a remote node across this line (see the *DECnet/E Network Installation Guide* for node startup). If required, the remote node must submit a

password, which is verified by Transport. This parameter controls only Transport line initialization and does not affect traffic on the line. In particular, it does NOT relate to the accounting verification of Connect Initiate Messages.

ANSWER	Indicates that verification is required to initialize the line and that this side is the *answering* side (the other side is the ORIGINATE). The passwords used are the ANSWER passwords defined as node parameters.
OFF	Indicates that no verification is required to initialize the line.
ORIGINATE	Indicates that verification is required to initialize the line and that this side is the *originating* side. The passwords used are the ORIGINATE passwords defined as node parameters.

Examples:

The following command sets the line associated with DMC unit 0 to operate in a full duplex mode.

```
NCP>SET LINE DMC-0 DUPLEX FULL
```

The following command sets DMP-1 to multipoint master mode. All nodes connected to DMP-1 must have their lines set to PROTOCOL DDCMP TRIBUTARY so they act as multipoint slaves.

```
NCP>SET LINE DMP-1 PROTOCOL DDCMP CONTROL
```

The following command sets DMP-0 to point-to-point mode rather than multipoint mode, using DDCMP V4.0. Any node connected to DMP-0 must use a line operating in that mode and use that version of DDCMP. Note that a DMC cannot connect to this line because it uses V3.1 of DDCMP.

```
NCP>SET LINE DMP-0 PROTOCOL DDCMP POINT
```

The following command sets DMV-1 to DMC type protocol mode. Because DMC devices use an earlier version of DDCMP (DDCMP V3.1), any device connected to a DMC must be set to this protocol mode. This parameter is unnecessary for DMC devices because they operate only in this mode.

```
NCP>SET LINE DMV-1 PROTOCOL DDCMP DMC
```

Use the SET ALL LOGGING command if you need to stop or restart the Event Logger program while the network is running. Use the DEFINE ALL LOGGING command to determine whether the Event Logger program ($EVTLOG) is enabled when the SET SYSTEM command is entered.

Command Format:

```
SET    ALL  LOGGING  STATE  ON
DEFINE                     OFF
```

NOTE

It is recommended that the Event Logger program is ON while the network is running. Otherwise, network events will not be recorded, and there will be no record of lost events if they occur.

Keyword Parameters:

STATE Specifies the operating state of the Event Logger program.

- OFF The Event Logger program is not operating. Network events are not recorded while the program is in this state.
- ON The Event Logger program is operating. Network events are recorded while the program is in this state.

Examples:

The following command disables the Event Logger program and causes event logging to stop.

```
NCP>SET ALL LOGGING STATE OFF
```

SET ALL LOGGING STATE ON restarts $EVTLOG and resumes logging using the parameters most recently established through the SET LOGGING commands. This command is also used to restart the Event Logger program ($EVTLOG) if the program stopped for some reason (for example, because of a KILL command in the RSTS/E program UTILTY or because of a program error).

The following command defines the Event Logger program as being in the ON state in the permanent parameter file ($NETPRM.SYS). Therefore, $EVTLOG is enabled when the SET SYSTEM command is entered.

```
NCP>DEFINE ALL LOGGING STATE ON
```

DEFINE ALL LOGGING STATE OFF specifies that $EVTLOG is not enabled when the SET SYSTEM command is entered.

SET LOGGING
DEFINE

Use the SET LOGGING command to create or modify specified logging parameters in the volatile parameter file on the executor node. Use the DEFINE LOGGING command to create or modify specified logging parameters in the permanent parameter file on the executor node.

Command Format:

```
SET KNOWN LOGGING                CIRCUIT circuit-id
DEFINE LOGGING                   EVENTS event-list      SINK EXECUTOR
                    CONSOLE      KNOWN EVENTS               NODE node-id
                    FILE         LINE line-id
                    MONITOR      NAME sink-name
                                 NODE node-id
                                 STATE  ON
                                        OFF
```

NOTE

The keyword parameters NAME and STATE affect the executor node only. If the SINK parameter is omitted, it defaults to SINK EXECUTOR. The source qualifier parameters CIRCUIT *circuit-id*, LINE *line-id*, and NODE *node-id* are valid only when specified with an events parameter.

Entity Identifiers:

KNOWN LOGGING	When specified with the SET command, identifies all logging sink types in the volatile parameter file known to the executor node. When specified with the DEFINE command, identifies all logging sink types in the permanent parameter file known to the executor node.
LOGGING CONSOLE	Indicates the specified parameters for the logging console are to be created or modified.
LOGGING FILE	Indicates the specified parameters for the logging file are to be created or modified.
LOGGING MONITOR	Indicates the specified parameters for the logging monitor are to be created or modified.

Keyword Parameters:

CIRCUIT *circuit-id*
: Specifies the circuit identifier where one or more events originated.

EVENTS *event-list*
: Indicates which events are to be logged on the specified logging sink.

KNOWN EVENTS
: Indicates that all event classes and types are to be logged on the specified logging sink.

LINE *line-id*
: Specifies the line identifier for the line where one or more events originated.

NAME *sink name*
: Identifies the sink for the executor node as follows:

  CONSOLE
  : Identifies the terminal device name that receives the events, for example, KB12:. This parameter can be changed while the logging state is ON.

  FILE
  : Identifies the file-spec that receives the events, for example, [2,8]NET.LOG. This parameter can be changed while the logging state is ON. However, if this happens, the Event Logger closes the previous file being used and opens a new file with the specified name.

  MONITOR
  : Identifies the Receiver ID (see the Declare Receiver description in the DECnet/E programming manuals) of the user program that receives the events, for example, USRMON. This parameter can be changed while the logging state is ON.

SINK
: Identifies the sink node that is to receive events. The three possibilities are SINK EXECUTOR (the local node), SINK NODE *node-id* (a node in the network), or KNOWN SINKS. The default is EXECUTOR.

NOTE

The sink node parameters can be used only with EVENTS and KNOWN EVENTS.

NODE *node-id* — Specifies the node identifier for the node where one or more events originated.

STATE — Specifies the operational state of the sink. The two possible states are:

- OFF — The sink is not ready to receive events.
- ON — The sink is ready to receive events.

Examples:

The following commands cause all events to be logged on console KB3:. In this example, the event filter for the console is set to pass all events.

```
NCP>SET LOGGING CONSOLE NAME KB3:
NCP>SET LOGGING CONSOLE KNOWN EVENTS
```

The following command causes all class 3 events, together with any other events that might already be enabled on this sink, to be logged to the logging console on remote node ASPEN. In this example, the event filter for the console sink for node ASPEN is modified to pass all class 3 events, without affecting any other event types it might be filtering or passing out. Note that nothing changes on node ASPEN itself; only what the local executor sends to node ASPEN changes to include events 3.*.

```
NCP>SET LOGGING CONSOLE EVENTS 3.* SINK NODE ASPEN
```

Use the SET NODE command to create or modify node parameters in the volatile parameter file for any node other than the executor. Use the DEFINE NODE command to create or modify node parameters in the permanent parameter file for any node other than the executor.

Command Format:

SET	KNOWN NODES	ADDRESS *number*
DEFINE	NODE *node-id*	ALIAS *name*
		ALL
		CIRCUIT *circuit-id*
		COUNTER TIMER *seconds*
		NAME *nodename*
		RECEIVE ORIGINATE PASSWORD *password*
		RECEIVE ANSWER PASSWORD *password*
		TRANSMIT ORIGINATE PASSWORD *password*
		TRANSMIT ANSWER PASSWORD *password*

NOTE

The keyword parameters ADDRESS number, ALIAS name, ALL, CIRCUIT *circuit-id*, and NAME *nodename* are meaningful only with the NODE *node-id* entity identifier. The keyword parameters ALL and CIRCUIT *circuit-id* are meaningful only with the SET command.

Entity Identifiers:

KNOWN NODES

When specified with the SET command, identifies all nodes in the volatile parameter file known to the executor node.

When specified with the DEFINE command, identifies all nodes in the permanent parameter file known to the executor node.

NODE *node-id*

Specifies a node identifier.

Keyword Parameters:

ALIAS *name*

Specifies an alternate name for a node.

ALL

Copies all parameters for the specified node from the permanent parameter file ($NETPRM.SYS) to the volatile parameter file.

CIRCUIT *circuit-id*	Specifies a circuit identifier on the node.
COUNTER TIMER *seconds*	Sets a timer whose expiration causes a node counter logging event. The value range is 1 - 65535. A value of 0 disables the timer.
NAME *nodename*	Specifies a name for the node.
RECEIVE ANSWER PASSWORD *password*	Specifies the password you expect to receive if you require verification when answering your node-initialization request.
RECEIVE ORIGINATE PASSWORD *password*	Specifies the password you expect to receive if you require verification when originating a node-initialization request.
TRANSMIT ANSWER PASSWORD *password*	Specifies the password you transmit if another node requires verification from you when you answer its node-initialization request.
TRANSMIT ORIGINATE PASSWORD *password*	Specifies the password you transmit if another node requires verification from you when you originate a node-initialization request.

The passwords are RSTS-specific parameters that provide a security check from "intruders" on your network. The passwords verify which node is on the other end of the circuit you are connecting with. You use the TRANSMIT passwords if another node requires verification from you. You use the RECEIVE passwords if you require verification from another node. The ANSWER and ORIGINATE passwords provide a further check to protect your network from intruders. If a line on a node in your network is set only to receive connections from other nodes, only the ANSWER passwords need to be defined. However, if the lines on your nodes are enabled to send and receive connections, then both the ANSWER and ORIGINATE passwords should be defined.

Examples:

The following series of commands verifies that the connection request from node SANFRN to node LA is valid.

```
NCP>TELL LA SET NODE SANFRN TRANSMIT ORIGINATE PASSWORD VANILLA

NCP>TELL LA SET NODE SANFRN RECEIVE ORIGINATE PASSWORD COFFEE

NCP>TELL SANFRN SET NODE LA TRANSMIT ANSWER PASSWORD COFFEE

NCP>TELL SANFRN SET NODE LA RECEIVE ANSWER PASSWORD VANILLA
```

The following command sets the counter timer in the volatile parameter file. The counter timer affects all nodes known to the executor.

```
NCP>SET KNOWN NODES COUNTER TIMER
```

Use the SET OBJECT command to create or modify object parameters in the volatile parameter file on the executor node. Use the DEFINE OBJECT command to create or modify object parameters in the permanent parameter file on the executor node.

Command Format:

SET OBJECT *object-id*
DEFINE

FILE *file-spec*
NAME *object name*
PARAMETERS *P1, P2*
TYPE *object type*

Entity Identifier:

OBJECT *object-id* — Specifies an object identifier.

Keyword Parameters:

FILE *file-spec* — Specifies the file specification of the program. NSP automatically starts this program upon receiving a connect initiate message. The specified file must be an executable file. A device specification can be included, but it is not required. The default is the public structure where a project-programmer number must be supplied.

NAME *object name* — Identifies the receiver-id of the object when an incoming connect initiate message asks for an object by name. Object names must be unique.

PARAMETERS *P1, P2* — Specifies parameters passed to the program when it is started (see Section 2.7.2).

TYPE *object type* — Specifies the object number. This number (in the range 0 to 255) is used by NSP to identify the object when an incoming Connect Initiate Message asks for an object by number.

NOTE

When you first define an object you must specify the object-id with a number (object-type). Example: SET OBJECT 17 NAME FAL

You can then specify the TYPE keyword parameter to change FAL's object-type to another number, provided that the new number is not already defined. Example: SET OBJECT FAL TYPE 21

Examples:

The following command defines an object type number (17) and the program location for the File Access Listener (FAL) program. Because no name is specified, this object cannot be connected to by name.

```
NCP>SET OBJECT TYPE 17 FILE FAL.TSK
```

The following command identifies a user program to be started by NSP when an incoming Connect Initiate Message asks for object SERVER by name. An object type = 0 is assumed unless specified otherwise. Note that only one object can be defined for each object type, including type 0.

```
NCP>SET OBJECT NAME SERVER FILE DB2:[100,32]MYPROG.BAC
```

4.6 SHOW/LIST Commands

Use the SHOW commands to display information from the volatile parameter file about the status of the running network. Use the LIST commands to display and verify information from the permanent parameter file. Each display appears on the local node and contains information from the parameter files on the executor node. See Section 3.2 for details on displaying information.

NOTE

The keyword parameters COUNTERS and STATUS are meaningful only to the SHOW command. In the following commands, if the qualifying keyword parameter is omitted, the default is SUMMARY. If the *file-spec* is omitted, the output is typed on the terminal.

SHOW CIRCUIT
LIST

Use the SHOW CIRCUIT command to display circuit information from the volatile parameter file on the executor node. Use the LIST CIRCUIT command to display circuit information from the permanent parameter file on the executor node.

Command Format:

SHOW LIST	ACTIVE CIRCUITS CIRCUIT *circuit-id* KNOWN CIRCUITS	CHARACTERISTICS COUNTERS STATUS SUMMARY	TO *file-spec*

NOTE

The ACTIVE CIRCUITS entity identifier is meaningful only to the SHOW command.

Entity Identifiers:

ACTIVE CIRCUITS	Displays all circuits in the ON or SERVICE state.
CIRCUIT *circuit-id*	Displays a circuit identifier.
KNOWN CIRCUITS	Displays all circuits that have a name.

Keyword Parameters:

CHARACTERISTICS	Displays static local circuit information.
COUNTERS	Displays local circuit error and performance statistics.
STATUS	Displays dynamic local circuit information.
SUMMARY	Displays useful circuit information.
TO *file-spec*	Displays an output file. If a file is specified, the data is appended to that file if the file exists. Otherwise, a new file is created to receive the data. If this parameter is not specified, the data is displayed on your terminal.

Example:

The following example displays summary information for circuit DMR-1.

```
NCP> SHOW CIRCUIT DMR-1 SUMMARY
Circuit Volatile Summary as of 12-Nov-81 15:44:60

Circuit = DMC-1

State                    = On-Starting
Loopback Name            = TESTLP


NCP>
```

SHOW EXECUTOR
LIST

Use the SHOW EXECUTOR command to display executor node information from the volatile parameter file. Use the LIST EXECUTOR command to display executor node information from the permanent parameter file.

Command Format:

SHOW LIST	EXECUTOR	CHARACTERISTICS COUNTERS STATUS SUMMARY	TO *file-spec*

Entity Identifier:

EXECUTOR	Displays the specified keyword parameters for the executor node.

Keyword Parameters:

CHARACTERISTICS	Displays static information for the executor node.
COUNTERS	Displays executor node error and performance statistics.
STATUS	Displays dynamic information for the executor node.
SUMMARY	Displays the executor node's state and system identification. This is the default display type.
TO *file-spec*	Displays an output file. If a file is specified, the data is appended to that file if the file exists. Otherwise, a new file is created to receive the data. If this parameter is not specified, the data is displayed on your terminal.

Examples:

The following command displays executor node characteristics. The display includes values you have set for the executor node as well as supplemental information on the software versions of Network Management (NCP/NML), Network Services (NSP), and Routing (Transport). Note that only privileged users can see the four passwords.

```
NCP> SHOW EXECUTOR CHARACTERISTICS
Node Volatile Characteristics as of 13-Nov-81 08:04:44

Executor Node = 140 (NYC)

State                           = On
Identification                  = DECnet/E V2.0 Prototype
Management Version              = 3.0.0
Loop Count                      = 1
Loop Length                     = 128
Loop With                       = Mixed
Counter Timer                   = 0
Incoming Timer                  = 120
Outgoing Timer                  = 120
NSP Version                     = 3.2.0
Maximum Links                   = 32
Delay Factor                    = 48
Delay Weight                    = 3
Inactivity Timer                = 120
Retransmit Factor               = 5
Routing Version                 = 1.3.0
Type                            = Routing
Routing Timer                   = 120
Maximum Address                 = 255
Maximum Circuits                = 6
Maximum Cost                    = 255
Maximum Hops                    = 16
Maximum Visits                  = 20
Maximum Buffers                 = 40
Buffer Size                     = 568
Alias                           = PHILLY
Default Account                 = [3,100]
Data Xmit Queue Max             = 8
Int/LS Queue Max                = 5
Transmit ANSWER PASSWORD        = MOCHA
Transmit ORIGINATE PASSWORD     = BANANA
Receive ANSWER PASSWORD         = MOCHA
Receive ORIGINATE PASSWORD      = BANANA
NCP>
```

The following command displays counter information for the executor node. Refer to Appendix B for a description of each node counter.

```
NCP> SHOW EXECUTOR COUNTERS
Node Volatile Counters as of 13-Nov-81 08:07:41

Executor Node = 140 (NYC)

        65534  Seconds Since Last Zeroed
       125314  Bytes Received
       125389  Bytes Sent
         1416  Messages Received
         1423  Messages Sent
           97  Connects Received
           97  Connects Sent
            0  Response Timeouts
            0  Received Connect Resource Errors
           11  Maximum Logical Links Active
            4  Aged Packet Loss
            0  Node Unreachable Packet Loss
            0  Node Out-of-range Packet Loss
            0  Oversized Packet Loss
            0  Packet Format Error
            0  Partial Routing Update Loss
            0  Verification Reject
          120  Current Reachable Nodes
          127  Maximum Reachable Nodes


NCP>
```

The following command displays status information for the executor node. The display includes the operational state for the executor node and dynamic information for the network as perceived by the executor node.

```
NCP> SHOW EXECUTOR STATUS
Node Volatile Status as of 13-Nov-81 08:09:44

Executor Node =140 (NYC)

State                     = On
Active Links              = 2
Type                      = Routing
NCP>
```

The following command displays summary information for the executor node. The summary display is similar to the status display.

```
NCP> SHOW EXECUTOR SUMMARY
Node Volatile Summary as of 13-Nov-81 08:11:49

Executor Node =140 (NYC)

State                     = On
Identification            = DECnet/E V2.0 Prototype
Active Links              = 2
NCP>
```

Use the SHOW LINE command to display line information from the volatile parameter file on the executor node. Use the LIST LINE command to display information from the permanent parameter file on the executor node.

Command Format:

SHOW LIST	ACTIVE LINES KNOWN LINES LINE line-id	CHARACTERISTICS COUNTERS STATUS SUMMARY	TO *file-spec*

NOTE

The entity identifier ACTIVE LINES is meaningful only to the SHOW command. The keyword parameters COUNTERS and STATUS are meaningful only to the SHOW command.

Entity Identifiers:

ACTIVE LINES	Displays information for all lines whose circuits are in the ON or SERVICE state.
KNOWN LINES	Displays information for all lines that have a name.
LINE *line-id*	Displays a line identifier.

Keyword Parameters:

CHARACTERISTICS	Displays static line information.
COUNTERS	Displays line error and performance statistics.
STATUS	Displays dynamic line information.
SUMMARY	Displays the line state. This is the default display type.
TO *file-spec*	Displays an output file. If a file is specified, the data is appended to that file if the file exists. Otherwise, a new file is created to receive the data. If this parameter is not specified, the data is displayed on your terminal.

Examples:

The following command displays line characteristics for all active lines, for example, those whose state is ON. The display includes values for line parameters that you set for individual lines.

```
NCP> SHOW ACTIVE LINES CHARACTERISTICS
Active Line Volatile Characteristics as of 13-Nov-81 08:13:36

Line = DMC-0

State                   = On
Receive Buffers         = 4
Controller              = Normal
Duplex Mode             = Full Duplex
Protocol Type           = DDCMP DMC
Retransmit Timer        = 3000
Verification            = Off

Line = DMC-1

State                   = On-Starting
Receive Buffers         = 4
Controller              = Normal
Duplex Mode             = Full Duplex
Protocol Type           = DDCMP DMC
Retransmit Timer        = 3000
Verification            = Off

Line = DMC-4

State                   = On-Starting
Receive Buffers         = 4
Controller              = Normal
Duplex Mode             = Half Duplex
Protocol Type           = DDCMP DMC
Retransmit Timer        = 3000
Verification            = Off


NCP>
```

The following command displays status information for all known lines connected to the executor node. The display includes the current state of the line.

```
NCP> SHOW KNOWN LINES STATUS
Known Line Volatile Status as of 13-Nov-81 08:14:60

Line = DMC-0

State                   = On

Line = DMC-1

State                   = On-Starting

Line = DMC-2

State                   = Off
```

```
Line = DMC-3

State                    = Off

Line = DMC-4

State                     On-Starting

Line  =DMC-5

State                    = Off

Line = DMP-0

State                    = Off


NCP>
```

The following command displays summary information for line DMC-4. The display is the same for status information.

```
NCP> SHOW LINE DMC-4 SUMMARY
Line Volatile Summary as of 13-Nov-81 08:17:45

Line =DMC-4

State                    = On-Starting

NCP>
```

Use the SHOW LINKS command to display dynamic logical link information available to the executor node.

Command Format:

SHOW	ACTIVE LINKS	STATUS TO *file-spec*
	KNOWN LINKS	SUMMARY
	LINK *link-id*	

NOTE

A link is a logical path between two nodes. Links are identified by "link addresses" which are arbitrary unique numbers assigned by local and remote nodes. The local link address (lla) is the identifier assigned to the link by the local node. A remote link address (rla) is the identifier assigned to the link by a remote node.

Entity Identifiers:

ACTIVE LINKS	Displays information about all active logical links.
KNOWN LINKS	Displays information about all logical links known to the executor node.
LINK *link-id*	Displays a logical link identifier. A logical link identifier is a number assigned by NSP to the logical link when the link is created.

Keyword Parameters:

STATUS	Displays dynamic logical link information.
SUMMARY	Displays useful logical link information.
TO *file-spec*	Displays an output file. If a file is specified, the data is appended to that file if the file exists. Otherwise, a new file is created to receive the data. If this parameter is not specified, the data is displayed on your terminal.

Examples:

The following command displays information for active links known to the executor node.

```
NCP> SHOW ACTIVE LINKS
Active Link Volatile Summary as of 13-Nov-81 11:27:55

Local Link Address = 57116

RLA                     = 30755
Link State              = RUN (Link established)
Node                    = 232 VIENNA
Receiver Name           = NML024
Receiver Job Number     = 24


Local Link Address = 55321

RLA                     = 12746
Link State              = RUN (Link established)
Node                    = 143 MONACO
Receiver Name           = FAL015
Receiver Job Number     = 15


Local Link Address = 46339

RLA                     = 20186
Link State              = RUN (Link established)
Node                    = 135 MILAN
Receiver Name           = NWPK08
Receiver Job Number     = 8


Local Link Address = 22596

RLA                     = 10312
Link State              = CIR (CI received)
Node                    = 100 PARIS
Receiver Name           = NWPK08
Receiver Job Number     = 8


Local Link Address = 9614

RLA                     = 52276
Link State              = RUN (Link established)
Node                    = 146 LONDON
Receiver Name           = FAL016
Receiver Job Number     = 16


NCP>
```

The following command displays status information for active links known to the executor node.

```
NCP> TELL ARK SHOW KNOWN LINKS STATUS
Known Link Volatile Status as of 8-Dec-81 10:29:24

Local Link Address = 23496

RLA                          = 23945
ULA                          = 1
Link State                   = RUN (Link established)
Node                         = 140 NYC
```

```
Receiver Name                = NCP022
Receiver Job Number          = 22
Receiver RIB Number          = 0
Local Flow Control           = None
Remote Flow Control          = None
Local Data Request Count     = 0
Remote Data Request Count    = 0
Local Intr Request Count     = 1
Remote Intr Request Count    = 1

Local Link Address = 23945

RLA                          = 23496
ULA                          = 1
Link State                   = RUN (Link established)
Node                         = 140 NYC
Receiver Name                = NML029
Receiver Job Number          = 29
Receiver RIB Number          = 0
Local Flow Control           = None
Remote Flow Control          = None
Local Data Request Count     = 0
Remote Data Request Count    = 0
Local Intr Request Count     = 1
Remote Intr Request Count    = 1

Local Link Address = 22149

RLA                          = 17678
ULA                          = 1
Link State                   = RUN (Link established)
Node                         = 146 LONDON
Receiver Name                = NWPK07
Receiver Job Number          = 7
Receiver RIB Number          = 0
Local Flow Control           = Message
Remote Flow Control          = None
Local Data Request Count     = 2
Remote Data Request Count    = 0
Local Intr Request Count     = 1
Remote Intr Request Count    = 1

Local Link Address = 22616

RLA                          = 4999
ULA                          = 2
Link State                   = RUN (Link established)
Node                         = 135 MILAN
Receiver Name                = EVTTRN
Receiver Job Number          = 8
Receiver RIB Number          = 2
Local Flow Control           = None
Remote Flow Control          = None
Local Data Request Count     = 0
Remote Data Request Count    = 0
Local Intr Request Count     = 1
Remote Intr Request Count    = 1
```

Use the SHOW LOGGING command to display logging information from the volatile parameter file on the executor node. Use the LIST LOGGING command to display logging information from the permanent parameter file on the executor node.

Command Format:

SHOW LIST	ACTIVE LOGGING KNOWN LOGGING LOGGING *sink-type* CONSOLE FILE MONITOR	CHARACTERISTICS EVENTS STATUS SUMMARY	EXECUTOR KNOWN SINKS SINK NODE *node-id*	TO *file-spec*

NOTE

The ACTIVE LOGGING entity identifier is meaningful only to the SHOW command. If the SINK NODE keyword parameter is omitted, the default is SINK EXECUTOR.

Entity Identifiers:

ACTIVE LOGGING	Displays information for all active logging, for example, for all sinks on the executor node whose logging state is ON.
KNOWN LOGGING	Displays information for all logging sink types known to the executor node.
LOGGING *sink-type*	Displays information for the console, file, or monitor logging sink.

Keyword Parameters:

CHARACTERISTICS	Displays static logging information.
EVENTS	Displays event logging information.
sink-node	Identifies a sink node for which information is to be displayed. The options are SINK NODE *node-id*, SINK EXECUTOR, and KNOWN SINKS. Information about the executor node is displayed by default.
STATUS	Displays dynamic logging information.

SUMMARY — Displays selected information pertaining to the CHARACTERISTICS, EVENTS, and STATUS keywords for a logging sink node or sink type. This is the default display type.

TO *file-spec* — Displays an output file. If a file is specified, the data is appended to that file if the file exists. Otherwise, a new file is created to receive the data. If this parameter is not specified, the data is displayed on your terminal.

Examples:

The following command displays events being logged to the logging file sink for all known sinks. The display lists the sink node and the actual events that are sent to the logging file at that node.

```
NCP> SHOW LOGGING FILE EVENTS KNOWN SINKS
Logging Volatile Events as of 13-Nov-81 12:30:44

Logging sink = File

Sink Node               = 140 NYC
                                      0.0,2-7
                                      2.0-1
                                      3.0-2
                                      4.0-14
                                      5.0-9
                                      6.0-5
                                      33.0
                                      34.0-1
                                      36.0

Logging sink = File

Sink Node               = 135 LONDON


NCP>
```

The following command displays status information for all active logging. This display identifies each logging component and information particular to that component. The status display is similar to the events display except that the name of the component and its operational state are included in this display.

```
NCP> SHOW ACTIVE LOGGING STATUS
Active Logging Volatile Status as of 13-Nov-81 12:31:64

Logging sink = Console

State                   = On

Logging sink = Monitor

State                   = On


NCP>
```

The following command displays summary information to the executor node for all known logging. The summary display is the same as the events display.

```
NCP> SHOW KNOWN LOGGING SUMMARY EXECUTOR
Known Logging Volatile Summary as of 13-Nov-81 12:34:17

Logging sink = Console

State                   = On
Name                    = KB0:
Sink Node               = 140 NYC
                                    2.1
                                    Node 143 (PARIS) 4.14
                                    33.0

Logging sink = File

State                   = Off
Name                    = SY:[1,2]EVTLOG.LOG
Sink Node               = 140 NYC
                                    0.0,2-7
                                    2.0-1
                                    3.0-2
                                    4.0-14
                                    5.0-9
                                    6.0-5
                                    33.0
                                    34.0-1
                                    36.0

Logging sink = Monitor

State                   = On
Name                    = NETMON
Sink Node               = 140 NYC
                                    Circuit DMC-1 0.1
                                    4.10


NCP>
```

SHOW NODE
LIST

Use the SHOW NODE command to display node information, from memory or from the volatile parameter file on the executor node. Use the LIST NODE command to display node information from the permanent parameter file on the executor node.

Command Format:

SHOW	ACTIVE NODES	CHARACTERISTICS	TO *file-spec*
LIST	EXECUTOR	COUNTERS	
	KNOWN NODES	STATUS	
	LOOP NODES	SUMMARY	
	NODE *node-id*		

NOTE

The entity identifiers ACTIVE NODES and LOOP NODES are meaningful only to the SHOW command. The keyword parameters affect only the SHOW commands. If the keyword parameter is omitted, the default is SUMMARY. If *file-spec* is omitted, output is typed on the terminal.

Entity Identifiers:

ACTIVE NODES	Displays information for all nodes Transport perceives as reachable.
EXECUTOR	Displays information for the executor node.
KNOWN NODES	Displays information for all nodes known to the executor.
LOOP NODES	Displays information for all loop nodes.
NODE *node-id*	Displays a node identifier.

Keyword Parameters:

CHARACTERISTICS	Displays static node information.
COUNTERS	Displays node error and performance statistics. Note that counters for a node might not be available when there are no active links to that node. This information is not kept indefinitely. Executor counters are always available, however.
STATUS	Displays dynamic node information.

SUMMARY — Displays the node state, loop node name, and adjacent node name and address. This is the default display type.

TO *file-spec* — Displays an output file. If a file is specified, the data is appended to that file if the file exists. Otherwise, a new file is created to receive the data. If this parameter is not specified, the data is displayed on your terminal.

Examples:

The following command displays characteristics for all active nodes. The display includes values you have set for both the executor node and remote nodes.

```
NCP> SHOW ACTIVE NODES CHARACTERISTICS
Active Node Volatile Characteristics as of 13-Nov-81 13:18:27

Executor Node = 122 (SANFRN)

State                    = On
Identification           = DECnet/E V2.0 Prototype
Management Version       = 3.0.0
Loop Count               = 4
Loop Length              = 256
Loop With                = 2
Counter Timer            = 0
Incoming Timer           = 120
Outgoing Timer           = 120
NSP Version              = 3.2.0
Maximum Links            = 32
Delay Factor             = 48
Delay Weight             = 3
Inactivity Timer         = 120
Retransmit Factor        = 5
Routing Version          = 1.3.0
Type                     = Routing
Routing Timer            = 120
Maximum Address          = 255
Maximum Circuits         = 6
Maximum Cost             = 255
Maximum Hops             = 16
Maximum Visits           = 20
Maximum Buffers          = 40
Buffer Size              = 568
Alias                    = CITY
Default Account          = 3,100
Data Xmit Queue Max      = 8
Int/LS Queue Max         = 5

Remote Node = 5 (TAHOE)

Counter Timer            = 0

Remote Node = 6 (RENO)

Counter Timer            = 0

Remote Node = 8 (VEGAS)

Counter Timer            = 0
```

```
Remote Node = 12 (CHCAGO)

Counter Timer            = 0

Remote Node = 17 (DENVER)

Counter Timer            = 0

Remote Node = 18 (DALLAS)

Counter Timer            = 0

Remote Node = 19 (PHILLY)

Counter Timer            = 0

Remote Node = 20 (BOSTON)

Counter Timer            = 0
```

The following command displays status information for remote node MONACO. Note that full status information is displayed only for reachable nodes.

```
NCP> SHOW NODE MONACO STATUS
Node Volatile Status as of 13-Nov-81 13:41:25

Remote Node =135 (MONACO)

State                    = Reachable
Type                     = Routing
Cost                     = 1
Hops                     = 1
Circuit                  = DMC-2


NCP>
```

The following command displays summary information for all known nodes. The summary display contains a subset of the information contained in the status display. Note that the summary display also includes loop nodes and the circuit associated with the loop node name.

```
NCP> SHOW KNOWN NODES SUMMARY
Known Node Volatile Summary as of 13-Nov-81 14:43:07

Executor Node = 110 (WASHDC)

State                    = On
Identification           = DECnet/E V2.0 Prototype
Active Links             = 4

Remote Node = 1 (DENVER)

State                    = Unreachable

Remote Node = 2 (DALLAS)

State                    = Unreachable
```

```
Remote Node = 3 (ASPEN)

State                    = Unreachable

Remote Node = 4 (AUSTIN)

State                    = Unreachable

Remote Node = 5 (SPOKAN)

State                    = Reachable
Circuit                  = DMC-0
```

The following command displays counter information for remote node DENVER. Note that remote node counters are a subset of those maintained for the local node. Refer to Appendix B for a description of each node counter.

```
NCP> SHOW NODE DENVER COUNTERS
Node Volatile Counters as of 13-Nov-81 15:04:32

Remote Node = 135 (DENVER)

        8776  Seconds Since Last Zeroed
      302524  Bytes Received
      302364  Bytes Sent
        1431  Messages Received
        1486  Messages Sent
          30  Connects Received
          16  Connects Sent
          20  Response Timeouts
           0  Received Connect Resource Errors


NCP>
```

The following command displays status information for all loop nodes.

```
NCP> SHOW LOOP NODES STATUS
Loop Node Volatile Status as of 13-Nov-81 15:15:47

Loop Node =  (TESTND)

State                    = Reachable
Active Links             = 0
Delay                    = 1
Type                     = Routing
Circuit                  = DMC-4

Loop Node =  (TESTLP)

State                    = Reachable
Active Links             = 0
Delay                    = 1
Type                     = Routing
Circuit                  = DMC-1


NCP>
```

SHOW OBJECT
LIST

Use the SHOW OBJECT command to display object information from the volatile parameter file on the executor node. Use the LIST OBJECT command to display object information from the permanent parameter file on the executor node.

Command Format:

SHOW KNOWN OBJECTS CHARACTERISTICS TO *file-spec*
LIST OBJECT *object-id* SUMMARY

Entity Identifiers:

KNOWN OBJECTS	Displays information for all objects known to the executor node.
OBJECT *object-id*	Displays an object identifier.

Keyword Parameters:

CHARACTERISTICS	Displays static object information.
SUMMARY	Displays information for object characteristics and status. This is the default display type.
TO *file-spec*	Displays the output file.

Examples:

The following command displays object characteristics for the FAL object. This format displays values that you have set for the object.

```
NCP> SHOW OBJECT FAL CHARACTERISTICS
Object Volatile Characteristics as of 13-Nov-81 15:27:16

Object = 17

Name                    = FAL
File                    = DR3:10,250FALX .SIL
Object Parameter 1      = 0
Object Parameter 2      = 511


NCP>
```

The following command displays summary information for the NML (Network Management Listener) object.

```
NCP> SHOW OBJECT 19 SUMMARY
Object Volatile Summary as of 13-Nov-81 15:37:46

Object = 19

Name                    = NML


NCP>
```

4.7 CLEAR/PURGE Commands

Use the CLEAR commands to change the volatile parameter file during network operation. Use the PURGE commands to change the permanent parameter file. See Section 2.10 for more information about the parameter files.

Use the CLEAR CIRCUIT command to delete circuit parameters from the volatile parameter file on the executor node. Use the PURGE CIRCUIT command to delete circuit parameters from the permanent parameter file on the executor node.

Command Format:

CLEAR	CIRCUIT *circuit-id*	ALL
PURGE	KNOWN CIRCUITS	OWNER

Entity Identifiers:

CIRCUIT *circuit-id*	Deletes the specified circuit identifier from the parameter file.
KNOWN CIRCUITS	Deletes the specified circuit parameters known to the executor node from the parameter file.

Keyword Parameters:

ALL	Deletes all circuit parameters either for the specified circuit or for all circuits known to the executor node from the parameter file.
OWNER	Relinquishes ownership of the circuit for DECnet use.

Example:

The following command deletes all circuit parameters in the volatile parameter file known to the executor node.

```
NCP>CLEAR KNOWN CIRCUITS ALL
```

CLEAR EXECUTOR
PURGE

Use the CLEAR EXECUTOR command to delete node parameters from the volatile parameter file on the executor node. Use the PURGE EXECUTOR command to delete node parameters from the permanent parameter file on the executor node.

Command Format:

CLEAR	EXECUTOR	ALIAS
PURGE		ALL
		COUNTER TIMER
		DEFAULT ACCOUNT
		IDENTIFICATION
		INCOMING TIMER
		NAME
		OUTGOING TIMER

Entity Identifer:

EXECUTOR — When specified with the CLEAR command, deletes the designated executor from the volatile parameter file.

When specified with the PURGE command, deletes the designated executor from the permanent parameter file.

Keyword Parameters:

ALIAS — When specified with the CLEAR command, deletes the alternate name to designate the executor node from the volatile parameter file.

When specified with the PURGE command, deletes the alternate name to designate the executor node from the permanent parameter file.

ALL — When specified with the CLEAR command, deletes all parameters from the volatile parameter file on the executor node. For the CLEAR command, this parameter is valid only when the executor is OFF.

When specified with the PURGE command, deletes all parameters from the permanent parameter file on the executor node.

COUNTER TIMER — Disables the counter timer.

DEFAULT ACCOUNT	Deletes the nonprivileged account assigned to network programs from the parameter file.
IDENTIFICATION	Deletes the node identification string for the executor node from the parameter file. For the CLEAR command, this parameter is valid only when the executor is OFF.
INCOMING TIMER	Deletes the incoming timer value for the local node from the parameter file. For the CLEAR command, this parameter is valid only when the executor is OFF.
NAME	Deletes the name associated with the specified node. For the CLEAR command, this parameter is valid only when the executor is OFF.
OUTGOING TIMER	Deletes the outgoing timer value for the local node from the parameter file.

Example:

The following command deletes the node's identification string from the volatile parameter file.

```
NCP>CLEAR EXECUTOR IDENTIFICATION
```

CLEAR EXECUTOR NODE

Use the CLEAR EXECUTOR NODE command to re-establish the default executor node (local node) as the executor for all NCP commands. Note that it is invalid to use the TELL prefix with this command.

Command Format:

CLEAR EXECUTOR NODE

Entity Identifier:

EXECUTOR Deletes the designated executor node for NCP.

Keyword Parameter:

NODE Deletes the node identifier designated to the executor for NCP.

Example:

The following command sets the executor node to node AGANA.

```
NCP>SET EXECUTOR NODE AGANA
```

The following command clears the executor node designation previously defined by the SET EXECUTOR NODE command and reinstates the default executor node (local node).

```
NCP>CLEAR EXECUTOR NODE
```

Use the CLEAR LINE command to delete line parameters from the volatile parameter file on the executor node. Use the PURGE LINE command to delete line parameters from the permanent parameter file on the executor node.

Command Format:

CLEAR KNOWN LINES ALL
PURGE LINE *line-id*

Entity Identifiers:

KNOWN LINES — Deletes the specified line parameters known to the executor node from the parameter file.

LINE *line-id* — Deletes the line identifier from the parameter file.

Keyword Parameter:

ALL — Deletes all parameters for either the specified line or for all known lines from the parameter file. For the CLEAR command, this parameter is valid only when the line is OFF.

Example:

The following command deletes all parameter entries for line DMC-0 in the volatile parameter file. As a result, the line no longer exists for the executor node as far as SET/SHOW commands are concerned. However, the line's parameters will still be in the permanent parameter file if a DEFINE command was ever issued for the line.

```
NCP>CLEAR LINE DMC-0 ALL
```

CLEAR LOGGING
PURGE

Use the CLEAR LOGGING command to delete specified logging parameters from the volatile parameter file. Use the PURGE LOGGING command to delete specified logging parameters from the permanent parameter file.

Command Format:

CLEAR	KNOWN LOGGING	ALL EVENTS	SINK EXECUTOR
PURGE	LOGGING CONSOLE	CIRCUIT *circuit-id*	NODE *node-id*
	LOGGING FILE	EVENTS *event-list*	KNOWN SINKS
	LOGGING MONITOR	KNOWN EVENTS	
		LINE *line-id*	
		NAME	
		NODE *node-id*	

NOTE

The keyword parameter NAME affects only the executor node. If the SINK parameter is omitted, SINK EXECUTOR is the default. The source qualifier parameters CIRCUIT *circuit-id*, LINE *line-id*, and NODE *node-id* are valid only when specified with an EVENTS parameter.

Entity Identifiers:

KNOWN LOGGING	Deletes the specified parameters for all logging known to the executor node from the parameter file.
LOGGING CONSOLE	Deletes the parameters controlling logging to the console from the parameter file.
LOGGING FILE	Deletes the parameters controlling logging to a file from the parameter file.
LOGGING MONITOR	Deletes the parameters controlling logging to a monitor program from the parameter file.

Keyword Parameters:

ALL EVENTS	Deletes all parameters controlling one or more specified logging sinks from the parameter file. The logging sinks are no longer recognized by the network.
CIRCUIT *circuit-id*	Deletes the circuit identifier as a source of events to be logged.

EVENTS *event-list*
: Identifies the classes and types of events to be deleted from the filters for the given logging sinks.

KNOWN EVENTS
: Deletes all event classes and types from known to the executor from the parameter file.

LINE *line-id*
: Deletes the line identifier as a source of events to be logged.

NAME
: When specified with the CLEAR command, deletes the name of the logging sink from the volatile parameter file. For the CLEAR command, this parameter is valid only when the sink is in the OFF state.

  When specified with the PURGE command, deletes the name of the logging sink from the permanent parameter file.

NODE *node-id*
: Deletes the node identifier as a source of events to be logged.

sink-node
: Identifies the node where the event logger logs the events. The three possibilities are SINK EXECUTOR, SINK NODE *node-id*, and KNOWN SINKS.

  Only events being logged to this identified node are deleted from the parameter file. If this parameter is omitted, the executor node is assumed to be the sink node affected. This parameter can be used only in combination with the EVENTS parameter.

Examples:

The following command clears from the event filter all class 2 events for the logging file at the executor node. Thus, class 2 events are no longer logged to that file.

```
NCP>CLEAR LOGGING FILE EVENTS 2.*
```

The following command clears the entire event filter for the logging console. Thus, no local events are logged to the console except events from remote nodes that use your node as a logging sink.

```
NCP>CLEAR LOGGING CONSOLE KNOWN EVENTS
```

The following command clears the logging console name from the parameter file. This parameter can be cleared only when the logging console is in the OFF state.

```
NCP>CLEAR LOGGING CONSOLE NAME
```

CLEAR NODE
PURGE

Use the CLEAR NODE command to delete node parameters from the volatile parameter file on the executor node. Use the PURGE NODE command to delete node parameters from the permanent parameter file on the executor node.

Some volatile executor node parameters can be changed only when the network is shut down. Permanent executor node parameters can be changed while the network is running. You can clear all parameters for a remote node (NODE *node-id*) at any time.

Command Format:

CLEAR	NODE *node-id*	ALIAS
PURGE	KNOWN NODES	ALL
		CIRCUIT
		COUNTER TIMER
		NAME

NOTE

The keyword parameter CIRCUIT is meaningful only to the CLEAR NODE *node-id* command. The keyword parameters meaningful to CLEAR/PURGE KNOWN NODES are: ALL, COUNTER TIMER, and NAME.

Entity Identifiers:

KNOWN NODES	Deletes specified parameters from the parameter files for all nodes known to the executor node.
NODE *node-id*	Deletes the specified node address or name from the parameter file.

Keyword Parameters:

ALIAS	Deletes the alternate node name from the parameter file.
ALL	Deletes all parameters for either the specified node or all known nodes from the parameter file.
COUNTER TIMER	Deletes the counter timer value from the parameter file. This turns off automatic logging of node counters.

CIRCUIT *circuit-id*	Deletes the loop node name for the specified circuit. The loop node to be removed is identified by its name. The keyword CIRCUIT is used to indicate that a loop node (for example, a node associated with a circuit) is to be removed. CLEAR KNOWN NODES CIRCUIT removes all loop nodes.
NAME	Deletes the name of the specified node or of all known nodes from the parameter file.

Examples:

The following command deletes all parameter entries for node PHILLY in the volatile parameter file. As a result, the node name no longer exists for local DECnet/E software.

```
NCP>CLEAR NODE PHILLY ALL
```

The following command deletes the name from node 14.

```
NCP>CLEAR NODE 14 NAME
```

CLEAR OBJECT
PURGE

Use the CLEAR OBJECT command to delete object parameters from the volatile parameter file on the executor node. Use the PURGE OBJECT command to delete object parameters from the permanent parameter file on the executor node.

Command Format:

CLEAR KNOWN OBJECTS ALL
PURGE OBJECT *object-id*

NOTE

The object name or object type specified in the CLEAR or PURGE OBJECT command must correspond to those specified when the object was defined. If the object was defined by type only, the *object-id* must consist only of the object type. If the object was defined by name only, the *object-id* must consist only of the object name. If the object was defined with both a name and type, the *object-id* may consist of either the name or the type.

Entity Identifiers:

KNOWN OBJECTS — Deletes the specified parameters for all objects known to the executor node.

OBJECT *object-id* — Deletes the object identifier from the parameter file.

Keyword Parameter:

ALL — Deletes all parameters for the specified object or for all objects known to the executor node.

Example:

The following command deletes all parameter entries for object BERYT in the volatile parameter file. As a result, the object no longer exists for the executor node.

```
NCP>CLEAR OBJECT BERYT
```

Use the CLEAR SYSTEM command after DECnet/E network is turned off to delete the values from the volatile parameter file. Then use the SET SYSTEM command to reset the volatile parameter file from the permanent parameter file.

NOTE

The CLEAR SYSTEM command stops the event logger. If you want to shut down the network completely, it is recommended that you use the NETOFF utility (see Section 3.6).

Command Format:

CLEAR SYSTEM

Example:

The following commands turn off the network and delete the values from the volatile parameter file during network operation.

```
NCP>SET EXECUTOR STATE OFF
NCP>CLEAR SYSTEM
```

4.8 Special NCP Commands

The four commands noted in this section are special NCP commands used to control and monitor your network.

4.8.1 ABORT LINK

Use the ABORT LINK command to immediately disconnect a logical link. See any of the DECnet/E programming manuals for a complete description of logical links.

Command Format:

ABORT LINK *link-id*

Entity Identifier:

LINK *link-id* Specifies the local link address (LLA). The value range is 1-32767. Use the SHOW ACTIVE LINKS command to obtain the value of the *link-id* you want to abort.

Example:

The following command immediately disconnects link 254.

```
NCP>ABORT LINK 254
```

4.8.2 EXIT

The EXIT command causes an orderly termination of NCP and returns to the RSTS/E command level.

Command Format:

EXIT

NOTE

Typing CTRL/C or CTRL/Z in response to the NCP prompt (NCP>) has the same effect as typing EXIT. Typing CTRL/C during command execution causes NCP to terminate and returns control to the RSTS/E command level. It is recommended that you never use CTRL/C to leave a program because doing so may leave the volatile parameter file unsynchronized with information kept dynamically in memory. This does not cause a system crash but could result in system problems later on.

4.8.3 LOOP

Use the LOOP NODE command to test a node in the network. If that node is the executor node, use the LOOP EXECUTOR command. Each command causes test blocks of data to be transmitted to the specified node. The parameters are optional and can be entered in any order.

Command Format:

LOOP EXECUTOR [*access control information*] COUNT *number*
NODE *node-id* LENGTH *number*
WITH
MIXED
ONES
ZEROS

NOTE

Access control information can appear in any order after the *node-id* and is optional (see Section 2.9.3).

Entity Identifiers:

EXECUTOR — Identifies the executor node for loopback testing.

NODE *node-id* — Identifies the node for loopback testing.

Keyword Parameters:

COUNT *number* — Specifies the number of blocks to be sent during loopback testing. The count must be a decimal integer from 1 to 65535. If the parameter is omitted, only one block is looped.

LENGTH *number* — Specifies the length (in bytes) of the blocks to be sent during loopback testing. The length must be a decimal integer in the range of 1 through 65535. If the parameter is omitted, a block length of 128 bytes is used.

WITH — Specifies the type of binary information to be sent during testing. Three types of data can be sent:

MIXED (combination of ones and zeros)
ONES (all one bits)
ZEROS (all zero bits)

If omitted, MIXED is the default field.

Examples:

The following command creates a loop node name (TEST) for the associated circuit.

```
NCP>SET NODE TEST CIRCUIT DMC-0
```

The following command initiates a node level loopback test with the loop node name TEST.

```
NCP>LOOP NODE TEST
```

4.8.4 ZERO

Use the ZERO command to clear the values for circuit, line, or node counters on the executor node.

Command Format:

	ACTIVE CIRCUITS LINES NODES	
	KNOWN CIRCUITS LINES NODES	
ZERO	CIRCUIT *circuit-id* EXECUTOR LINE *line-id* NODE *node-id*	COUNTERS

Entity Identifiers:

ACTIVE CIRCUITS LINES NODES	Resets the counters for all active circuits, lines, or nodes.
CIRCUIT *circuit-id*	Resets the counters for the specified circuit.
EXECUTOR	Resets the counters for the executor node.
KNOWN CIRCUITS LINES NODES	Resets the counters for all known circuits, lines, or nodes.
LINE *line-id*	Resets the counters for the specified line.
NODE *node-id*	Resets the counters for the specified node.

Keyword Parameter:

COUNTERS — Indicates that the specified entity counters are to be zeroed. This parameter is optional and is included for clarity only.

Example:

The following command resets all node counters maintained by the executor for node NYC.

```
NCP>ZERO NODE NYC
```

4.9 TELL

Use the TELL prefix to identify the executor node for an NCP command. TELL sets the node as the executor only for the one command that it prefixes. This command allows you, optionally, to specify access control information as part of the node-id.

Command Format:

TELL node-id [*access control information*] command

NOTE

Access control information can appear in any order after the node-id and is optional (see Section 2.9.3).

Entity Identifier:

node-id Specifies a node name or address.

Example:

The following command sets the executor to node BOSTON (for this one command) and establishes new values for LINE DMC-2 in the volatile parameter file on node BOSTON.

```
NCP>TELL BOSTON [access control information] SET LINE DMC-2 ALL
```

Appendix A
NCP Command Summary

This appendix summarizes the NCP commands, components, and parameters that you can use for network management. The commands are listed alphabetically by command name.

ABORT	LINK *link-id*		
CLEAR	KNOWN CIRCUITS	ALL	
	CIRCUIT *circuit-id*	OWNER	
CLEAR	EXECUTOR	ALIAS	
		ALL	
		COUNTER TIMER	
		DEFAULT ACCOUNT	
		IDENTIFICATION	
		INCOMING TIMER	
		NAME	
		OUTGOING TIMER	
CLEAR	EXECUTOR	NODE	
CLEAR	KNOWN LINES	ALL	
	LINE *line-id*		
CLEAR	KNOWN LOGGING	ALL EVENTS	
	LOGGING CONSOLE	EVENTS *evt-list*	
	LOGGING FILE	KNOWN EVENTS	
	LOGGING MONITOR	NAME	
CLEAR	KNOWN LOGGING	ALL EVENTS	CIRCUIT *circuit-id*
	LOGGING CONSOLE	EVENTS *evt-list*	LINE *line-id*
	LOGGING FILE	KNOWN EVENTS	NODE *node-id*
	LOGGING MONITOR		

```
CLEAR   KNOWN LOGGING          ALL EVENTS          SINK
        LOGGING CONSOLE        EVENTS evt-list         EXECUTOR
        LOGGING FILE           KNOWN EVENTS            NODE node-id
        LOGGING MONITOR

CLEAR   KNOWN LOGGING          ALL EVENTS          CIRCUIT circuit-id    SINK
        LOGGING CONSOLE        EVENTS evt-list     LINE line-id          EXECUTOR
        LOGGING FILE           KNOWN EVENTS        NODE node-id          NODE node-id
        LOGGING MONITOR

CLEAR   KNOWN NODES            ALIAS
        NODE node-id           ALL
                               CIRCUIT
                               COUNTER TIMER
                               NAME

CLEAR   KNOWN OBJECTS          ALL
        OBJECT object-id

CLEAR   SYSTEM

DEFINE  ALL LOGGING    STATE
                           OFF
                           ON

DEFINE  KNOWN CIRCUITS         AUTORESTART
        CIRCUIT circuit-id                 OFF
                                           ON
                               COST cost
                               HELLO TIMER seconds
                               OWNER EXECUTOR
                               TRIBUTARY trib-address

DEFINE  EXECUTOR       ADDRESS number
                       ALIAS name
                       BUFFER SIZE number
                       COUNTER TIMER seconds
                       DATA TRANSMIT QUEUE MAXIMUM number
                       DEFAULT ACCOUNT user-id
                       DELAY FACTOR number
                       DELAY WEIGHT number
                       IDENTIFICATION string
                       INACTIVITY TIMER seconds
                       INCOMING TIMER seconds
                       INTERRUPT TRANSMIT QUEUE MAXIMUM number
                       MAXIMUM ADDRESS number
                       MAXIMUM BUFFERS number
                       MAXIMUM CIRCUITS number
                       MAXIMUM COST number
                       MAXIMUM HOPS number
                       MAXIMUM LINKS number
                       MAXIMUM VISITS number
```

```
                    NAME node name
                    OUTGOING TIMER seconds
                    RETRANSMIT FACTOR number
                    ROUTING TIMER seconds
                    VOLATILE PARAMETER FILE NAME file-spec

DEFINE   KNOWN LINES    CONTROLLER
         LINE line-id                 LOOPBACK
                                      NORMAL
                        DEAD TIMER milliseconds
                        DELAY TIMER milliseconds
                        DUPLEX
                                FULL
                                HALF
                        PROTOCOL
                                  DDCMP CONTROL
                                  DDCMP DMC
                                  DDCMP POINT
                                  DDCMP TRIBUTARY
                        RECEIVE BUFFER number
                        RETRANSMIT TIMER milliseconds
                        VERIFICATION
                                    ANSWER
                                    OFF
                                    ORIGINATE

DEFINE   KNOWN LOGGING    STATE
         LOGGING CONSOLE          OFF
         LOGGING FILE             ON
         LOGGING MONITOR

DEFINE   LOGGING CONSOLE  NAME sink-name
         LOGGING FILE
         LOGGING MONITOR

DEFINE   KNOWN LOGGING    ALL EVENTS
         LOGGING CONSOLE  EVENTS evt-list
         LOGGING FILE     KNOWN EVENTS
         LOGGING MONITOR

DEFINE   KNOWN LOGGING    ALL EVENTS      CIRCUIT circuit-id
         LOGGING CONSOLE  EVENTS evt-list LINE line-id
         LOGGING FILE     KNOWN EVENTS    NODE node-id
         LOGGING MONITOR

DEFINE   KNOWN LOGGING    ALL EVENTS      KNOWN SINKS
         LOGGING CONSOLE  EVENTS evt-list SINK
         LOGGING FILE     KNOWN EVENTS          EXECUTOR
         LOGGING MONITOR                        NODE node-id
```

```
DEFINE   KNOWN LOGGING          ALL EVENTS          CIRCUIT circuit-id   KNOWN SINKS
         LOGGING CONSOLE        EVENTS evt-list     LINE line-id         SINK
         LOGGING FILE           KNOWN EVENTS        NODE node-id             EXECUTOR
         LOGGING MONITOR                                                     NODE node-id

DEFINE   KNOWN NODES            COUNTER TIMER seconds
         NODE node-id           RECEIVE ANSWER PASSWORD passwd
                                RECEIVE ORIGINATE PASSWORD passwd
                                TRANSMIT ANSWER PASSWORD passwd
                                TRANSMIT ORIGINATE PASSWORD passwd

DEFINE   NODE node-id           ALIAS name
                                COUNTER TIMER seconds
                                NAME node name
                                RECEIVE ANSWER PASSWORD passwd
                                RECEIVE ORGINATE PASSWORD passwd
                                TRANSMIT ANSWER PASSWORD passwd
                                TRANSMIT ORIGINATE PASSWORD passwd

DEFINE   OBJECT object-id       FILE file-spec
                                NAME object name
                                PARAMETERS p1, p2
                                TYPE object type

DEFINE   PERMANENT PARAMETER FILE

EXIT

LIST     KNOWN CIRCUITS         CHARACTERISTICS     TO file-spec
         CIRCUIT circuit-id     SUMMARY

LIST     EXECUTOR               CHARACTERISTICS     TO file-spec
                                SUMMARY

LIST     KNOWN LINES            CHARACTERISTICS     TO file-spec
         LINE line-id           SUMMARY

LIST     KNOWN LOGGING          CHARACTERISTICS     TO file-spec
         LOGGING CONSOLE        EVENTS
         LOGGING FILE           SUMMARY
         LOGGING MONITOR

LIST     KNOWN LOGGING          CHARACTERISTICS     KNOWN SINKS          TO file-spec
         LOGGING CONSOLE        EVENTS              SINK
         LOGGING FILE           SUMMARY                 EXECUTOR
         LOGGING MONITOR                                NODE node-id

LIST     KNOWN NODES            CHARACTERISTICS     TO file-spec
         NODE node-id           SUMMARY

LIST     KNOWN OBJECTS          CHARACTERISTICS         TO file-spec
         OBJECT object-id       SUMMARY
```

```
LOOP    EXECUTOR
        NODE node-id

LOOP    EXECUTOR        COUNT number
        NODE node-id    LENGTH number
                        WITH
                                MIXED
                                ONES
                                ZEROES

LOOP    EXECUTOR        COUNT number    LENGTH number    WITH
        NODE node-id                                             MIXED
                                                                 ONES
                                                                 ZEROES

LOOP    EXECUTOR        USER user-id    PASSWORD passwd    ACCOUNT acct
        NODE node-id

LOOP    EXECUTOR        USER user-id    PASSWORD passwd    ACCOUNT acct    COUNT number
        NODE node-id                                                       LENGTH number
                                                                           WITH
                                                                                MIXED
                                                                                ONES
                                                                                ZEROES

PURGE   KNOWN CIRCUITS  ALL
        CIRCUIT circuit-id  OWNER

PURGE   EXECUTOR        ALIAS
                        ALL
                        COUNTER TIMER
                        DEFAULT ACCOUNT
                        IDENTIFICATION
                        INCOMING TIMER
                        NAME
                        OUTGOING TIMER

PURGE   KNOWN LINES     ALL
        LINE line-id

PURGE   KNOWN LOGGING       ALL EVENTS
        LOGGING CONSOLE     EVENTS evt-list
        LOGGING FILE        KNOWN EVENTS
        LOGGING MONITOR     NAME

PURGE   KNOWN LOGGING       ALL EVENTS      CIRCUIT circuit-id
        LOGGING CONSOLE     EVENTS evt-list LINE line-id
        LOGGING FILE        KNOWN EVENTS    NODE node-id
        LOGGING MONITOR

PURGE   KNOWN LOGGING       ALL EVENTS      SINK
        LOGGING CONSOLE     EVENTS evt-list         EXECUTOR
        LOGGING FILE        KNOWN EVENTS            NODE node-id
        LOGGING MONITOR
```

```
PURGE   KNOWN LOGGING      ALL EVENTS          CIRCUIT circuit-id   SINK
        LOGGING CONSOLE    EVENTS evt-list     LINE line-id              EXECUTOR
        LOGGING FILE       KNOWN EVENTS        NODE node-id              NODE node-id
        LOGGING MONITOR

PURGE   KNOWN NODES        ALIAS
        NODE node-id       ALL
                           COUNTER TIMER
                           NAME

PURGE   KNOWN OBJECTS      ALL
        OBJECT object-id

SET     ALL LOGGING    STATE
                               OFF
                               ON

SET     KNOWN CIRCUITS         ALL
        CIRCUIT circuit-id     AUTORESTART
                                       OFF
                                       ON
                               COST cost
                               HELLO TIMER seconds
                               OWNER EXECUTOR
                               STATE
                                       OFF
                                       ON
                               TRIBUTARY trib-address

SET     EXECUTOR   ADDRESS number
                   ALIAS name
                   ALL
                   BUFFER SIZE number
                   COUNTER TIMER seconds
                   DATA TRANSMIT QUEUE MAXIMUM number
                   DEFAULT ACCOUNT user-id
                   DELAY FACTOR number
                   DELAY WEIGHT number
                   IDENTIFICATION string
                   INACTIVITY TIMER seconds
                   INCOMING TIMER seconds
                   INTERRUPT TRANSMIT QUEUE MAXIMUM number
                   MAXIMUM ADDRESS number
                   MAXIMUM BUFFERS number
                   MAXIMUM CIRCUITS number
                   MAXIMUM COST number
                   MAXIMUM HOPS number
                   MAXIMUM LINKS number
                   MAXIMUM VISITS number
                   NAME node name
                   OUTGOING TIMER seconds
```

RETRANSMIT FACTOR *number*
ROUTING TIMER *seconds*
STATE
OFF
ON
RESTRICTED
SHUT

SET EXECUTOR NODE *node-id*

SET EXECUTOR NODE *node-id* USER *user-id* PASSWORD *passwd* ACCOUNT *acct*

SET KNOWN LINES ALL
LINE *line-id* CONTROLLER
LOOPBACK
NORMAL
DEAD TIMER *milliseconds*
DELAY TIMER *milliseconds*
DUPLEX
FULL
HALF
PROTOCOL
DDCMP CONTROL
DDCMP DMC
DDCMP POINT
DDCMP TRIBUTARY
RECEIVE BUFFER *number*
RETRANSMIT TIMER *milliseconds*
VERIFICATION
ANSWER
OFF
ORIGINATE

SET KNOWN LOGGING STATE
LOGGING CONSOLE OFF
LOGGING FILE ON
LOGGING MONITOR

SET LOGGING CONSOLE NAME *sink-name*
LOGGING FILE
LOGGING MONITOR

SET KNOWN LOGGING ALL EVENTS
LOGGING CONSOLE EVENTS *evt-list*
LOGGING FILE KNOWN EVENTS
LOGGING MONITOR

SET KNOWN LOGGING ALL EVENTS CIRCUIT *circuit-id*
LOGGING CONSOLE EVENTS *evt-list* LINE *line-id*
LOGGING FILE KNOWN EVENTS NODE *node-id*
LOGGING MONITOR

```
SET   KNOWN LOGGING      ALL EVENTS       KNOWN SINKS
      LOGGING CONSOLE    EVENTS evt-list  SINK
      LOGGING FILE       KNOWN EVENTS           EXECUTOR
      LOGGING MONITOR                           NODE node-id

SET   KNOWN LOGGING      ALL EVENTS       CIRCUIT circuit-id   KNOWN SINKS
      LOGGING CONSOLE    EVENTS evt-list  LINE line-id         SINK
      LOGGING FILE       KNOWN EVENTS     NODE node-id                  EXECUTOR
      LOGGING MONITOR                                                   NODE node-id

SET   KNOWN NODES        ALL
      NODE node-id       COUNTER TIMER seconds
                         RECEIVE ANSWER PASSWORD passwd
                         RECEIVE ORIGINATE PASSWORD passwd
                         TRANSMIT ANSWER PASSWORD passwd
                         TRANSMIT ORIGINATE PASSWORD passwd

SET   NODE node-id       ALIAS name
                         CIRCUIT circuit-id
                         COUNTER TIMER seconds
                         NAME node name
                         RECEIVE ANSWER PASSWORD passwd
                         RECEIVE ORGINATE PASSWORD passwd
                         TRANSMIT ANSWER PASSWORD passwd
                         TRANSMIT ORIGINATE PASSWORD passwd

SET   OBJECT object-id   FILE file-spec
                         NAME object name
                         PARAMETERS p1, p2
                         TYPE object type

SET   SYSTEM

SHOW  ACTIVE CIRCUITS    CHARACTERISTICS  TO file-spec
      KNOWN CIRCUITS     COUNTERS
      CIRCUIT circuit-id STATUS
                         SUMMARY

SHOW  EXECUTOR           CHARACTERISTICS  TO file-spec
                         COUNTERS
                         STATUS
                         SUMMARY

SHOW  ACTIVE LINES       CHARACTERISTICS  TO file-spec
      KNOWN LINES        COUNTERS
      LINE line-id       STATUS
                         SUMMARY

SHOW  ACTIVE LINKS       STATUS           TO file-spec
                         SUMMARY
```

SHOW	ACTIVE NODES KNOWN NODES NODE *node-id*	CHARACTERISTICS COUNTERS STATUS SUMMARY	TO *file-spec*	
SHOW	KNOWN LOGGING LOGGING CONSOLE LOGGING FILE LOGGING MONITOR	CHARACTERISTICS EVENTS STATUS SUMMARY	TO *file-spec*	
SHOW	KNOWN LOGGING LOGGING CONSOLE LOGGING FILE LOGGING MONITOR	CHARACTERISTICS EVENTS STATUS SUMMARY	KNOWN SINKS SINK EXECUTOR NODE *node-id*	TO *file-spec*
SHOW	KNOWN OBJECTS OBJECT *object-id*	CHARACTERISTICS SUMMARY	TO *file-spec*	
SHOW	LOOP NODES	COUNTERS STATUS SUMMARY	TO *file-spec*	
ZERO	ACTIVE CIRCUITS KNOWN CIRCUITS CIRCUIT *circuit-id*	COUNTERS		
ZERO	ACTIVE LINES KNOW LINES LINE *line-id*	COUNTERS		
ZERO	ACTIVE LINES KNOW LINES NODE *node-id*	COUNTERS		
ZERO	EXECUTOR	COUNTERS		
TELL	*node-id*	*command*		
TELL	*node-id*	USER *user-id*	PASSWORD *passwd*	*ACCOUNT acct command*

Appendix B
Network Circuit, Line, and Node Counter Summary

This appendix describes the circuit, line, and node counters for DECnet/E entities. Where possible, the description for each counter includes the probable causes for this type of occurrence. In some cases, the counters correspond to network events. The events and event descriptions provide additional information about the specific occurrence. The individual counter descriptions state which counters correspond to events. See Appendix D for complete descriptions of these events.

NCP reports the contents of all circuit, line, and node counters in decimal. Counter content displays with an angle bracket (>) indicate that the counter has overflowed. For example:

```
NCP>SHOW LINE DMP-0 COUNTERS
Line Volatile Counters as of 17-Nov-81 20:48:18

Line = DMP-0

         >65534   Seconds Since Last Zeroed
             96   Data Errors Inbound, including:
                    NaKs Sent, Header Block Check error
              0   Remote Process Errors
            250   Local Process Errors, including:
                    NaKs Sent, Receiver Overrun
                    Transmit Underruns
                    NAKs Received, Header Format error


NCP>
```

B.1 Circuit Counters

NOTE

If error counters cannot be attributed to a particular circuit, they are considered line counters (see Section B.2).

Circuit counters and their descriptions are as follows:

Bytes received

This indicates the number of bytes in all packets received on a particular circuit.

Bytes sent

This indicates the number of bytes in all packets sent over a particular circuit.

Circuit down

This indicates the number of failures – operator or software induced – that cause the circuit to shut down. This may include any number of hardware-, software-, or operator-caused problems. This counter corresponds to events 4.7-9 (circuit down).

Data blocks received

This indicates the number of packets received on a particular circuit.

Data blocks sent

This indicates the number of packets sent over a particular circuit.

Data errors inbound

This indicates the number of times a packet with invalid data was received.

Data errors outbound

This indicates the number of times a packet with invalid data was sent.

Initialization failure

This indicates the number of times a circuit at the local node failed to initialize with the Transport module at a remote node. This may include any number of hardware-, software-, or operator-caused problems. This counter corresponds to 4.11-13 (initialization failure).

Local buffer errors

This indicates buffers are not available at the local station.

Local process errors

This indicates a general type of hardware/software error occurring at the local station. This type of error includes "bad packet format" that indicates the contents of the packet is invalid and "selection address error" that indicates, for example, that you polled tributary 5, but received a response from tributary 6.

Local reply timeouts

This indicates the number of times a message was sent and no acknowledgment was received by the adjacent node.

Originating packets sent

This indicates the number of data packets sent by the Transport layer from the NSP layer on the executor node.

Remote buffer errors

This indicates buffers are not available at the remote station.

Remote process errors

This indicates a general type of hardware/software error occurring at a remote station. These errors are the same as those listed for local process errors, however, they occur at a remote station.

Remote reply timeouts

This indicates the number of times the adjacent node did not receive an acknowledgment.

Seconds since last zeroed

This indicates the number of seconds that elapsed since the circuit counters were zeroed. This counter provides a time frame for other counter values. The software increments this counter every second and clears it when other counters are cleared.

Selection intervals elapsed

This indicates the number of times the tributary at the other end of a circuit is polled. This counter is applicable to multipoint control stations only.

Selection timeouts

This indicates the number of tributary polls that did not produce a response. This counter is applicable to multipoint control stations only.

Terminating congestion loss

This indicates the number of incoming packets that were discarded by Transport because of insufficient buffer space.

Terminating packets received

This indicates the number of data packets received by the Transport layer for the NSP layer on the executor node.

Transit congestion loss

This indicates the number of transit packets discarded because of insufficient buffer space. This counter is maintained only on routing nodes. If congestion loss increases, increase the MAXIMUM BUFFERS parameter for the executor node and/or the RECEIVE BUFFERS parameter for the line.

Transit packets received

This indicates the number of data packets received over the circuit to be routed through the executor node to another node. This counter is maintained only on routing nodes.

Transit packets sent

This indicates the number of data packets sent over the circuit to be routed through the executor to another node. This counter is maintained only on routing nodes.

B.2 Line Counters

Line counters and their descriptions are as follows:

Data errors inbound

This indicates the number of times a packet with invalid data was received.

Local process errors

This indicates a general type of hardware/software error occurring at the local station. This type of error includes "bad packet format" that indicates the contents of the packet is invalid and "selection address error" that indicates, for example, that you polled tributary 5 but received a response from tributary 6.

Remote process errors

This indicates a general type of hardware/software error occurring at a remote station. These errors are the same as those listed for local process errors, however, they occur at a remote station.

Seconds since last zeroed

This indicates the number of seconds that elapsed since the line counters were zeroed. This counter provides a time frame for other counter values. The software increments this counter every second and clears it when other counters are cleared.

B.3 Node Counters

The node counters and their descriptions are as follows:

Aged packet loss

This indicates the number of data packets discarded for visiting too many nodes. This usually occurs while the parameter files throughout the network are recovering from a disruption (for example, a line is going down). This counter is maintained only on routing nodes. This counter corresponds to event 4.0 (aged packet loss).

Bytes received

This indicates the number of user data bytes received from a remote node. This includes interrupt messages, but excludes connect, accept, reject, and disconnect messages.

Bytes sent

This indicates the number of user data bytes sent to a remote node.

Connects received

This indicates the number of logical link connection requests received by the executor node.

Connects sent

This indicates the number of logical link connection requests sent by the executor node.

Maximum logical links active

This indicates the largest number of logical links that have been active since DECnet/E software was started or since executor counters were zeroed.

Messages received

This indicates the number of NSP messages received from a remote node.

Messages sent

This indicates the number of NSP messages sent to a remote node.

Node out-of-range packet loss

This indicates the number of data packets discarded due to the destination node's address being greater than the maximum address defined for the executor node. This counter corresponds to event 4.2 (node out-of-range packet loss).

Node unreachable packet loss

This indicates the number of data packets lost because the destination node could not be accessed. This counter is maintained only on full routing nodes. This counter corresponds to event 4.1 (node unreachable packet loss).

Oversized packet loss

This indicates the number of received data packets that could not be forwarded because the block size of the data link to transmit them was too small. This counter is maintained only on routing nodes. This counter corresponds to event 4.3 (oversized packet loss).

Packet format error

This indicates the number of packet format errors that occur due to invalid packet control information. This counter corresponds to event 4.4 (packet format error).

Partial routing update loss

This indicates the number of received routing messages that were too long to process. Part of a routing update may be lost if it contains a reachable node with an address greater than the maximum address defined for the executor node. This counter is maintained only on full routing nodes. This counter corresponds to event 4.5 (partial routing update loss).

Received connect resource errors

This indicates the number of inbound connect messages for which the local node did not have sufficient resources. These errors may result from dynamic memory problems or too few logical link slots (for example, the MAXIMUM LINKS parameter is too small).

Response timeouts

This indicates the number of times there was no response to an NSP segment within the allotted timeout period. This implies that the executor node has to retransmit messages. This can be caused either by messages being discarded in the network or by a wide variance in the round trip delay to the node. This normally indicates an overload condition in the network.

Seconds since last zeroed

This indicates the number of seconds that elapsed since the node counters were zeroed. This counter provides a time frame for other counter values. The software increments this counter every second and clears it when other counters are cleared.

Verification reject

This indicates the number of received verification messages that were invalid. This counter corresponds to event 4.6 (verification reject).

Appendix C
Network Parameter and Counter Type Numbers

Every parameter or counter affected by the SET, SHOW, and CLEAR commands across the network is assigned an internal type number. Parameter and counter type numbers in the range of 2100 to 2299 are RSTS/E specific.

These type numbers appear in displays of information (from remote nodes) for which the local node is unable to display corresponding text. For example, if you were on node DALLAS and you set the executor to node BOSTON, and then you issued the SHOW LINE command, you might receive a display similar to the one below:

```
NCP>TELL BOSTON SHOW LINE DMC-1 CHARACTERISTICS
Line Volatile Characteristics as of 25-SEP-81 11:17:28
Line = DMC-1

State                    = On
Receive Buffers          = 4
Controller               = Normal
Duplex Mode              = Full Duplex
Protocol Type            = DDCMP DMC
Retransmit Timer         = 3000
Parameter #2111          = 0

NCP>
```

The parameter that is DECnet/E specific in this example is number 2111, the verification parameter. Table C-1 describes entity parameters that are DECnet/E system-specific. The *Network Management Functional Specification* describes all standard type numbers for all parameters and counters.

NOTE

Turning the network OFF and then ON does not reset circuit and line counters to zero. However, turning the network OFF and then ON resets node counters to zero.

Table C-1: DECnet/E Specific NICE Protocol Codes

Parameter Type Number	NICE Data Type	Description
Circuit parameters		
2110	C-1	Autorestart
		Autorestart Values 0 = On 1 = Off
Line parameters		
2111	C-1	Verification
		Line Verification Mode Values 0 = Off (No Verification) 1 = Use ORIGINATE Passwords 2 = Use ANSWER Passwords
Link parameters **RSTS-specific entity type = 5**		
2130	DU-2	Local Link Address (LLA)
2131	DU-2	Remote Link Address (RLA)
2132	DU-1	User Link Address (ULA)
2133	C-1	Link State
		Link State Values 0 = Reserved 1 = Connect Initiate Delivered 2 = Connect Initiate Sent 3 = Connect Initiate Received 4 = Connect Confirm Sent 5 = Run 6 = Disconnect Pending 7 = Disconnect Sent
2134	CM-2	Node Number and Name
2135	AI-6	Object Name
2136	DU-1	Object Job Number
2137	DU-1	Object RIB Number
2138	C-1	Local Flow Control Option
2139	C-1	Remote Flow Control Option
		Flow Control Values 0 = No Flow Control 1 = Segment Flow Control 2 = Message Flow Control
2140	DS-1	Local Data Request Count
2141	DS-1	Remote Data Request Count
2142	DS-1	Local Interrupt Request Count
2143	DS-1	Remote Interrupt Request Count

(continued on next page)

Table C–1 (Cont.): DECnet/E Specific NICE Protocol Codes

Parameter Type Number	NICE Data Type	Description
Node parameters		
2120	AI-8	Receive Password Originate
2121	AI-8	Receive Password Answer
2122	AI-8	Transmit Password Originate
2123	AI-8	Transmit Password Answer
2124	AI-6	Alias Name
2125	AI-16	Default Account
2126	DU-1	Data Transmit Queue Maximum
2127	DU-1	Interrupt Transmit Queue Maximum
2128	AI-28	Volatile Parameter File Name *file-spec*
Object parameters RSTS-specific entity type = 7		
2100	AI-32	File *file-spec*
2101	DU-2	Parameter 1 (P1)
2102	DU-2	Parameter 2 (P2)
2103	DU-1	Type
Node counters		
2200	16	Number of reachable nodes
2201	16	Highest number of reachable nodes

Table C-2: DECnet/E Specific Codes in Event Messages

Event	Parameter Type Number	NICE Data Type	Description
33.0	0	C-1	Access 0 = Local 1 = Remote
	1	C-1	Function 0 = <Reserved> 1 = OPEN/Read 2 = OPEN/Write 3 = <Reserved> 4 = DELETE 5 = <Reserved> 6 = DIRECTORY 7 = SUBMIT 8 = EXECUTE
	3	CM-1/2 DU-2 AI-6	Remote node node address node name (if present)

(continued on next page)

Table C–2 (Cont.): DECnet/E Specific Codes in Event Messages

Event	Parameter Type Number	NICE Data Type	Description
	4	CM-1/2/3/4	Remote process
		DU-1	object type
		DU-2	group code (if present)
		DU-2	user code (if present)
		AI-16	process name (if present)
	5	CM-1/2/3/4	Local process
		DU-1	object type
		DU-2	group code (if present)
		DU-2	user code (if present)
		AI-16	process name (if present)
	6	AI-39	User
	9	CM-1/2	File accessed
		DU-2	filesize
		AI-39	filename
34.0			Object spawned
	3	CM-1/2	source node
	4	CM-1/2/4	source process
	5	CM-1/2/4	destination process
	6	AI-39	user
	7	C-1	password
			0 (if password present)
	8	AI-39	account
34.1			Object spawn failed
	0	C-1	reason
			0 = I/O error on object data base
			1 = spawn directive failed
			2 = unknown object identification
	3	CM-1/2	source node
	4	CM-1/2/4	source process
	5	CM-1/2/4	destination process
	6	AI-39	user
	7	C-1	password
			0 (if password present)
	8	AI-39	account
36.0			Routing data base error
	0	OC-2	virtual address
	1	OC-2	MMU address

Appendix D
Object Type Codes

0	General Task, User Process
1	File Access (FAL/DAP-Version 1)
2	Unit Record Service (URDs)
3	Application Terminal Services (ATS)
4	Command Terminal Services (CTS)
5	RSX-11M Task Control-Version 1
6	Operator Services Interface
7	Node Resource Manager
8	IBM 3270-BSC Gateway
9	IBM 2780-BSC Gateway
10	IBM 3790-SDLC Gateway
11	TPS Application
12	RT-11 DIBOL Application
13	TOPS-20 Terminal Handler
14	TOPS-20 Remote Spooler
15	RSX-11M Task Control-Version 2
16	TLK (LSN on DECnet/E) Utility
17	File Access (FAL/DAP-Version 4 and later)
18	RSX-11S Remote Task Loader
19	Network Management Listener (NICE Process)
20	RSTS/E Media Transfer Program (NETCPY)
21	Reserved
22	Mail Listener (DECnet-based electronic mail system)
23	Host Terminal Handler (NPKDVR)
24	Concentrator Terminal Handler
25	Loopback Mirror
26	Event Receiver
27	VAX/VMS Personal Message Utility
28	File Transfer Spooler (FTS)

Appendix E
Event Class and Type Summary

Event classes are listed in Table 1 below.

Table E-1: Event Classes

Event Class	Description
0	Network Management layer
1	Applications layer
2	Session Control layer
3	Network Services layer
4	Transport layer
5	Data Link layer
6	Physical Link layer
7-31	Reserved for other common classes
32-63	RSTS system specific 33 Applications layer 34 Session Control layer 36 Transport layer
64-95	RSX system specific
96-127	TOPS-20 system specific
128-159	VMS system specific
160-479	Reserved for future use
480-511	Customer specific (DECnet/E supports only 480)

Table 2 lists the event types for each class. An entity related to an event indicates that the event can be filtered specific to that entity. Binary logging data is formatted under the same rules as the data in NICE data blocks. For definitions of parameters and counters, see the *DNA Network Management Functional Specification.*

Table E-2: Events

Class	Type	Entity	Standard Text	Event Parameters and Counters *
0	0	none	Event records lost	none
0	1	node	Automatic node counters	Node counters
0	2	line	Automatic line counters	Line counters
0	3	line	Automatic line service	Service, Status
0	4	cir	Circuit counters zeroed	Circuit counters
0	5	node	Node counters zeroed	Node counters
0	6	cir	Passive loopback	Operation
0	7	cir	Aborted service request	Reason
2	0	none	Local node state change	Reason Old state New state
2	1	none	Access control reject	Source node Source process Destination process User Password Account
3	0	none	Invalid message	Message
3	1	none	Invalid flow control	Message Current flow control
3	2	node	Data base reused	NSP node counters
4	0	none	Aged packet loss	Packet header
4	1	cir	Node unreachable packet loss	Packet header
4	2	cir	Node out-of-range packet loss	Packet header
4	3	cir	Oversized packet loss	Packet header
4	4	cir	Packet format header	Packet beginning
4	5	cir	Partial routing update loss	Packet header Highest address
4	6	cir	Verification reject	Node
4	7	cir	Circuit down, circuit fault	Reason
4	8	cir	Circuit down, software fault	Reason Packet header
4	9	cir	Circuit down, operator fault	Reason Packet header Expected node
4	10	cir	Circuit up	Node
4	11	cir	Initialization failure, line fault	Reason
4	12	cir	Initialization failure, software fault	Reason Packet header
4	13	cir	Initialization failure, operator fault	Reason Packet header Received version
4	14	node	Node reachability change	Status
5	0	cir	Locally initiated state change	Old state New state
5	1	cir	Remotely initiated state change	Old state New state
5	2	cir	Protocol restart received in maintenance mode	none

(continued on next page)

Table E–2 (Cont.): Events

Class	Type	Entity	Standard Text	Event Parameters and Counters *
5	3	cir	Send error threshold	Circuit counters
5	4	cir	Receive error threshold	Circuit counters
5	5	cir	Select error threshold	Circuit counters
5	6	cir	Block header format error	Header (optional)
5	7	cir	Selection address error	Selected tributary Received tributary Previous tributary
5	8	line	Streaming tributary	Tributary status Recieved tributary
5	9	line	Local buffer too small	Block length Buffer length
6	0	line	Data set ready transition	New state
6	1	line	Ring indicator transition	New state
6	2	line	Unexpected carrier transition	New state
6	3	line	Memory access error	Device register
6	4	line	Communications interface error	Device register
6	5	line	Performance error	Device register
33	0	none	File access	Access Function Remote node Remote process Local process User File accessed
34	0	none	Object spawned	Source node Source process Destination process User Password Account
34	1	none	Object spawned failed	Reason Source node Source process Destination process User Password Account
36	0	none	Routing data base error	Error virtual address Error MMU address

* Counters are defined in Appendix A of the *DNA Network Management Functional Specification.*

Appendix F
Event Message File Formats

The purpose of this appendix is to acquaint you with the structure of the Event File. During the logging of events, the Event Processor takes raw events, as queued to it by the various layers of the DECnet software, processes and filters them, and passes them to the appropriate logging sinks. If the receiving sink is the Event File, the events are recorded in the file in machine-readable form. To examine the file, you ordinarily run the Event Logger, supplying the name of the file as input (as described in Section 2.7). The Event Logger reads each event record and converts it to an output form identical to that used for logging events to the console.

There may be circumstances, however, under which you might want to write a program to open and read the logging file apart from the Event Logger – perhaps to run a statistical analysis on certain types or classes of events. The Event Logger can be directed to send events to a user-written event monitor, using the RSTS/E local message send function. The format of such messages is exactly the same as that of the *data* in the records described below. (The messages are not prefixed with the two-byte length field.)

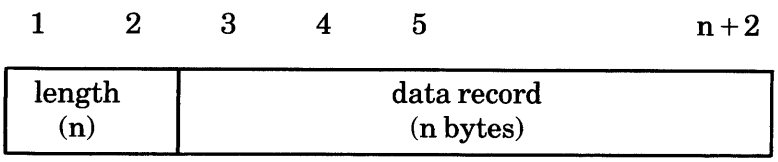
The event logging file is a sequential RMS file with a variable-length record format. That is, records can be any length, up to a maximum of 200 bytes, as indicated in the file attributes per NCP specification.

The following sample of output from PIP shows the RMS attributes of the file. (See the *RSTS/E User's Guide* for instructions on how to run PIP.)

Output is taken from a directory listing: PIP $EVTLOG.LOG/S.

```
SY:[1,2]
 Name .Ext  Size    Prot Access      Date        Time      Clu     RTS  Pos
EVTLOG.LOG  410     <60> 22-Jun-81   17-Jun-81   05:00PM   4    ...RSX 6621
 RF:VAR=200 FO:SEQ USED:410:312    RECSI:118
```

For each record, RMS-11 maintains a record length field specifying the number of data bytes in the record. The length field contains a two-byte binary count and is placed in front of the data record. The two bytes for the field length are not included in the count. Note that the field length is always word aligned.



Each event record is a binary record that conforms to the Network Information and Control Exchange (NICE) protocol, as dictated by the DIGITAL Network Architecture (see *DECnet DIGITAL Network Architecture, Network Management Functional Specification, Version 3.0.0*).

Each record has a header containing a set of fixed and variable length fields. The header has the following format:

1 byte — Function code: A binary 1, indicating that this is an event logging message.

1 byte — Sinks flags: A bit map indicating which sinks (on the local node) are to receive a copy of this event, one bit per sink. The bits are assigned as follows:

Bit	Sink
0	Console
1	File
2	Monitor

2 bytes — Event code: A bit map identifying the specific event. The bits are assigned as follows:

Bits	Meaning
0-4	Event type
6-14	Event class

6 bytes — Event time: The source node date and time of the event processing. This field consists of three subfields:

- 2 bytes — Julian half day: Number of half days since 1 January 1977 and before 9 November 2021. Range: 0-32767 binary.
- 2 bytes — Second: Second within the current half day. Range: 0-43199 binary.
- 2 bytes — Millisecond: Millisecond within the current second. Range: 0-999 binary.

3-9 bytes — Source node: The source node of this event. This field consists of two subfields:

- 2 bytes — Node address: The binary address of the node.

1-7 bytes — Node name: The ASCII node name of the node (if any) in "image" format. That is, the first byte contains the binary number of ASCII characters that follow (zero if none).

1-18 bytes — Event entity: The entity involved in the event, if applicable. This field contains two subfields:

1 byte — Entity type: A code indicating the type of entity. The following types are defined:

Value	Entity Type
−1	none
0	Node
1	Line
3	Circuit

1-17 bytes — Entity ID: The identifier of the entity involved. The length and form depends on the entity type:

Node ID	3 to 9 bytes, in Source Node format (above)
Line ID	1 to 17 bytes, in "image" format
Circuit ID	1 to 17 bytes, in "image" format

Following this header, the event record contains event-specific data. This consists of zero or more data entries, formatted under the same rules as the data in NICE data blocks. See the *DECnet DIGITAL Network Architecture, Network Management Functional Specification, Version 3.0.0* for specific details. Data block format specifications are found in Section 6.13 and event parameter definitions are found in Section 7.11-12.

The following example shows an event record in octal:

```
000041  003401  000416  006271  043212  000372  000214  040406  045522
000000  000000  000220  053006  054101  000064  003400  100400  000000
```

The ASCII output for that record, generated by the event logger, contains the following information:

Output is taken from a RUN $EVTLOG :

```
Event type 4.14, Node reachability change
Occurred 17-Jun-81 17:00:58.2 on node 140 (ARK)
Node address 144 (VAX4)
Reachable
```

Appendix G
NCP Error Message Summary

NCP generates standard error messages for a variety of error conditions attributable to both network and system conditions. These messages are listed below. A brief description giving the reason for the error and appropriate recovery procedures (if possible) follows each message.

There are two types of error messages:

1. Local NCP messages
2. Local or remote NML messages

Local NCP error messages include syntax errors, command I/O errors, and errors in network communication with a remote NML.

NML error messages include network management program errors, resource errors, and file I/O errors. There are three parts to the NML error message.

1. NML base response code
2. Error detail
3. Error text

The RSTS/E standard for error messages defines the following prefixes:

% The percent (%) prefix represents a warning condition that indicates the operation might complete but will likely produce a nonfatal error. For batch processing, the controlled job is terminated if the /ERROR:NONE option is selected.

? The query (?) prefix represents a fatal condition that indicates the operation cannot complete or will complete with invalid results. For batch processing, the controlled job is terminated if the /ERROR:NONE or /ERROR:WARNING options are selected.

Messages that do not contain the (%) or (?) signs are informational messages that do not imply a RSTS-level error condition has occurred and do not terminate batch processing.

NOTE

Some error messages listed below are followed by another message that contains the RSTS/E error code and the RSTS/E error message associated with the cause of the error. This information should be included when submitting Software Performance Reports (SPRs) for NCP.

Local NCP Error Messages

%Can't CLEAR SYSTEM – system not SET

Description: A CLEAR SYSTEM command was entered when the system was not previously set with the SET SYSTEM command.

Action: You must issue the SET SYSTEM command before you can clear the system.

?Can't SET SYSTEM – permanent parameter file owned by another user

Description: Upon entering the SET SYSTEM command, another user was found to have modification rights to the permanent parameter file.

Action: Retry the SET SYSTEM command after all other users finish running NCP.

?Can't SET SYSTEM – system already SET

Description: A SET SYSTEM command was entered while the system was already set.

Action: You should turn off NSP and execute a CLEAR SYSTEM command if it is necessary to reset the system.

?Can't SET SYSTEM – volatile parameter file owned by another user

Description: Upon entering the SET SYSTEM command, another user was found to have modification rights to the volatile parameter file.

Action: Retry the SET SYSTEM command after all other users finish running NCP.

?Error during permanent parameter file populate

Description: An error occurred during initialization of the permanent parameter file from the DEFINE PERMANENT PARAMETER FILE command.

Action: Consult your Software Specialist or submit a Software Performance Report (SPR).

?Error during volatile parameter file access in SET EXECUTOR STATE

Description: An error was detected when NCP attempted to access the volatile parameter file with the SET EXECUTOR STATE command.

Action: Consult your Software Specialist or submit an SPR.

?Error during volatile parameter file open in SET EXECUTOR STATE

Description: An error was detected when NCP attempted to open the volatile parameter file with the SET EXECUTOR STATE command.

Action: Consult your Software Specialist or submit an SPR.

?Fatal parameter file error during line auto-start function

Description: A fatal error was detected when NCP attempted to access the permanent parameter file with the SET EXECUTOR STATE ON command.

Action: Consult your Software Specialist or submit an SPR.

?Invalid command — \\(text of invalid command)

Description: NCP did not recognize all or part of the command. The text that it could not interpret is reprinted and enclosed with two backslashes (\\).

Action: You should correct and retype the command.

?NCP – Cannot find executor parameters

Description: A consistency error was detected when NCP tried to read the executor portion of either the permanent or volatile parameter files.

Action: Submit an SPR.

NCP – Sorry, you already have a remote executor

Description: A TELL command was entered while a remote node was acting as the executor node, through the SET EXECUTOR NODE *nnnnn* command.

Action: You cannot tell a remote executor to TELL another remote executor to perform a function for you; you must issue a SET EXECUTOR NODE *nnnnn* command to establish the correct node as the remote executor.

?Permanent or volatile parameter file OPEN failed

Description: An unexpected error was detected when NCP attempted to open one of its parameter files.

Action: The RSTS/E reason code for the failure should accompany the above message. If you are unable to correct the error, consult your Software Specialist or submit an SPR.

%Permanent parameter file already exists – please delete it

Description: You entered a DEFINE PERMANENT PARAMETER FILE command when a permanent parameter file already existed.

Action: As a safeguard against accidental initialization, you must delete the permanent parameter file ($NETPRM.SYS) with the PIP utility (or DCL DELETE command) before entering a DEFINE PERMANENT PARAMETER FILE command.

?Permanent parameter file create failed

Description: NCP could not create the permanent parameter file on the correct disk. The permanent parameter file requires a minimum of 120 (10) blocks.

Action: If the problem is not lack of available disk space, consult your Software Specialist or submit an SPR. Otherwise, you must free up enough space on the system disk to allow creation of the permanent parameter file.

?Permanent parameter file not found – please CREATE it

Description: An operation that required use of the permanent parameter file could not find the file ($NETPRM.SYS).

Action: Use the DEFINE PERMANENT PARAMETER FILE command to create the permanent parameter file.

?Unexpected error during CREATE of volatile parameter file –

Description: An error was detected when NCP tried to create the volatile parameter file.

Action: The RSTS/E reason code for the failure should accompany the above message. If you are unable to correct the error, consult your Software Specialist or submit an SPR.

?Unexpected error during permanent parameter file open –

Description: An error was detected when NCP tried to open the permanent parameter file.

Action: The RSTS/E reason code for the failure should accompany the above message. If you are unable to correct the error, consult your Software Specialist or submit an SPR.

?Unexpected error during permanent parameter file read –

Description: An error was detected when NCP tried to read data from the permanent parameter file.

Action: The RSTS/E reason code for the failure should accompany the above message. If you are unable to correct the error, consult your Software Specialist or submit an SPR.

?Unexpected error during volatile parameter file close –

Description: An error was detected when NCP tried to close the volatile parameter file.

Action: The RSTS/E reason code for the failure should accompany the above message. If you are unable to correct the error, consult your Software Specialist or submit an SPR.

?Unexpected error during volatile parameter file install –

Description: An error was detected when NCP tried to install the volatile parameter file to NSP and Transport.

Action: Submit an SPR.

?Unexpected error during volatile parameter file name lookup

Description: An error was detected when NCP tried to find the name of the volatile parameter file.

Action: Submit an SPR.

?Unexpected error during volatile parameter file open –

Description: An error was detected when NCP tried to open the volatile parameter file.

Action: Submit an SPR.

?Unexpected error during volatile parameter file write –

Description: An error was detected when NCP tried to write to the volatile parameter file.

Action: Submit an SPR.

?Unexpected error during volatile parameter SYS-file lookup

Description: An error was detected when NCP tried to determine the name and location of the volatile parameter file from RSTS/E.

Action: Submit an SPR.

?Unexpected error during volatile parameter SYS-file remove

Description: An error was detected when NCP tried to separate the volatile parameter file from RSTS/E.

Action: Submit an SPR.

Local or Remote NML Error Messages

The NML base response code prints ASCII text if the error is compatible with DECnet/E V2.0. If a message is not compatible with this version of DECnet, the NML base response code prints the error number. You should then refer to the documentation of the system you are using for information about the cause of the error.

The Error detail depends on the NML base response code. If the detail is compatible with this version of DECnet, ASCII text that describes the cause of error is printed. Otherwise, you should refer to the documentation of the system you are using for information about the cause of the error.

The Error text further explains the cause of the error and is system-specific. On RSTS, you receive a two part message that contains the RSTS error text for system errors and the NML function that was attempted.

Example:

"File Open Error"	(NML base response)
"Volatile data base"	(Error detail)
"Too many open files on unit"	(Error text)

Refer to the *RSTS/E System User's Guide* for a complete list of system error messages and their descriptions.

Already have maximum allowed number of circuits on

The upper limit for maximum circuits has been exceeded. Either turn one circuit off, or turn the executor state off, reset the maximum circuits parameter to a higher number, and turn the executor state on again.

Bad loopback response

A loopback message did not match what was expected for either content or length.

Component in wrong state

The executor node determined that the component affected by the current command was in an unacceptable state. For example, a node name to be used for a loop node name was already assigned a node address.

Error text may include the following:

Executor node in wrong state
Illegal modification for active lines and circuits

The executor is off, and the system has not been set

A SHOW command has been issued before the SET SYSTEM command has been issued. You cannot display parameters in the volatile parameter file until the SET SYSTEM command has been issued.

File I/O error

The executor node detected an error when it accessed an auxiliary file.

Error text may include the following:

Error reading parameter file
Error writing parameter file

File open error

The executor node found an error when it tried to open one of the parameter files.

Error text may include the following:

Error finding volatile parameter file

Hardware failure

The hardware associated with the request could not perform the operation requested.

Incompatible management version

The capability requested cannot be performed because of differences between NCP and the remote executor node.

Invalid file contents

The executor node detected invalid data in one of its files.

Invalid identification

NCP sent an invalid message to the local or a remote executor.

Invalid message format

NCP sent an invalid message to the local or a remote executor.

Invalid parameter grouping

A request to change multiple parameters contained some parameters that cannot be changed with others.

Invalid parameter value

NCP sent an invalid message to either the local executor or a remote executor.

Line communication error

The remote executor node detected an error on the communications line. Correct the problem and try again.

Line protocol error

NCP sent an invalid message to a remote executor node.

NOTE

It is important that you consult your Software Specialist or submit an SPR (Software Performance Report) that provides all error text and error detail for any Network Management Program errors that occur on your system. A UTILTY SNAP dump would also be helpful.

Management program error

The executor node detected an unexpected error.

Error text may include the following:

Error finding executor state
Volatile parameter file corrupted, network hung
Error getting network table pointers
Error getting Transport output line vector
Error getting Transport routing message
Error getting Transport circuit pointer vector
Error getting network device DDB
Error initializing network device DDB pointers
Error getting network device name
Error getting device counters
Error deassigning circuit
Error changing multipoint parameter
Unable to find executor node block
Error finding executor link count
Error finding Transport counters
Error initializing node block search
Error finding node block
Error finding node counter block
Error initializing receiver-id block search
Error finding receiver-id block
Error finding logical link block
Dynamic and file node data inconsistent
Error clearing loop node
Error defining loop node
Error finding node name dynamically
Error getting node counters

Mirror connect failed

The executor node could not connect to the Network Management Loopback Mirror when it executed a loop node command.

Mirror link disconnected

The other node to which a loop node command is active prematurely terminated the link.

No room for new entry

The executor node has insufficient table space to store the new entry.

Only circuit 0 can be set on when not in DDCMP control mode

Tributary stations and point-to-point stations can set only circuit 0. For example, DMP-0.0 is valid. DMP-0.*n* ($n > 0$) is not valid unless the station is in multipoint control mode.

Operation failure

The remote executor node could not perform the requested operation.

Oversized management command message

NCP sent an invalid message to the executor node.

Parameter missing

NCP sent an invalid message to the local or a remote executor node.

Parameter not applicable

The executor node determined that some parameter of the current command is not applicable to the specified entity. For example, an attempt to set a controller to loopback mode when that controller does not support loopback mode.

Parameter value too long

NCP sent an invalid message to the local or a remote executor node.

Privilege violation

You do not have sufficient privileges on the executor node to execute the specified operation.

Resource error

The remote executor node was unable to obtain some necessary resource.

Error text may include the following:

Error Assigning Circuit

System-specific management function not supported

A command was sent to the remote executor node that was recognized by NCP but not supported by the remote Network Management Listener.

Unrecognized component

NCP has sent an invalid message to the local or remote executor node.

Unrecognized function or option

The current executor node does not recognize the specified function or option.

Unrecognized parameter type

NCP has sent an invalid message to the local or a remote executor.

Volatile file-spec must be of form: *xx*:[0,1]*xxxxxx*.SYS

The volatile parameter must be specified in the form *xx*:[0,1]*xxxxxx*.SYS.

Wrong name in file for line; purge entry and correct

A device was not correctly specified in the permanent parameter file. For example, the command DEFINE CIRCUIT DMR-0 OWNER EXECUTOR was issued when the device was a DMC.

Glossary

Active Circuits

Active circuits are all circuits in the ON or SERVICE state.

Access Control

Access control information identifies you to a program on a remote node.

Active Lines

Active lines are all lines that are in the ON or SERVICE state.

Active Nodes

Active nodes are all nodes perceived as reachable by the executor node.

Adjacent Node

An adjacent node is a node removed from the executor by a single circuit (one hop away).

Characteristics

Characteristics are parameters that are static values in memory or in the parameter file. Characteristics can be SET or DEFINED. Characteristics are usually operator controllable.

Circuit

A circuit is a logical communications path between two adjacent nodes.

Controller

The controller is the peripheral hardware that manages communications over a line. The controller name is part of a line identification.

Control Station

A control station is a multipoint circuit that controls the multipoint line. A control station is equivalent to the master.

Cost

Cost is an arbitrary integer value you assign to a circuit between two adjacent nodes. Each circuit has a separate cost. Packets are routed on paths with the least cost. Nodes on either end of a circuit can assign different costs to the same circuit.

Counters

Counters are error and performance statistics for individual circuits, links, lines, and nodes.

DDCMP

Digital Data Communications Message Protocol (DDCMP) is the protocol for DECnet that is used to control data link level communication.

Display Type

The display type parameter used in the SHOW command controls the type of information to be displayed. Each command qualifier parameter and counter is associated with one or more display types (for example, CHARACTERISTICS, COUNTERS, EVENTS, or SUMMARY).

Entity Identifier

An entity identifier is a keyword that is part of a command (for example, a LINE or NODE in the case of SET LINE or SET NODE). Each entity identifier has several parameters with options. There are also plural entity identifiers (for example, KNOWN LINES or ACTIVE LINES).

Event Logger

The Network Management entity that routes event data to logging sinks such as a console or file.

Events

Events are occurrences that are logged by DECnet for eventual recording by Network Management.

Executor Node

The executor node is the node that executes the command.

Hop

A hop is the distance from one node to a neighbor (adjacent node).

Known Circuits

Known circuits are all circuits addressable by network management functions on the executor node. All known circuits are not necessarily usable at the same time.

Known Lines

Known lines include all lines addressable by network management functions on the executor node. All known lines are not necessarily usable at the same time.

Known Logging

Known logging includes all logging sink types addressable by network management functions.

Known Nodes

Known nodes are all nodes that have a name in the parameter file, are reachable, or both.

Line

A line is a physical path.

Line Identification

The line identification refers to the device, controller, and unit associated with a line.

Local Node

The local node is the one to which the terminal you use is physically connected.

Logging

Logging involves the recording of network events at a terminal, in a file, or by a user written program.

Logical Link

A logical link is a virtual data path between two programs.

Loop Node

A loop node is a node name associated with a circuit for loop testing purposes. The SET NODE CIRCUIT *circuit-id* command sets the loop node name.

Multipoint

A multipoint configuration is one where one physical line allows multiple logical circuits by connecting a master (control station) with slaves (tributaries).

Node

A node is a computer system that supports DECnet software within a network.

Node Address

A node address is a number that uniquely identifies a node within the network.

Node Identification

Node identification is a node name or a node address.

Node Level Loopback

The node level loopback is used to test a logical link by sending repeated messages directly to the Session Control, Network Services Protocol, and Transport layers within one node or from one node to another and back again.

Node Name

A node name is 1 to 6 alphanumeric characters that can be associated with a node address in a strict one to one mapping. In each node, no name may be used more than once. The node name must begin with an alphabetic character.

Object

An object is a network program that is automatically started upon request over a logical link.

OFF State

The OFF state, in the case of a node, is one in which the node no longer processes network traffic. In the case of a circuit or line, the OFF state is one in which the circuit or line is unavailable for any kind of traffic. In the case of logging, the OFF state is one in which an event sink is not operating, and all events for it are discarded.

ON State

The ON state, in the case of a node, is one which allows normal network operation. In the case of a circuit or line, the ON state allows normal circuit or line use. In the case of logging, the ON state allows events to be logged.

Network Services Protocol

The Network Services Protocol (NSP) is the part of DECnet that manages the creation, use, and deletion of logical links.

Reachable Node

A reachable node is a node to which the executor node's Transport module has identified a usable communications path.

Remote Executor

A remote executor is a remote node on which the NML program is executing an NCP command as a result of the NCP SET EXECUTOR NODE command or TELL prefix. This enables an operator to execute NCP commands at remote nodes.

Remote Node

A remote node is any node other than the local node.

Routing

Routing is the part of DECnet that determines the existence of a path to a destination node, decides on the next circuit in the path, and forwards a packet.

SHUT State

The SHUT state is a node state where existing logical links are undisturbed, but new ones are prevented. When you execute the NCP SET EXECUTOR STATE SHUT command, the node proceeds to a SHUT state and then to an OFF state when all logical links have been disconnected.

Station

A station is a physical termination on a circuit.

STATUS

STATUS refers to dynamic information about entities, for example, their STATE. Unlike CHARACTERISTICS, STATUS can change without explicit operator intervention.

Substate

A substate is an internal circuit STATE that is displayed with the circuit STATE.

SUMMARY

SUMMARY information is a display type that provides a subset of all available information.

Transport

Transport is the part of DECnet that controls routing, congestion, and packet lifetime.

Tributary

A tributary is a physical termination other than a control station on a multipoint line.

Unit

A unit is part of a line identification. A unit together with the controller forms a station.

Index

A

B

C

D

E

F

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

READER'S COMMENTS

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

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

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

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

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

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code_____
or
Country

—Do Not Tear - Fold Here and Tape —

digital

No Postage Necessary if Mailed in the United States

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

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE DOCUMENTATION
1925 ANDOVER STREET TW/E07
TEWKSBURY, MASSACHUSETTS 01876

Do Not Tear - Fold Here and Tape

Cut Along Dotted Line