DECnet–ULTRIX

digital

NCP Command Reference

Order Number: AA–PB62A–TE

DECnet–ULTRIX

NCP Command Reference

May 1990

This manual describes the Network Control Program (ncp) commands you use to define, monitor, and test your network.

Supersession/Update Information:	This is a new manual.
Operating System and Version:	ULTRIX V4.0
Software Version:	DECnet–ULTRIX V4.0

digital™

Order Number: AA–PB62A–TE

AA–PB62A–TE
May 1990

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 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.

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227–7013.

The following are trademarks of Digital Equipment Corporation:

DEC	PDP	VAX
DECnet	ULTRIX	VMS
DECUS	UNIBUS	digital™

UNIX is a registered trademark of AT&T in the USA and other countries.

This manual was produced by Networks and Communications Publications.

Contents

Preface

This manual explains how to use the Network Control Program (**ncp**) commands to manage a Phase IV DECnet–ULTRIX node within the DECnet environment.

Manual Objectives

This manual describes the **ncp** commands that you can use to configure, monitor, and test your network. Additional reference information summarizes the DECnet–ULTRIX network objects, event logging, network counters, and Ethernet addressing information.

Intended Audience

This manual is for anyone responsible for configuring, maintaining, and managing the network. The manual refers to all such people as the network manager.

Structure of This Manual

This manual is divided as follows:

Chapter 1	Gives a detailed explanation of how to invoke the Network Control Program and gives guidelines for using **ncp** commands.
Chapter 2	Describes each **ncp** command, including its function, syntax, parameters, and gives an example.
Chapter 3	Lists and explains all error messages that can occur from the execution of an **ncp** command.
Chapter 4	Lists and explains the network management event-logging messages.
Chapter 5	Lists and explains all network counters maintained by the DECnet–ULTRIX software.
Appendix A	Summarizes all **ncp** commands and their parameters.
Appendix B	Defines all DECnet–ULTRIX objects by name, number, file, and function.
Appendix C	Describes physical and multicast addressing for nodes on Ethernet lines.

Related Documents

For more information about DECnet-ULTRIX software, see the following manuals:

- *DECnet–ULTRIX Release Notes*

  Contains information and updates not included in the DECnet–ULTRIX documentation set.

- *DECnet–ULTRIX Installation*

  Contains step-by-step procedures for installing your DECnet–ULTRIX software and testing your node's operation in the network.

- *DECnet–ULTRIX Use*

  Describes the DECnet–ULTRIX user commands and explains how to use them to perform file transfer and other user tasks.

- *DECnet–ULTRIX Network Management*

  Introduces the network manager to DECnet databases and components and describes how to use the Network Control Program (**ncp**) to configure, monitor, and test these components.

- *DECnet–ULTRIX Programming*

  Describes the DECnet–ULTRIX system calls and subroutines. Also provides information about application programming within the DECnet environment and contains supplemental information for programming the DECnet–ULTRIX socket interface.

- *DECnet–ULTRIX DECnet-Internet Gateway Use and Management*

  Describes the DECnet-Internet Gateway and explains how to use, manage, and install it.

To obtain a detailed description of the Digital Network Architecture, refer to the following document:

- *DECnet Digital Network Architecture (Phase IV), General Description*

Acronyms

The following acronyms are used in this manual:

DNA	Digital Network Architecture
dtr	DECnet Test Receiver
dts	DECnet Test Sender
ECL	End Communication layer
evl	Event Logger
fal	File Access Listener
mir	Loopback Mirror
ncp	Network Control Program
nml	Network Management Listener
NSP	Network Services Protocol

Conventions Used in This Manual

Convention	Meaning
`Example`	Examples appear in this special type.
Red type	Red type in examples indicates text that a user enters.
special	All commands and parameters appear in **special type**.
lowercase	If a command appears in lowercase type in a command format or in an example, you must enter it in lowercase.
italic	Italic type in command formats and system displays indicates a variable, for which either you or the system must supply a value.
{ }	Braces indicate that you must specify one of the enclosed options, but no more than one. Do not type the braces when you enter the command.
[ ]	Square brackets indicate that you can use one, and only one, of the enclosed options. Do not type the brackets when you enter the command.
( )	Parentheses enclose a set of options that you must specify together or not at all.
Vertical list of options	A vertical list of options not enclosed within braces, brackets, or parentheses indicates that you can specify any number of options, or none at all.
key	This is a symbol for a keyboard key. CTRL/*key* represents a CTRL key sequence, where you press the CTRL key at the same time as the specified key.
%	This is the default user prompt in multiuser mode.
#	This is the default superuser prompt.

All Ethernet addresses are hexadecimal; all other numbers are decimal unless otherwise noted.

Chapter 1

Understanding the Network Control Program

This chapter tells you how to use the Network Control Program (**ncp**) on DECnet–ULTRIX nodes in the following ways:

- Invoke and exit the Network Control Program.
- Execute **ncp** commands.
- Issue **ncp** commands from your terminal for execution at a remote node.
- Maintain network security with superuser privileges.
- Down-line load a remote node using **load** and **trigger** commands.

1.1 Getting Started with ncp

The Network Control Program lets you issue **ncp** commands from a terminal or from a shell script. You can execute most **ncp** commands either locally or remotely.

1.1.1 Invoking ncp

You can invoke **ncp** in three ways:

- Enter **ncp** at the prompt.

  ```
  % ncp RET
  ```

  The program then prompts you as follows:

  ```
  ncp>
  ```

  Enter your **ncp** command after the prompt and press RET.

- Enter an entire **ncp** command line, for example:

  ```
  % ncp show known circuits counters RET
  ```

  where **show known circuits counters** is a valid **ncp** command. After the command executes, you return to the shell.

- Enter **ncp** with a shell script, for example:

  ```
  % ncp <scripta
  ```

  where *scripta* is the name of a shell script that contains a sequence of **ncp** commands. Your shell script can use the exit status returned by **ncp** commands.

The following example shows a sample shell script:

```
ncp sho line una-0
if ( $status != 0 ) then
        echo ""
        echo "This ncp command failed."
        echo ""
endif
```

This sample shell script uses the exit status from an **ncp** command to determine whether or not to echo a message. If the **ncp show line** command fails, the shell script echoes the message.

NOTE

You can insert comment lines in an **ncp** shell script by prefacing each comment line with a pound sign (#).

1.1.2 Exiting ncp

To exit **ncp**, use either the **exit** command, the **quit** command, or CTRL/D at the **ncp** prompt.

1.2 Executing ncp Commands

The following sections explain the **ncp** command syntax and procedures for executing them.

1.2.1 Command Syntax

Most commands consist of three parts: the command verb, a component on which the command operates, and one or more parameters that further qualify the action to be taken on the component. You enter a command at the **ncp** prompt in the following order:

ncp *command-verb component parameter*

EXAMPLE:

This example shows a **list** command.

```
list line una-0 characteristics RET
```

For each command, you must supply one verb and one component option. The number of parameters that you can supply varies with each command.

Some commands have a list of optional parameters, any number of which you can specify. For example, the **list line** command lets you select one or more of the following parameters:

list { **line** *line-id* | **known lines** } [**characteristics** | **counter** | **status** | **summary**]

Some commands have the **all** parameter. When you specify **all**, you cannot specify any other parameters for that command. If you do not specify **all**, you can use any number of the remaining parameters. For example, you can use **all** or you can specify any other **clear node** command parameters:

clear { **node** *node-id* | **known nodes** } [**all** | **diagnostic file** | **dump file** | **hardware address** | **host** | **load file** | **name** | **secondary loader** | **service circuit** | **service password** | **tertiary loader**]

EXAMPLE:

This example shows how you can use **all** to specify all **clear node** parameters for an object.

```
ncp>clear node NAVAHO all RET
```

Some commands have a bracketed list of optional parameters of which you can specify only one option. For example, the **show object** command lets you select only one parameter for a specified object or for all known objects.

show { **objects** *object-name* | **known objects** } [**characteristics** | **summary**]

1.2.2 Typing Command Lines

Enter the command keywords separated by spaces. You can abbreviate any keyword to the shortest number of unique characters that **ncp** accepts. For example, the following versions of the same command are equally valid:

```
ncp>show logging console RET

ncp>sho log con RET

ncp>sh lo c RET
```

1.2.3 Using the Help Facility

Enter **help** at the **ncp** prompt for assistance in selecting network management commands and parameter options. The **ncp** utility returns a list of subjects for which help information is available.

EXAMPLE:

This example shows how the **help** command displays available information.

```
ncp>help RET

Help available for:
clear           command         define          exit
help            list            load            loop
parameters      prompting       purge           set
show            tell            trigger         zero

Topic?
```

For additional information, enter one of the command verbs at the **Topic?** prompt.

1.2.4 Command Prompting

Command prompting provides on-line assistance when you are entering **ncp** commands. If you press RET at the **ncp** prompt, **ncp** displays all legal options for that command keyword.

EXAMPLE:

This example shows you the kind of information you must provide when you press RET after entering the **show** command.

```
>ncp RET
ncp>show RET

(active, adjacent, area, circuit, executor, known, line,
logging, loop, module, node, object):
```

If you enter **known** at the colon (:) prompt, **ncp** prompts you for additional information, as shown in the following example:

```
(active, adjacent, area, circuit, executor, known, line,
logging, loop, module, node, object): known

(areas, circuits, lines, logging, nodes, objects):
```

If you enter an incomplete **set** or **define** command, **ncp** prompts you individually for each possible parameter. You can respond in one of several ways:

- Enter RET to omit the current parameter and to be prompted for the next one.
- Enter a parameter value and press RET to set this parameter and to be prompted for the next one.
- Enter a period (.) and press RET when you have specified the parameters that you want and are ready to exit the prompting loop to execute the command.
- Enter CTRL/D to cancel the entire command.

1.2.5 Error Reporting

If a command executes successfully, the **ncp** prompt appears on the next line. If a command does not complete successfully, an error message is displayed to indicate the reason for the failure. Chapter 3 lists **ncp** error messages.

1.2.6 Event Logging

The logging monitor interface from the DECnet event-logging facility provides a mechanism by which a user-written program can process network events. You must specify the name of the monitoring program and the events to be logged by using the following **ncp set logging** commands:

set logging monitor name *name* **state on**

set logging monitor events *event-list*

where

name	Is the file descriptor of the program to receive the event information (default: **evl**).

event-list Identifies one or more event classes and types to be logged. (See Chapter 4 for a list of event classes and types.)

NOTE

The event monitor facility can be used in conjunction with event logging, either on the console or to a file.

When the monitor program is **evl** (default), events are logged at the logging console. If you write your own monitor program to process network events, your program starts when the network starts. When the network starts up, **evl** passes two arguments to your monitor program:

Argument 1 — your monitor program file name

Argument 2 — the file descriptor of a pipe from which to read

Your monitor program reads the events in ASCII from the pipe and then processes them according to your specifications.

1.3 Executing ncp Commands Remotely

You can use **ncp** to modify parameters or display information at a remote node. You can execute **ncp** commands on a remote node while logged in to your local node by using either the **ncp tell** prefix or the **ncp set executor node** command.

Before attempting to execute an **ncp** command remotely:

- Refer to network management documentation for the remote node to see which commands it supports.
- Check for any access restrictions.

1.3.1 Using the tell Prefix

To execute a single **ncp** command at a remote node, issue the command with the **tell** prefix. The following example shows how you can use the **tell** prefix to send the **ncp** command **show executor node** to a remote node NAVAHO.

```
ncp>tell navaho show exec node char RET
```

1.3.2 Using the set executor Command

To execute a series of **ncp** commands at a remote node, use the **set executor node** command to set the specified remote node as the executor. Subsequent commands that you issue are executed at that node until you restore control to your own node by issuing a **clear executor node** command. If you exit **ncp** while the executor is set to a remote node, control is automatically returned to the local node when you reenter **ncp**. For example:

```
% ncp RET
ncp>set executor 2.95 RET
show executor characteristics
  .
  .
  .
```

```
define executor address 2.95
set executor incoming timer
  .
  .
  .
ncp> clear executor node RET
```

In this example, 2.95 is the node address.

1.3.3 Access Restrictions on Remote Nodes

Before you issue commands to be executed at a remote node, you may have to supply access control information. Depending on the types of access restrictions set up by the system manager of the remote node, you can gain access to a remote node from a DECnet–ULTRIX node in the following ways:

- Append access control information to the node ID in the **ncp** command string.
- Include access control information in an "alias" definition for the node.
- Use proxy verification on the remote node.

For the procedure on how to supply access control information, see the *DECnet–ULTRIX Use* manual.

1.4 Network Management Privileges

You must have superuser privileges to use an **ncp** command that modifies a database. However, anyone can exercise the **ncp show** or **list** commands locally to display component information from the databases.

On ULTRIX systems, some **ncp** commands (such as **purge**, **define**, **set**, and **zero**) require superuser privileges. Other systems may also require privileges for the same **ncp** commands. For example, some DECnet–VAX NCP commands require system privileges (SYSPRV), others require operator privileges (OPER), and still others do not require privileges. To determine the privileges you need to issue commands at either a DECnet–ULTRIX node for remote execution or a remote node, see the Network Control Program documentation for the remote node.

1.5 Issuing load and trigger Commands

When you issue the **load** command, the load host must have service enabled on its Ethernet circuit or it cannot perform a down-line load. When you issue the **trigger** command, potential load hosts must have service enabled on their Ethernet circuits or they cannot perform a down-line load.

To enable service, use the following command format:

set circuit *circuit-id* **service enabled**

where *circuit-id* identifies the Ethernet circuit for the host.

On the command line, enter either the DECnet node name or the DECnet node address of the server. The **load** and **trigger** commands have a similar syntax:

load node *node-name*
trigger node *node-name*

or

load node *node-address*
trigger node *node-address*

The following examples use the **load** command to load a node named NAVAHO with a node address of 55.1024.

```
ncp>load node NAVAHO  RET
ncp>load node 55.1024  RET
```

If the remote node you want to load has an enabled password, you must specify this password on the **load** and **trigger** command lines. For example, to load node NAVAHO with service password FF55, type:

```
ncp>load node NAVAHO service password FF55  RET
```

or

```
ncp>load node 55.1024 service password FF55  RET
```

or

```
ncp>trigger node NAVAHO service password FF55  RET
```

or

```
ncp>trigger node 55.1024 service password FF55  RET
```

Chapter 2

NCP Command Descriptions

This chapter gives detailed descriptions and examples of each **ncp** command. Table 2-1 summarizes **ncp** command functions:

Table 2–1: ncp Command Functions

Command	Function
clear circuit	Removes specified circuit parameters from the volatile database.
clear executor	Resets specified executor parameters to their default values in the volatile database.
clear executor node	Resets the default executor to the local node.
clear logging	Removes specified logging parameters from the volatile database.
clear node	Removes names associated with the nodes or removes specified parameters from the volatile database.
clear object	Removes specified objects from the volatile database.
define	Creates or modifies specified parameters in the permanent database.
define circuit	Modifies specified circuit parameters in the permanent database.
define executor	Modifies specified executor node parameters in the permanent database.
define line	Modifies specified line parameters in the permanent database.
define logging	Creates or modifies logging component parameters in the permanent database.
define node	Changes the name or address associated with a node or resets specified node parameters in the permanent database.
define object	Creates or modifies parameters for the specified objects in the permanent database.
help	Displays information about **ncp** commands and parameters.
list	Displays specified parameters in the permanent database.
list circuit	Displays specified circuit information stored in the permanent database.
list executor	Displays specified local node information stored in the permanent database.

(continued on next page)

Table 2–1 (Cont.): ncp Command Functions

Command	Function
list line	Displays specified line information stored in the permanent database.
list logging	Displays specified logging information stored in the permanent database.
list node	Displays specified node information stored in the permanent database.
list object	Displays specified object information stored in the permanent database.
load node	Down-line loads a specified remote node on the same Ethernet.
load via	Down-line loads a remote node on the same Ethernet by way of the specified circuit.
loop circuit	Tests a circuit between the executor and another node in the network.
loop node/executor	Tests a node in the network.
purge	Removes specified parameters in the permanent database.
purge circuit	Removes specified circuit parameters in the permanent database.
purge executor	Resets specified executor parameters to their default values in the permanent database.
purge logging	Removes specified logging parameters in the permanent database.
purge node	Removes the names associated with the nodes or removes specified node parameters from the permanent database.
purge object	Removes specified objects from the permanent database.
set	Creates or modifies specified parameters in the volatile database.
set circuit	Modifies specified circuit parameters in the volatile database.
set executor	Creates or modifies specified executor node parameters in the volatile database.
set executor node	Sets a default executor for subsequent **ncp** commands.
set line	Modifies specified line parameters in the volatile database.
set logging	Modifies logging component parameters in the volatile database.
set node	Changes the name or address associated with a node or resets specified node parameters in the volatile database.
set object	Creates or modifies parameters for specified objects in the volatile database.
show	Displays specified parameters in the volatile database.
show circuit	Displays specified circuit information stored in the volatile database.
show executor	Displays specified local node information stored in the volatile database.
show line	Displays specified line information stored in the volatile database.

(continued on next page)

Table 2–1 (Cont.): ncp Command Functions

Command	Function
show logging	Displays specified logging information stored in the volatile database.
show node	Displays specified node information stored in the volatile database.
show object	Displays specified object information stored in the volatile database.
tell	Sends an **ncp** command to a remote node for execution.
trigger	Starts the bootstrap operation of a remote node on the same Ethernet so that the node multicasts a request for a down-line load.
trigger via	Starts the bootstrap operation of a remote node on the same Ethernet so that the node multicasts a request for a down-line load through a specified circuit.
zero	Sets counters to zero.
zero circuit	Sets circuit counters to zero for the specified circuit.
zero executor	Sets node counters associated with and maintained on the executor node to zero.
zero line	Sets line counters to zero for the specified line.
zero node	Sets line counters to zero for the specified node. The executor node maintains node counters on a per-node basis.

clear circuit

DESCRIPTION

Removes specified circuit parameters from the volatile database.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

clear circuit *circuit-id* [**receive password** | **transmit password**]

where

circuit *circuit-id*	Specifies the circuit for which parameters are to be removed.
receive password	Removes the circuit's receive password.
transmit password	Removes the circuit's transmit password.

EXAMPLE

This command deletes all parameters for circuit una-0 from the volatile database.

```
ncp>clear circuit una-0 RET
```

clear executor

DESCRIPTION

Resets specified executor parameters to their default values in the volatile database.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

clear executor { **identification** | **incoming timer** | **outgoing timer** }

where

identification	Removes the text identification string for the executor node.
incoming timer	Resets the incoming timer to its default value.
outgoing timer	Resets the outgoing timer to its default value.

EXAMPLE

This command resets the local node's incoming timer to its default value in the volatile database.

```
ncp> clear executor incoming timer RET
```

clear executor node

DESCRIPTION

Resets the default executor to the local node.

RESTRICTION

Do not use the **tell** prefix with this command.

SYNTAX

clear executor node

EXAMPLE

This command returns **ncp** command execution from a remote node to the local node.

```
ncp>clear executor node RET
```

clear logging

DESCRIPTION

Removes specified logging parameters from the volatile database.

RESTRICTIONS

You must have superuser privileges to execute this command.

Whenever you specify a *circuit*, *line*, *node*, or *sink* in a **clear logging** command, you must also include an **events** list or **known events** parameter.

SYNTAX

clear { **known logging** | **logging console** | **logging file** | **logging monitor** }

name

[**events** *event-list*

known events

[**circuit** *circuit-id* | **line** *line-id* | **node** *node-id*]

[**sink** { **executor** | **node** *node-id* }]]

where

circuit *circuit-id*
Inhibits event logging for the specified circuit.

events *event-list*
Removes the event class and types specified in *event-list* for the specified component.

known events
Removes all known events that DECnet–ULTRIX can generate for the specified component.

known logging
Removes parameters for all known logging components.

line *line-id*
Inhibits event logging for the specified line.

logging console
Removes parameters for the console-logging component.

logging file
Removes parameters for the file-logging component.

logging monitor
Removes parameters for the monitor-logging component.

name
Returns the specified logging component to its default name (console: **/device/console**; file: **/usr/adm/eventlog**; monitor **evl**.)

node *node-id*
Inhibits event logging for the specified node.

sink
Inhibits logging of the specified events at the specified node. You must specify one of the following qualifiers:

- **executor** — This is the default setting. Events are not to be logged at the executor node.
- **node** *node-id* — Events are not to be logged at the specified node.

EXAMPLE

This command ceases logging of event 2.1 to the console.

```
ncp> clear logging console event 2.1 RET
```

clear node

DESCRIPTION

Removes names associated with the nodes or removes specified node parameters from the volatile database.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

clear { **node** *node-id* | **known nodes** } { **diagnostic file** | **dump file** | **hardware address** | **host** | **load file** | **name** | **secondary loader** | **service circuit** | **service password** | **tertiary loader** }

or

clear { **node** *node-id* | **known nodes** } **all**

where

all	Removes all parameters for the specified node(s) so that the network no longer recognizes the nodes. Use of **all** precludes the use of any other parameters.
diagnostic file	Removes the identification of the down-line load diagnostic file.
dump file	Removes the up-line dump file identification.
hardware address	Removes the Ethernet address of the system hardware.
host	Removes the host node identification.
known nodes	Performs the specified function for all known nodes.
load file	Removes the identification of the down-line load file.
name	Removes the node name(s) associated with the specified node(s).
node *node-id*	Performs the specified function for the specified node only.
secondary loader	Removes the identification of the secondary down-line loading file.
service circuit	Removes the circuit parameter associated with the node for down-line loading purposes.
service password	Removes the password parameter. This password parameter is required to trigger the bootstrap mechanism of the node to be down-line loaded.

tertiary loader	**Removes the identification of the tertiary down-line loading file.**

EXAMPLE

This command removes all information for node BOSTON from the volatile database.

```
ncp> clear node boston all RET
```

clear object

DESCRIPTION

Removes specified objects from the volatile database.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

clear { **object** *object-name* | **known objects** }

where

object *object-name*	Specifies the object for which parameters are to be removed.
known objects	Specifies that parameters are to be removed for all known objects.

EXAMPLE

This command removes the network terminal handler (**dtermd**) from the volatile database.

```
ncp> clear object dtermd RET
```

define circuit

DESCRIPTION

Modifies specified circuit parameter(s) in the permanent database.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

define circuit *circuit-id* { **hello timer** *seconds* | **receive password** *password* | **state** *circuit-state* { **off** | **on** } | **transmit password** *password* }

where

circuit *circuit-id*
Specifies the circuit for which parameters are to be modified.

hello timer *seconds*
Specifies the frequency of routing hello messages sent to adjacent nodes on the circuit. The range is 1 through 8191.

receive password *password*
Specifies a 1- to 8-character ASCII password associated with the circuit that the executor expects to receive from the remote node during a routing initialization sequence.

state *circuit-state*
Specifies the circuit's operational state. There are two possible states:

- **off** — The circuit is not in use.
- **on** — The circuit is available for normal use or service functions.

transmit password *password*
Specifies a 1- to 8-character ASCII password associated with the circuit that the executor sends to the remote node during a routing initialization sequence.

EXAMPLES

This command makes circuit una-0 unavailable for use.

```
ncp>define circuit una-0 state off RET
```

define executor

DESCRIPTION

Creates or modifies specified executor node parameters in the permanent database.

NOTE

If you use the **define executor** command to issue a series of commands at a remote node, you can use the **tell** prefix to issue an **ncp** command to yet another remote node.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

define executor
- **address** *node-address*
- **delay factor** *number*
- **delay weight** *number*
- **gateway access** { **disable** | **enable** }
- **gateway user** *login-name*
- **identification** *id-string*
- **inactivity timer** *seconds*
- **incoming proxy** { **disable** | **enable** }
- **incoming timer** *seconds*
- **maximum links** *number*
- **maximum node counters** *number*
- **name** *node-name*
- **outgoing proxy** { **disable** | **enable** }
- **outgoing timer** *seconds*
- **pipeline quota** *number*
- **retransmit factor** *number*
- **segment buffer size** *number*

where

address *node-address*
: Specifies the address of the executor node. (Remember, **define** commands do not take effect until the next time you start up the network.)

delay factor *number*
: Specifies a number to multiply times 1/16 of the estimated round-trip delay to a node in order to set the retransmission timer for that node. The range is 1 through 255.

delay weight *number*
: Specifies a value to apply to a current round-trip delay to a remote node in order to update the estimated round-trip delay time. The range is 1 through 255.

gateway access

Specifies whether or not the executor allows the DECnet objects **fal** and **dlogind** to provide access to the Internet network. Choose one of the following modes:

disable — Turns off file transfer and remote log-in Internet access.

enable — Turns on file transfer and remote log-in Internet access.

The default value is **disable** if you choose not to install the Gateway during the DECnet–ULTRIX installation procedure. Otherwise, the default value is **enable**.

gateway user *login-name*

Specifies a default log-in name under which the DECnet objects **fal** and **dlogind** are to run if gateway access is enabled and a gateway request is received. The range is 1 through 32 characters.

identification *id-string*

Specifies the text identification string for the executor node. The range is 1 through 32 characters. You must use double quotation marks (") to delimit any string containing blanks or tabs. To include a quoted string within an identification string, enclose it within single quotation marks to distinguish the quotes from a string delimiter.

inactivity timer *seconds*

Specifies the maximum time the executor allows a link to remain idle (no user data traffic) before it checks to see whether the circuit still works. The range is 1 through 1024.

incoming proxy

Specifies whether the executor honors or ignores proxy log-in requests present on incoming logical links. Choose one of the following modes:

disable — All incoming proxy requests are ignored. Instead, access control information supplied in each connect request is used to validate the request.

enable — (Default.) Incoming proxy requests are honored, based on the source user, the source node, and the access control information.

incoming timer *seconds*

Specifies the maximum time a process has to answer an incoming connect request. If the process does not answer the connect request within this time interval, the node rejects the connect request on behalf of the process. The range is 1 through 1024.

maximum links *number*

Specifies the maximum number of active logical links for the executor node. The range is 1 through 1024. If you use this parameter in a **set** command, you must specify a value greater than the current maximum number of links. To decrease the value, you must issue a **define** command and restart the system.

maximum node counters *number*

Specifies the maximum number of node counters allowed. The range is 4 through 255, and the default is 32.

If you use this parameter in a **set** command, you must specify a value that is greater than the current maximum number of node counters. To decrease the value, you must issue a **define** command and restart the system.

name *node-name*
Specifies the node name of the executor.

outgoing proxy
Specifies whether or not the executor requests proxy log-in on outgoing connect requests. Choose one of the following modes:

- **disable** — The executor does not request proxy log-in on outgoing logical links, even when the user process attempts to enable it.
- **enable** — (Default.) The executor requests proxy log-in on all outgoing logical links, unless the user process explicitly disables it.

outgoing timer *seconds*
Specifies the maximum time the executor node waits for a pending connect request to be answered at a destination node. If the request is not answered in this time interval, the source process receives an error indication. The range is 1 through 1024.

pipeline quota *number*
Specifies the maximum number of bytes of buffer space that NSP can use for transmission and reception for each logical link. The range is 2,000 to 16,000. For satellite communications, you should use a value of 6,000 bytes or greater. The default is 4,096.

retransmit factor *number*
Specifies the number of times the executor restarts the retransmission timer before the logical link is disconnected. The range is 1 through 1024.

segment buffer size *number*
Specifies the size of the NSP message segment to be sent. This value is the maximum size message that the End Communications layer can transmit; it does not include routing or data link overhead. The range is 1 through 1,478, and the default is 576.

EXAMPLES

1. This command defines the executor address to 4.21 in the permanent database the next time the network is started up.

```
ncp>define executor address 4.21 RET
```

2. This command defines the maximum number of active logical links for the executor node to 20 in the volatile database.

```
ncp>define executor maximum links 20 RET
```

define line

DESCRIPTION

Modifies specified line parameters in the permanent database.

RESTRICTION

When you change the protocol and duplex parameter values of a point-to-point device, the operation mode of the device does not change until you restart the circuit. Therefore, you should turn the circuit off before changing the parameter, and then turn it back on afterward. For example:

```
ncp>define circuit dmv-0 state off RET
ncp>define line dmv-0 protocol ddcmp dmc duplex half RET
ncp>define circuit dmv-0 state on RET
```

SYNTAX

define line *line-id* { **controller** { **loopback** | **normal** } | **duplex** { **full** | **half** } | **protocol** { **ddcmp dmc** | **ddcmp point** | **ethernet** } }

where

line *line-id*
Specifies the line for which parameters are to be modified.

controller
Specifies the controller mode for the line. Choose one of the following modes:

- **loopback** — Internal device loopback mode.
- **normal** — Normal operating mode.

duplex
Represents the Physical Link hardware duplex mode of the line device. (Valid for DDCMP point-to-point devices only.) Choose one of the following modes:

- **full** — Full-duplex mode.
- **half** — Half-duplex mode.

protocol
Specifies the Data Link protocol to be used on the line. Choose one of the following modes:

- **ddcmp dmc** — Protocol for the DMC emulator. This setting is valid for dmv lines only.
- **ddcmp point** — Protocol for one end of a DDCMP point-to-point connection. This is the default for all DDCMP point-to-point connections, including dmv, dmc, and dmr lines.
- **ethernet** — Protocol for an Ethernet line. This is the default and the only valid choice for Ethernet lines (una, qna, sva, and bnt).

EXAMPLE

This command sets the controller for line una-0 to the **loopback** state.

```
ncp>define line una-0 controller loopback RET
```

define logging

DESCRIPTION

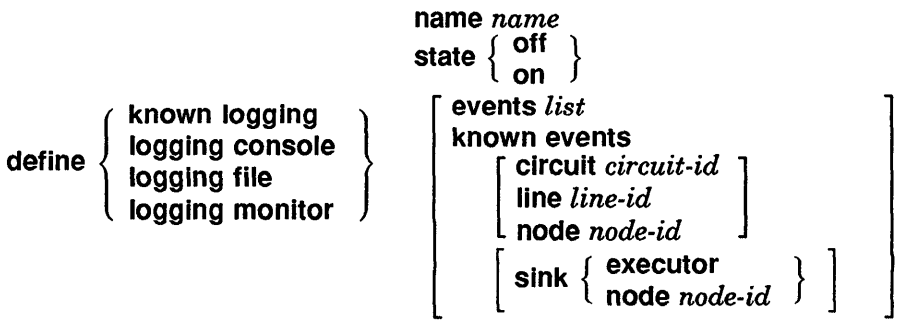
Creates or modifies logging component parameters in the permanent database.

RESTRICTIONS

You must have superuser privileges to use this command.

Whenever you specify a *circuit*, *line*, *node*, or *sink* in a **define logging** command, you must also specify an **events** *list* or **known events** parameter.

SYNTAX



where

circuit *circuit-id*	Logs the specified event(s) occurring on the specified circuit.
events *list*	Specifies the event class and type(s) to be logged.
known events	Specifies that all events that DECnet–ULTRIX can generate are to be logged.
known logging	Indicates that the specified parameters are to be created or modified for all known logging components.
line *line-id*	Logs the specified event(s) occurring on the specified line.
logging console	Indicates that the specified parameters are to be created or modified for the console logging component.
logging file	Indicates that the specified parameters are to be created or modified for the file logging component.
logging monitor	Indicates that the specified parameters are to be created or modified for the monitor logging component.
name *name*	Specifies the name of the console, monitor program, or file to which events are to be logged. The default name for a console is **/device/console**; for a monitor program it is **evl** and for a file it is **/usr/adm/eventlog**.
node *node-id*	Logs the specified event(s) occurring on the specified node.

sink	Identifies the node on which the specified events are to be logged. Choose one of the following qualifiers: **executor** (Default) executor node. **node** *node-id* The specified remote node.
state	Sets the operational state of the logging component on the executor node. When the state is **off**, events are discarded.

EXAMPLES

1. This command directs any known events on circuit una-0 to node BOSTON.

```
ncp> define known logging known events circuit una-0
 sink node boston RET
```

2. This command directs any occurrence of event 2.1 to the console at node PARIS.

```
ncp> define logging console event 2.1 sink node paris RET
```

define node

DESCRIPTION

Adds or modifies a node specification in the permanent database.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

define node *node-id* { **address** *node-address* | **diagnostic file** *file* | **dump file** *file* | **hardware address** *E-address* | **host** *node-id* | **load file** *file* | **name** *node-name* | **secondary [loader]** *file* | **service** { **disable** | **enable** } | **service circuit** *circuit-id* | **[service] password** *service-password* | **tertiary [loader]** *file* }

where

address *node-address*
: Specifies a new (unassigned) node address to be associated with the node-name used as the *node-id* in this command.

diagnostic file *file*
: (For Ethernet nodes only.) Specifies the file to be read when the node is down-line loaded and requests diagnostics.

dump file *file*
: Specifies the file that is to receive a copy of the system at the time of the crash when the node is up-line dumped.

hardware address *E-address*
: Identifies the Ethernet address that was originally assigned to the Ethernet controller for the node. This address is used during operations such as down-line load to communicate with the system before it has set up its physical address.

host *node-id*
: Specifies a host node for all service operations. The default value is the executor node ID.

load file *file*
: Specifies a file containing the system software for down-line loading to a node.

name *node-name*
: Specifies a new (unassigned) node name to be associated with the *node-address* used as the *node-id* in this command.

node *node-id*
: Specifies the node for which the specified function is to be performed.

secondary [loader] *file*
: Specifies a file containing secondary loader software for down-line loading to the node.

service	Specifies whether the node is enabled or disabled for down-line loading.
service circuit *circuit-id*	Specifies the circuit to be used for down-line loading and up-line dumping. This circuit is the default value for the **via** parameter of the **load** command.
[service] password *service-password*	Specifies the password required to trigger the bootstrap mechanism on the node. The password is a hexadecimal value of 1 to 16 characters.
tertiary [loader] *file*	Specifies a file containing tertiary loader software for down-line loading to the node.

EXAMPLE

This command associates the name BURGER with node 12.

```
ncp> define node 12 name burger RET
```

define object

DESCRIPTION

Creates or modifies parameters for specified object(s) in the permanent database.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

define { **object** *object-name* | **known objects** } { **accept** { **deferred** | **immediate** } | **default user** *login-name* | **file** *file-id* | **number** *number* | **type** { **sequenced packet** | **stream** } }

where

accept Specifies whether incoming connect requests are to be accepted or rejected by the DECnet object spawner or whether they are to be forwarded to the object for processing. Choose one of the following modes:

- **deferred** The object processes any optional data sent with a connect request and can set optional data to be returned when it accepts or rejects the connect.
- **immediate** (Default.) The DECnet object spawner determines whether to accept or reject all connect requests.

default user *login-name* Specifies a default log-in name under which the object is to run if no access control information is supplied with a connect request and no proxy information is defined locally.

file *file-id* Specifies an executable file or shell script used to invoke the specified object.

known objects Specifies that parameters are to be created or modified for all known objects.

number *number* Specifies the object number to be identified with the specified object name. The default is 0. See Appendix B for a list of object names and numbers.

object *object-name* Specifies that parameters are to be created or modified for the named object only (a maximum of 16 alphanumeric characters).

type	Identifies the socket type. Choose one of the socket types:
sequenced packet	(Default) provides a bidirectional, reliable, sequenced, and unduplicated flow of data while preserving record boundaries.
stream	provides same data flow properties as above without record boundaries.

EXAMPLE

This command defines the Network Management Listener (**nml**) as object number 19 with **/etc/nml** as the executable file. It also specifies that **nml** is to run under the log-in name "guest" if no access control information is supplied with a connect request and no proxy information is defined locally.

```
ncp>define object nml number 19 file /etc/nml default user guest RET
```

help

DESCRIPTION

Displays general information about **ncp** commands and parameters. Typing **help** displays the topics for which information is available; typing help and a specific topic or command name displays information about that topic or command.

SYNTAX

help [*topic...*]

topic — A command word listed in the **help** display. You may specify up to eight topics separated by spaces or tabs.

EXAMPLES

1. This command displays all command verbs for which further information exists.

```
ncp>help RET

Help available for:
clear           command         define          exit
help            list            load            loop
parameters      prompting       purge           set
show            tell            trigger         zero
ncp>
```

2. This command provides a description of the **ncp** command **clear circuit** and displays command words for which further information exists.

```
ncp>help clear circuit RET

clear circuit

Use the clear circuit command to remove circuit parameters from the
volatile database on the executor node.  Use the purge circuit
command to remove circuit parameters from the permanent database
on the executor node.

clear    circuit circuit-id     (parameters...)

Help available for:
circuit-id          all         receive password     transmit password
examples
ncp>
```

3. This command provides a description of the **ncp** command **show** and displays command words for which further information exists.

```
ncp>help show RET

show

Use the show command to display information from the volatile database
on the executor node.  Use the list command to display information
from the permanent database on the executor node.
```

```
Help available for:
characteristics     counters        events          status
summary             known           circuit         executor
line                logging         node            object
module
ncp>
```

list circuit

DESCRIPTION

Displays specified circuit information stored in the permanent database.

SYNTAX

list { **circuit** *circuit-id* | **known circuits** } [**characteristics** | **status** | **summary**]

where

characteristics	Displays parameters that are currently set for the circuit.
circuit *circuit-id*	Displays information for the specified circuit only.
known circuits	Displays information for all known circuits.
status	Shows information that usually reflects network activity for the running network. Depending on the component, this can include the local node and its operational state, reachable and unreachable nodes, and circuits and lines and their operational states.
summary	(Default) shows abbreviated information provided for the **characteristics** and **status** display types.

EXAMPLE

This command displays circuit characteristics for all known circuits in the permanent database.

```
ncp> list known circuits characteristics RET

Known Circuit Permanent Characteristics as of Tue Nov 21 10:57:23 EST 1990

Circuit  =  UNA-0

Hello timer                          = 10
Type                                 = Ethernet

ncp>
```

list executor

DESCRIPTION

Displays specified local node information stored in the permanent database.

SYNTAX

list executor [**characteristics** | **status** | **summary**]

where

characteristics — Displays parameters that are currently set for the executor.

status — Shows information that usually reflects network activity for the running network. Depending on the component, this can include the local node and its operational state, reachable and unreachable nodes, and circuits and lines and their operational states.

summary — (Default) shows abbreviated information provided for the **characteristics** and **status** display types.

EXAMPLE

This command displays local node status information from the permanent database.

```
ncp> list executor status RET

Executor Permanent Status as of WED Nov 15 16:13:45 EST 1990

Executor node =  2.95 (OHIO)

State                                = On
ncp>
```

list line

DESCRIPTION

Displays specified line information stored in the permanent database.

SYNTAX

list { **line** *line-id* | **known lines** } [**characteristics** | **status** | **summary**]

where

line *line-id*	Displays information for the specified line only.
known lines	Displays information for all known lines.
characteristics	Displays parameters that are currently set for the line.
status	Shows information that usually reflects network activity for the running network. Depending on the component, this can include the local node and its operational state, reachable and unreachable nodes, and circuits and lines and their operational states.
summary	(Default) shows abbreviated information provided for the **characteristics** and **status** display types.

EXAMPLE

This command displays information about line una-0.

```
ncp> list line una-0 summary RET
Line Permanent Summary as of Wed Nov 15 16:17:06 EST 1990
Line                    State
BNT-0                   On
ncp>
```

list logging

DESCRIPTION

Displays specified logging information stored in the permanent database.

SYNTAX

list { **known logging** | **logging console** | **logging file** | **logging monitor** } [**characteristics** | **status** | **summary** | **events**] [**known sinks** | **sink node** *node-id*]

where

characteristics Displays parameters that are currently set for the executor, line, or circuit.

events Displays event class and type information for the given logging component. Choose one of the following qualifiers:

- **known sinks** (Default) displays logging information for all known sink nodes.
- **sink node** *node-id* Displays logging information for the specified sink node only.

known logging Displays information for all known logging components.

logging console Displays information for the console logging component.

logging file Displays information for the file logging component.

logging monitor Displays information for the monitor logging component.

status Shows information that usually reflects network activity for the running network. Depending on the component, this can include the local node and its operational state, reachable and unreachable nodes, and circuits and lines and their operational states.

summary (Default) shows abbreviated information provided for the **characteristics** and **status** display types.

EXAMPLE

This command displays event class and type information for the logging file on node N1834P.

```
ncp>list logging file events sink node n1834p RET
```

list node

DESCRIPTION

Displays specified node information stored in the permanent database.

SYNTAX

list { **node** *node-id* | **known nodes** } [**characteristics** | **status** | **summary**]

where

characteristics	Displays parameters that are currently set for the node.
known nodes	Displays information for all known nodes.
node *node-id*	Displays information for the specified node only.
status	Shows information that usually reflects network activity for the running network. Depending on the component, this can include the local node and its operational state, reachable and unreachable nodes, and circuits and lines and their operational states.
summary	(Default) shows abbreviated information provided for the **characteristics** and **status** display types.

EXAMPLE

This command displays error and performance statistics for all known nodes in the permanent database.

```
ncp> list node art summary RET

Node Permanent Summary as of Tue Nov 21 11:03:24 EST 1990

Executor node  =  2.39 (ART)

State                         = On
ncp>
```

list object

DESCRIPTION

Displays specified object information stored in the permanent database.

SYNTAX

list { **object** *object-name* / **known objects** } [**characteristics** / **summary**]

where

characteristics	Displays parameters that are currently set for the object.
counters	Provides counter information for circuits, lines, and nodes.
known objects	Displays information for all known objects.
object *object-name*	Displays information for the specified object only.
status	Shows information that usually reflects network activity for the running network. Depending on the component, this can include the local node and its operational state, reachable and unreachable nodes, and circuits and lines and their operational states.
summary	(Default) shows abbreviated information provided for the **characteristics** and **status** display types.

EXAMPLE

This command displays information about the Network Management Listener (**nml**).

```
ncp> list object nml RET

Object Permanent Summary as of Wed Nov 15 16:27:55 EST 1990

Object          Number          File

nml             19              /usr/ect/nml
ncp>
```

load node

DESCRIPTION

Down-line loads a specified remote node on the same Ethernet. Any parameter that you do not specify defaults to the value in the volatile database on the executor node.

With the **load node** command, the node that initiates the loading sequence is also the load host that performs the down-line load.

RESTRICTIONS

Before you can execute this command:

- You must have superuser privileges.
- The **mop_mom** utility must be installed during the the ULTRIX software installation.
- Service must be enabled on the remote node.
- You must have the service password if a DECnet service password is defined on the remote node.

SYNTAX

load node *node-id* { **address** *node-address* | **from** *load-file* | **host** *node-id* | **name** *node-name* | **physical address** *E-address* | **secondary [loader]** *file* | **[service] password** *service-password* | **tertiary [loader]** *file* | **via** *circuit-id* }

where

address *node-address*
: Specifies the address that the node is to use when it comes up.

from *load-file*
: Specifies the file containing the system software to be down-line loaded.

host *node-id*
: Specifies the default host that the node is to use when it comes up.

name *node-name*
: Specifies the name that the node is to use when it comes up.

node *node-id*
: Specifies the node to be down-line loaded.

physical address *E-address*
: (For Ethernet nodes only.) Identifies the Ethernet physical address that the node currently uses to identify itself. (Required for Ethernet circuits if the hardware address parameter has not been specified in the volatile database; see **set node**.)

secondary [loader] *file*	Specifies a file containing secondary loader software for down-line loading to the node.
[service] password *service password*	Specifies the password required to trigger the bootstrap mechanism on the node. The password is a hexadecimal value of 1 to 16 characters.
tertiary [loader] *file*	Specifies a file containing tertiary loader software for down-line loading to the node.
via *circuit-id*	Specifies the circuit over which the load is to take place.

EXAMPLE

This command loads node BOSTON using a service password.

```
ncp>load node boston service password FF55 RET
```

load via

DESCRIPTION

Down-line loads a remote node on the same Ethernet by way of the specified circuit. You must include the physical address in the command.

With the **load via** command, the node that initiates the loading sequence is also the load host that performs the down-line load.

RESTRICTIONS

Before you can execute this command:

- You must have superuser privileges.
- The **mop_mom** utility must be installed during the ULTRIX software installation.
- Service must be enabled on the remote node.
- You must have the service password if a DECnet service password is defined on the remote node.

SYNTAX

load via *circuit-id* { **address** *address* | **from** *load-file* | **host** *node-id* | **name** *node-name* | **physical address** *E-address* | **secondary [loader]** *file* | **[service] password** *service-password* | **tertiary [loader]** *file* }

where

address *node-address*
: Specifies the address that the node is to use when it comes up.

from *load-file*
: Specifies the file containing the system software to be down-line loaded.

host *node-id*
: Specifies the default host that the node is to use when it comes up.

name *node-name*
: Specifies the name that the node is to use when it comes up.

physical address *E-address*
: (For Ethernet nodes only.) Identifies the Ethernet physical address that the node currently uses to identify itself. If the circuit is an Ethernet circuit, you must include the physical address in the command.

secondary [loader] *file*
: Specifies a file containing secondary loader software for down-line loading to the node.

[service] password *service-password*	Specifies the password required to trigger the bootstrap mechanism on the node. The password is a hexadecimal value of 1 to 16 characters.
tertiary [loader] *file*	Specifies a file containing tertiary loader software for down-line loading to the node.
via *circuit-id*	Specifies the circuit over which the load is to take place.

EXAMPLE

This command loads the node connected to the executor node by circuit una-0.

```
ncp>load via una-0 physical address aa-00-03-00-01-19 service password FF55 RET
```

loop circuit

DESCRIPTION

Tests a circuit between the executor and another node in the network. You can specify a destination node using either its node name or its physical address. If you do not specify a destination node, the loop request is sent to the multicast address and the first node to respond completes the loop test.

If you specify a destination node, you can also specify a third node to assist with the test. The **help** parameter lets you specify the type of assistance you want.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

loop circuit [**node** *node-name* | **physical address** *E-address* [**help** *help-type* [{ **full** | **receive** | **transmit** } { **node** *node-name* | **assistant node** *node-name* | **physical address** *E-address* | **assistant physical address** *E-address* }]]]

[**count** *count* | **length** *length* | **with** { **mixed** | **ones** | **zeros** }]

where

assistant physical address *E-address*
: Specifies the physical address (not a multicast address) of an Ethernet node that is to assist in the loop circuit test.

assistant node *node-name*
: Specifies the name of an Ethernet node that is to assist in the loop circuit test.

circuit *circuit-id*
: (Does not apply to X.25 circuits.) Specifies the circuit to use for the loopback test.

count *count*
: Specifies the number of blocks to be sent during loopback testing. The valid range is 1 (default) through 1024.

help *help-type*

Specifies the degree to which a third node is to assist with an Ethernet loop circuit test. Choose one of the following types:

full (Default if an assistant node is specified but **help** is not.) The assistant node is to both receive and transmit the test packet (see the example).

receive The assistant node is only to receive the test packet.

transmit The assistant node is only to transmit the test packet.

NOTE

If you specify **help**, you must also specify either the **physical address** and **assistant physical address** parameters or, if the addresses are not known, the **node** and **assistant node** parameters.

length *length*

Specifies the length in bytes of each block to be sent during loopback testing. The default is 40 bytes. When testing over the Ethernet, the allowable length is from 1 byte to the maximum length of the data pattern, which varies according to the level of assistance:

Level of Assistance	Maximum Length
No assistance	1486 bytes
Transmit or receive assistance	1478 bytes
Full assistance	1470 bytes

node *node-name*

Specifies the name of an Ethernet node that is to be the destination of a loop-test message.

physical address *E-address*

Specifies the physical address (not a multicast address) of an Ethernet node that is to be the destination of a loop-test message.

with

Specifies the type of binary information to be sent during testing. If you omit this parameter, a combination of ones and zeros (the **mixed** data type) is sent. Choose one of the following data types:

mixed (Default.) A combination of ones and zeros.

ones

zeros

EXAMPLE

This command tests the circuit una-0 with the assistance of the node specified in assistant physical address by performing the following tasks:

1. The initiating node sends a test packet to the assistant node.
2. The assistant node processes the packet and passes the packet to the destination node specified in the physical address.
3. The destination node receives the packet and transmits the packet back to the assistant node.
4. The assistant node then returns the packet to the initiating node.
5. The **count** parameter indicates that this process is to be performed 10 times.

```
ncp>loop circuit una-0 help full physical address aa-00-04-00-f9-04
assistant physical address aa-00-04-00-04-a9 count 10 RET
```

loop executor

DESCRIPTION

Tests the executor node in the network. You can include access control information if the node requires it. This command causes test blocks of data to be transmitted to the specified node.

SYNTAX

loop executor [**count** *count* | **length** *length* | **with** { **mixed** | **ones** | **zeros** }]

where

executor — Specifies the executor node for loopback testing.

count *count* — Specifies the number of blocks to be sent during loopback testing. The valid range is 1 (default) through 1024.

length *length* — Specifies the length in bytes of each block to be sent during loopback testing. The length must be a decimal integer in the range of 1 through *n*, where *n* must be less than the smaller of either the local looper buffer size or the remote mirror buffer size. The default is 40 bytes.

node *node-id* — Specifies a node for loopback testing.

with — Specifies the type of binary information to be sent during testing. If you omit this parameter, a combination of ones and zeros (the **mixed** data type) is sent. Choose one of the following data types:

- **mixed** (Default.) A combination of ones and zeros.
- **ones**
- **zeros**

EXAMPLE

This command loops 10 blocks of mixed test messages to executor node. Each block is 40 bytes.

```
ncp> loop executor count 10 RET
```

loop node

DESCRIPTION

Tests a node in the network. You can include access control information if the node requires it. This command causes test blocks of data to be transmitted to the specified node.

SYNTAX

loop { **node** *node-id*[*acc-con-info*] } [**count** *count* | **length** *length* | **with** { **mixed** | **ones** | **zeros** }]

where

acc-con-info — Specifies access control information (if required by the remote node).

count *count* — Specifies the number of blocks to be sent during loopback testing. The valid range is 1 (default) through 1024.

length *length* — Specifies the length in bytes of each block to be sent during loopback testing. The length must be a decimal integer in the range of 1 through *n*, where *n* must be less than the smaller of either the local looper buffer size or the remote mirror buffer size. The default is 40 bytes.

node *node-id* — Specifies a node for loopback testing.

with — Specifies the type of binary information to be sent during testing. If you omit this parameter, a combination of ones and zeros (the **mixed** data type) is sent. Choose one of the following data types:

mixed (Default.) A combination of ones and zeros.

ones

zeros

EXAMPLE

This command loops 10 blocks of mixed test messages to remote node OHIO. Each block is 40 bytes.

```
ncp>loop node ohio count 10 RET
```

purge circuit

DESCRIPTION

Removes specified circuit parameters from the permanent database.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

purge circuit *circuit-id* [**receive password** | **transmit password**]

where

circuit *circuit-id*	Specifies the circuit for which parameters are to be removed.
receive password	Removes the circuit's receive password.
transmit password	Removes the circuit's transmit password.

EXAMPLE

This command deletes all parameters for circuit una-0 from the permanent database.

```
ncp>purge circuit una-0 RET
```

purge executor

DESCRIPTION

Resets specified executor parameters to their initial values in the permanent database.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

purge executor { **identification** | **incoming timer** | **outgoing timer** }

where

identification	Removes the text identification string for the executor node. The valid range is 1 to 32 alphanumeric characters.
incoming timer	Resets the local node's incoming timer to its default value.
outgoing timer	Resets the local node's outgoing timer to its default value.

EXAMPLE

This command resets the local node's incoming timer to its default value in the volatile database.

```
ncp> purge executor incoming timer RET
```

purge logging

DESCRIPTION

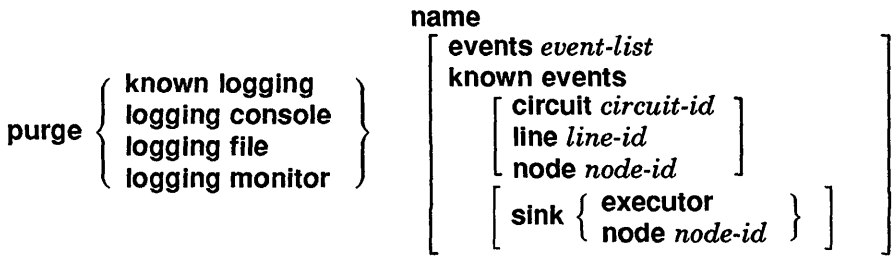
Removes specified logging parameters from the permanent database.

RESTRICTIONS

You must have superuser privileges to execute this command.

Whenever you specify a *circuit*, *line*, *node*, or *sink* in a **purge logging** command, you must also include an **event** *list* or **known events** parameter.

SYNTAX



where

circuit *circuit-id*
Inhibits event logging for the specified circuit.

events *event-list*
Removes the event class and types specified in *event-list* for the specified component.

known events
Removes all known events that DECnet–ULTRIX can generate for the specified component.

known logging
Removes parameters for all known logging components.

line *line-id*
Inhibits event logging for the specified line.

logging console
Removes parameters for the console-logging component.

logging file
Removes parameters for the file logging component.

logging monitor
Removes parameters for the monitor logging component.

name
Returns the specified logging component to its default name (console: **/device/console**; file: **/usr/adm/eventlog**; monitor **evl**).

node *node-id*
Inhibits event logging for the specified node.

sink
Inhibits logging of the specified events at the specified node. Choose one of the following parameters:

executor
This is the default setting. Events are not to be logged at the executor node.

node *node-id*
Events are not to be logged at the specified node

EXAMPLE

This command ceases logging of event 2.1 to the console.

```
ncp>purge logging console event 2.1 RET
```

purge node

DESCRIPTION

Removes the names associated with the nodes or removes specified node parameters from the permanent database.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

purge { **node** *node-id* | **known nodes** } { **diagnostic file** | **dump file** | **hardware address** | **host** | **load file** | **name** | **secondary loader** | **service circuit** | **service password** | **tertiary loader** }

or

purge { **node** *node-id* | **known nodes** } **all**

where

all	Removes all parameters for the specified node(s) so that the network no longer recognizes the node(s). Use of **all** precludes the use of any other parameters.
diagnostic file	Removes the identification of the down-line load diagnostic file.
dump file	Removes the up-line dump file identification.
hardware address	Removes the Ethernet address of the system hardware.
host	Removes the host node identification.
known nodes	Performs the specified function for all known nodes.
load file	Removes the identification of the down-line load file.
name	Removes the node name(s) associated with the specified node(s).
node *node-id*	Performs the specified function for the specified node only.
secondary loader	Removes the identification of the secondary down-line loading file.
service circuit	Removes the circuit parameter associated with the node for down-line loading purposes.
service password	Removes the password parameter. This password parameter is required to trigger the bootstrap mechanism of the node to be down-line loaded.

tertiary loader — Removes the identification of the tertiary down-line loading file.

EXAMPLE

This command removes all information for node BOSTON from the permanent database.

```
ncp>purge node boston all RET
```

purge object

DESCRIPTION

Removes specified object(s) from the permanent database.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

purge { **object** *object-name* | **known objects** }

where

known objects	Specifies that parameters are to be removed for all known objects.
object *object-name*	Specifies the object for which parameters are to be removed.

EXAMPLE

This command removes the network terminal handler **(dtermd)** from the permanent database.

```
ncp> purge object dtermd RET
```

set circuit

DESCRIPTION

Modifies specified circuit parameter(s) in the volatile database.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

set circuit *circuit-id* { **hello timer** *seconds* | **receive password** *password* | **state** { **off** | **on** } | **transmit password** *password* }

or

set circuit *circuit-id* **all**

where

all	Updates the volatile database with all of the parameters defined for the specified circuit in the permanent database. Use of **all** precludes use of any other parameters.
circuit *circuit-id*	Specifies the circuit for which parameters are to be modified.
hello timer *seconds*	Specifies the frequency of routing hello messages sent to adjacent nodes on the circuit. The range is 1 through 8,191.
receive password *password*	Specifies a 1- to 8-character ASCII password associated with the circuit that the executor expects to receive from the remote node during a routing initialization sequence.
transmit password *password*	Specifies a 1- to 8-character ASCII password associated with the circuit that the executor sends to the remote node during a routing initialization sequence.

EXAMPLES

1. This command updates the volatile database with all of the parameters defined for circuit qna-0 in the permanent database.

```
ncp>set circuit qna-0 all RET
```

2. This command makes circuit una-0 unavailable for use.

```
ncp>set circuit una-0 state off RET
```

set executor

DESCRIPTION

Creates or modifies specified executor node parameters in the volatile database.

NOTE

If you use the **set executor** command to issue a series of commands at a remote node, you can use the **tell** prefix to issue an **ncp** command to yet another remote node.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

set executor { **address** *node-address* | **delay factor** *number* | **delay weight** *number* | **gateway access** { **disable** | **enable** } | **gateway user** *login-name* | **identification** *id-string* | **inactivity timer** *seconds* | **incoming proxy** { **disable** | **enable** } | **incoming timer** *seconds* | **maximum links** *number* | **maximum node counters** *number* | **name** *node-name* | **outgoing proxy** { **disable** | **enable** } | **outgoing timer** *seconds* | **pipeline quota** *number* | **retransmit factor** *number* | **segment buffer size** *number* | **state** { **on** | **off** | **shut** | **restricted** } }

or

set executor all

where

all Updates the volatile database with all of the parameters defined for the executor node in the permanent database. Use of **all** precludes use of any other parameters.

delay factor *number*
Specifies a number to multiply times 1/16 of the estimated round-trip delay to a node in order to set the retransmission timer for that node. The range is 1 through 255.

delay weight *number*
Specifies a value to apply to a current round-trip delay to a remote node in order to update the estimated round-trip delay time. The range is 1 through 255.

gateway access
Specifies whether or not the executor allows the DECnet objects **fal** and **dlogind** to provide access to the Internet network. The default value is **disable** if you choose not to install the Gateway during the DECnet–ULTRIX installation procedure. Otherwise, the default value is **enable**. Choose one of the following values:

disable Turns off file transfer and remote log-in Internet access.

enable Turns on file transfer and remote log-in Internet access.

gateway user *login-name*
Specifies a default log-in name under which the DECnet objects **fal** and **dlogind** are to run if gateway access is enabled and a gateway request is received. The range is 1 through 32 characters.

identification *id-string*
Specifies a text identification string for the executor node. The range is 1 through 32 characters. You must use double quotation marks (") to delimit a string with blanks or tabs. To include a quoted string within an identification string, enclose the quoted string within single quotation marks.

inactivity timer *seconds*
Specifies the maximum time the executor will allow a link to remain idle (no user data traffic) before it checks to see whether the circuit still works. The range is 1 through 1024.

incoming proxy
Specifies whether the executor honors or ignores proxy log-in requests present on incoming logical links. Choose one of the following values:

disable All incoming proxy requests are ignored. Instead, access control information supplied in each connect request is used to validate the request.

enable (Default.) Incoming proxy requests are honored, based on the source user, the source node, and the access control information.

incoming timer *seconds*
Specifies the maximum time a process has to answer an incoming connect request. If the process does not answer the connect request within this time interval, the node will reject the connect request on behalf of the process. The range is 1 through 1024.

maximum links *number*
Specifies the maximum number of active logical links for the executor node. The range is 1 through 1024. If you use this parameter in a **set** command, you must specify a value greater than the current maximum number of links. To decrease the value, you must issue a **define** command and restart the system.

maximum node counters *number*

Specifies the maximum number of node counters allowed. The range is 4 through 255, and the default is 32.

NOTE

If you use this parameter in a **set** command, you must specify a value that is greater than the current maximum number of node counters. To decrease the value, you must issue a **define** command and restart the system.

name *node-name*

Specifies the node name of the executor.

outgoing proxy

Specifies whether or not the executor requests proxy log-in on outgoing connect requests. Choose one of the following values:

disable The executor does not request proxy log-in on outgoing logical links, even when the user process attempts to enable it.

enable (Default.) The executor requests proxy log-in on all outgoing logical links, unless the user process explicitly disables it.

outgoing timer *seconds*

Specifies the maximum time the executor node waits for a pending connect request to be answered at a destination node. If the request is not answered in this time interval, the source process receives an error indication. The range is 1 through 1024.

pipeline quota *number*

Specifies the maximum number of bytes of buffer space that NSP can use for transmission and reception for each logical link. The range is 2,000 to 16,000. For satellite communications, you should use a value of 6,000 bytes or greater. The default is 4,096.

retransmit factor *number*

Specifies the number of times the executor restarts the retransmission timer before the logical link is disconnected. The range is 1 through 1024.

segment buffer size *number*

Specifies the size of the NSP message segment to be sent. This value is the maximum size message that the End Communications layer can transmit; it does not include routing or data link overhead. The range is 1 through 1,478, and the default is 576.

state

Specifies the executor node's operational state. Use of **state** precludes use of any other parameters. Choose one of the following values:

on Allows logical links.

off Does not allow new links, terminates existing links, and stops route-through traffic.

restricted Does not allow new inbound links.

shut Does not allow new links but does not terminate existing links switches to the **off** state when all links have quit.

EXAMPLE

This command sets the maximum number of active logical links for the executor node to 20 in the volatile database.

```
ncp> set executor maximum links 20 RET
```

set executor node

DESCRIPTION

Sets the default executor as the specified remote node. This causes subsequent remotely executable **ncp** commands to be executed at the specified destination.

RESTRICTION

You cannot use the **tell** prefix with this command.

SYNTAX

set executor node *node-id*[*acc-con-info*]

where

acc-con-info	Specifies access control information (if required by the remote node).
node *node-id*	Specifies the remote node (by address, alias, or name) where subsequent **ncp** commands will be executed.

EXAMPLE

This command sets remote node EARTH (*user*: people; *password*: peace) to executor status. Future commands will be sent to node EARTH for execution.

```
ncp> set executor node earth/people/peace RET
```

set line

DESCRIPTION

Modifies the specified line parameter(s) in the volatile database.

RESTRICTION

When you change the protocol and duplex parameter values of a point-to-point device, the operation mode of the device does not change until you restart the circuit. Therefore, you should turn the circuit off before changing the parameter, and then turn it back on afterward. For example:

```
ncp>set circuit dmv-0 state off [RET]
ncp>set line dmv-0 protocol ddcmp dmc duplex half   [RET]
ncp>set circuit dmv-0 state on   [RET]
```

SYNTAX

set line *line-id* { **controller** { **loopback** \| **normal** } \| **duplex** { **full** \| **half** } \| **protocol** { **ddcmp dmc** \| **ddcmp point** \| **ethernet** } }

or

set line *line-id* **all**

where

all — Updates the volatile database with the parameters defined for the specified line in the permanent database.

controller — Specifies the controller mode for the line. Choose one of the following qualifiers:

- **loopback** — Internal device loopback mode.
- **normal** — Normal operating mode.

duplex — Represents the Physical Link hardware duplex mode of the line device. (Valid for DDCMP point-to-point devices only.) Choose one of the following qualifiers:

- **full** — Full-duplex mode.
- **half** — Half-duplex mode.

line *line-id* — Specifies the line for which parameters are to be modified.

protocol Specifies the Data Link protocol to be used on the line. Choose one of the following parameters:

ddcmp dmc	Protocol for the DMC emulator. This setting is valid for dmv lines only.
ddcmp point	Protocol for one end of a DDCMP point-to-point connection. This is the default for all DDCMP point-to-point connections, including dmv, dmc, and dmr lines.
ethernet	Protocol for an Ethernet line. This is the default and the only valid choice for Ethernet lines (una, qna, sva, and bnt).

EXAMPLE

This command sets the controller for line una-0 to the **loopback** state.

```
ncp> set line una-0 controller loopback RET
```

set logging

DESCRIPTION

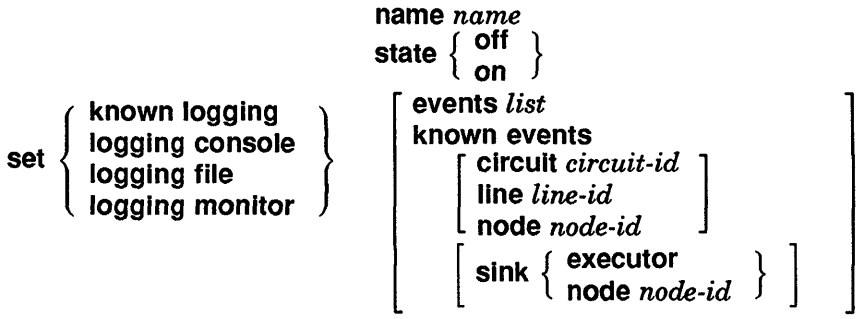
Creates or modifies logging component parameters in the volatile database.

RESTRICTIONS

You must have superuser privileges to use this command.

Whenever you specify a *circuit*, *line*, *node*, or *sink* in a **set logging** command, you must also include an **events** *list* or **known events** parameter.

SYNTAX



where

circuit *circuit-id*
Logs the specified event(s) occurring on the specified circuit.

events list
Specifies the event class and type(s) to be logged.

known events
Specifies that all events that DECnet–ULTRIX can generate are to be logged.

known logging
Indicates that the specified parameters are to be created or modified for all known logging components.

line *line-id*
Logs the specified event(s) occurring on the specified line.

logging console
Indicates that the specified parameters are to be created or modified for the console logging component.

logging file
Indicates that the specified parameters are to be created or modified for the file logging component.

logging monitor
Indicates that the specified parameters are to be created or modified for the monitor logging component.

name *name*
Specifies the name of the console, monitor program, or file to which events are to be logged. The default name for a console is **/device/console**; for a monitor program it is **evl** and for a file it is **/usr/adm/eventlog**.

node *node-id*
Logs the specified event(s) occurring on the specified node.

sink	Identifies the node where the specified events are to be logged. Choose one of the following qualifiers:
	executor (Default) executor node.
	node *node-id* The specified remote node.
state	Sets the operational state of the logging component on the executor node. When the state is **off**, events are discarded.

EXAMPLES

1. This command directs any known events on circuit una-0 to node BOSTON.

```
ncp>set known logging known events circuit una-0 sink node boston RET
```

2. This command directs any occurrence of event 2.1 to the console at node PARIS.

```
ncp>set logging console event 2.1 sink node paris RET
```

set node

DESCRIPTION

Modifies a node specification in the volatile database.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

set { **node** *node-id* } { **address** *node-address* | **diagnostic file** *file* | **dump file** *file* | **hardware address** *E-address* | **host** *node-id* | **load file** *file* | **name** *node-name* | **secondary [loader]** *file* | **service** { **disable** | **enable** } | **service circuit** *circuit-id* | **[service] password** *service-password* | **tertiary [loader]** *file* }

or

set { **node** *node-id* } **all**

where

all
Updates the volatile database with all of the parameters defined in the permanent database. Do not use any other parameter on the same command line.

address *node-address*
Specifies a new node address for the node.

diagnostic file *file*
(For Ethernet nodes only.) Specifies the file to be read when the node is down-line loaded and requests diagnostics.

dump file *file*
Specifies the file that is to receive a copy of the system at the time of the crash when the node is up-line dumped.

hardware address *E-address*
Identifies the Ethernet address that was originally assigned to the Ethernet controller for the node. This address is used during operations such as down-line load to communicate with the system before it has set up its physical address.

host *node-id*
Specifies a host node for all service operations. The default value is the executor node ID.

known nodes
(Valid for **all** only.) Specifies that the function is to be performed for all known nodes.

load file *file*	Specifies a file containing the system software for down-line loading to a node.
name *node-name*	Specifies a new (unassigned) node name to be associated with the *node-address* used as the *node-id* in this command.
node *node-id*	Specifies the node for which the specified function is to be performed.
secondary [loader] *file*	Specifies a file containing secondary loader software for down-line loading to the node.
service	Specifies whether the node is enabled or disabled for down-line loading.
service circuit *circuit-id*	Specifies the circuit to be used for down-line loading and up-line dumping. This circuit is the default value for the **via** parameter of the **load** command.
[service] password *service-password*	Specifies the password required to trigger the bootstrap mechanism on the node. The password is a hexadecimal value of 1 to 16 characters.
tertiary [loader] *file*	Specifies a file containing tertiary loader software for down-line loading to the node.

EXAMPLE

This command associates the name BURGER with node 12.

```
ncp>set node 12 name burger RET
```

set object

DESCRIPTION

Creates or modifies parameters for specified objects in the volatile database.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

set { **object** *object-name* / **known objects** } [**accept** { **deferred** / **immediate** } | **default user** *login-name* | **file** *file-id* | **number** *number* | **type** { **sequenced packet** / **stream** }]

or

set { **object** *object-name* / **known objects** } **all**

where

accept — Specifies whether incoming connect requests are to be accepted or rejected by the DECnet object spawner or whether they are to be forwarded to the object for processing. Choose one of the following modes:

- **deferred** — The object processes any optional data sent with a connect request and can set optional data to be returned when it accepts or rejects the connect.
- **immediate** — (Default.) The DECnet object spawner determines whether to accept or reject all connect requests.

all — Updates the volatile database with all of the parameters defined for the specified object in the permanent database. Use of **all** precludes use of any other parameters.

default user *login-name* — Specifies a default log-in name under which the object is to run if no access control information is supplied with a connect request and no proxy information is defined locally.

file *file-id* — Specifies an executable file or shell script used to invoke the specified object.

number *number* — Specifies the object number to be identified with the specified object name. The default is 0. See Appendix B for a list of object names and numbers.

known objects — Specifies that parameters are to be created or modified for all known objects.

object *object-name*	Specifies that parameters are to be created or modified for the named object only (a maximum of 16 alphanumeric characters).
type	Identifies the socket type. Choose one of the following socket types:
sequenced packet	(Default) provides a bidirectional, reliable, sequenced, and unduplicated flow of data while preserving record boundaries.
stream	Provides same data flow properties as above without record boundaries.

EXAMPLE

This command defines the Network Management Listener (**nml**) as object number 19 with **/etc/nml** as the executable file. It also specifies that **nml** is to run under the log-in name "guest" if no access control information is supplied with a connect request and no proxy information is defined locally.

```
ncp>set object nml number 19 file /etc/nml default user guest RET
```

show circuit

DESCRIPTION

Displays specified circuit information stored in the volatile database.

SYNTAX

show { **circuit** *circuit-id* | **known circuits** } [**characteristics** | **counters** | **status** | **summary**]

where

circuit *circuit-id*	Displays information for the specified circuit only.
characteristics	Displays parameters that are currently set for the circuit.
counters	Provides counter information for circuits, lines, and nodes.
known circuits	Displays information for all known circuits.
status	Shows information that usually reflects network activity for the running network. Depending on the component, this can include the local node and its operational state, reachable and unreachable nodes, and circuits and lines and their operational states.
summary	(Default) shows abbreviated information provided for the **characteristics** and **status** display types.

EXAMPLE

This command displays circuit error and performance statistics for all known circuits in the volatile database.

```
ncp>show known circuits counters RET

Known Circuit Volatile Counters as of Thur Nov 16 10:30:56 EST 1990

Circuit  =  una-0

          4127  Seconds since last zeroed
          6407  Terminating packets received
         11736  Originating packets sent
             0  Terminating congestions lost
             0  Transmit packets received
             0  Transmit packets sent
             0  Transmit congestion loss
             0  Initialization failure
       9679324  Bytes received
      61282059  Bytes sent
         75268  Data blocks received
         79489  Data blocks sent
             0  User buffer unavailable

ncp>
```

show executor

DESCRIPTION

Displays specified local node information stored in the volatile database.

SYNTAX

show executor [**characteristics** | **counters** | **status** | **summary**]

where

characteristics	Displays parameters that are currently set for the executor.
counters	Provides counter information for circuits, lines, and nodes.
status	Shows information that usually reflects network activity for the running network. Depending on the component, this can include the local node and its operational state, reachable and unreachable nodes, and circuits and lines and their operational states.
summary	(Default) shows abbreviated information provided for the **characteristics** and **status** display types.

EXAMPLE

This command displays local node status information from the volatile database.

```
ncp> show executor status RET

Executor Volatile Status as of Thu Nov 16 10:37:23 EST 1990

Executor node =  2.95 (OHIO)

State                                = On
Physical address                     = aa-04-00-00-53-10
ncp>
```

show line

DESCRIPTION

Displays specified line information stored in the volatile database.

SYNTAX

show { **line** *line-id* | **known lines** } [**characteristics** | **counters** | **status** | **summary**]

where

characteristics	Displays parameters that are currently set for the line.
counters	Provides counter information for circuits, lines, and nodes.
known lines	Displays information for all known lines.
line *line-id*	Displays information for the specified line only.
status	Shows information that usually reflects network activity for the running network. Depending on the component, this can include the local node and its operational state, reachable and unreachable nodes, and circuits and lines and their operational states.
summary	(Default) shows abbreviated information provided for the **characteristics** and **status** display types.

EXAMPLE

This command displays information about line una-0.

```
ncp>show line una-0 summary RET
Line Volatile Summary as of Thu Nov 16 10:40:17 EST 1990
Line                    State
UNA-0                   On
ncp>
```

show logging

DESCRIPTION

Displays specified logging information stored in the volatile database.

SYNTAX

show { **known logging** | **logging console** | **logging file** | **logging monitor** } [**characteristics** | **status** | **summary** | **events**] [**known sinks** | **sink node** *node-id*]

where

characteristics	Displays parameters that are currently set for the executor, line, or circuit.
events	Displays event class and type information for the given logging component.
known logging	Displays information for all known logging components.
known sinks	(Default) displays logging information for all known sink nodes.
logging console	Displays information for the console logging component.
logging file	Displays information for the file logging component.
logging monitor	Displays information for the monitor logging component.
sink node *node-id*	Displays logging information for the specified sink node.
status	Shows information that usually reflects network activity for the running network. Depending on the component, this can include the local node and its operational state, reachable and unreachable nodes, and circuits and lines and their operational states.
summary	(Default) shows abbreviated information provided for the **characteristics** and **status** display types.

EXAMPLE

This command displays event class and type information for the logging file on node N1834P.

```
ncp> show logging file events sink node n1834p RET
Logging Volatile Events as of Thu Nov 16 10:44:04 EST 1990
Logging = file
  No Information
ncp>
```

show node

DESCRIPTION

Displays specified node information stored in the volatile database.

RESTRICTION

No information is displayed for an end node until a link has been established to it. The node may appear to be unreachable even when it is not.

SYNTAX

show { **node** *node-id* | **active nodes** | **known nodes** } [**characteristics** | **counters** | **status** | **summary**]

where

active nodes	Displays information for all nodes that are adjacent, designated routers, or connected to the executor by a logical link.
characteristics	Displays parameters that are currently set for the node.
counters	Provides counter information for circuits, lines, and nodes.
known nodes	Displays information for all known nodes.
node *node-id*	Displays information for the specified node only.
status	Shows information that usually reflects network activity for the running network. Depending on the component, this can include the local node and its operational state, reachable and unreachable nodes, and circuits and lines and their operational states.
summary	(Default) shows abbreviated information provided for the **characteristics** and **status** display types.

EXAMPLE

This command displays error and performance statistics for all known nodes in the volatile database.

```
ncp> show node ohio counters RET

Node Volatile Counters as of Thu Nov 16 10:49:38 EST 1990

Executor node = 2.95 (OHIO)

                0  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
ncp>
```

show object

DESCRIPTION

Displays specified object information stored in the volatile database.

SYNTAX

show { **object** *object-name* | **known objects** } [**characteristics** | **summary**]

where

characteristics	Displays parameters that are currently set for the object.
known objects	Displays information for all known objects.
object *object-name*	Displays information for the specified object only.
summary	(Default) shows abbreviated information provided for the **characteristics** and **status** display types.

EXAMPLE

This command displays information about the Network Management Listener (**nml**).

```
ncp> show object nml RET

Object Volatile Summary as of Thu Nov 16 10:53:46 EST 1990

Object          Number        File

nml               19          /usr/etc/nml
ncp>
```

tell

DESCRIPTION

Sends an **ncp** command to a remote node for execution. **tell** sets the executor only for the command that it prefixes.

NOTE

If you use the **set executor** or **define executor** command to issue a series of commands at a remote node, you can still use the **tell** prefix to issue an **ncp** command to yet another remote node.

SYNTAX

tell *node-id* [*acc-con-info*] *ncp-command*

where

acc-con-info	Specifies access control information required by the remote node.
ncp *command*	Represents any **ncp** command that is remotely executable.
node-id	Specifies the node address or the node name of the remote node.

EXAMPLE

This command sends the **show executor** command to node ART, and tells ART to show characteristics of the executor node.

```
ncp>tell art show exec char RET
```

trigger node

DESCRIPTION

Initiates a down-line load to the specified remote node. Initiates the loading sequence for an unattended system.

NOTE

There is no way to determine the node that performs the down-line load.

RESTRICTIONS

Before you can execute this command:

- You must have superuser privileges.
- The **mop_mom** utility must be installed during the ULTRIX software installation.
- Service must be enabled on the remote node.
- You must have the service password if a DECnet service password is defined on the remote node.

SYNTAX

trigger node *node-id* { **physical address** *E-address* | **[service] password** *hex-password* | **via** *circuit-id* }

where

node *node-id*	Specifies the remote node to be loaded.
physical address *E-address*	Identifies the Ethernet physical address of the remote node. Required for Ethernet circuits if the hardware address parameter has not been specified in the volatile database.
[service] password *hex-password*	Specifies the DECnet service password for maintenance operations such as down-line loading. Specify a hexadecimal value of 16 digits.
via *circuit-id*	Specifies the circuit over which the operation is to take place.

EXAMPLE

This command initiates a down-line load to node BOSTON, whose DECnet service password is `aabb`.

NOTE

Even though node BOSTON initiates the down-line load, there is no way to determine which host will actually perform the down-line load.

```
ncp>trigger node boston password aabb RET
```

trigger via

DESCRIPTION

Initiates a down-line load to the specified remote node through the specified circuit. Initiates the loading sequence for an unattended system through the specified circuit. The circuit identification is obtained from the volatile database on the executor node.

NOTE

There is no way to determine the node that performs the down-line load.

RESTRICTIONS

Before you can execute this command:

- You must have superuser privileges.
- The **mop_mom** utility must be installed during the ULTRIX software installation.
- Service must be enabled on the remote node.
- You must have the service password if a DECnet service password is disabled on the remote node.

SYNTAX

trigger via *circuit-id* { **physical address** *E-address* | **[service] password** *service-password* }

where

via *circuit-id*	Specifies the circuit over which the operation is to take place.
physical address *E-address*	(For Ethernet nodes only.) Identifies the Ethernet physical address that the node currently uses to identify itself. If the circuit is an Ethernet circuit, you must include the physical address in the command.
service password *service-password*	Specifies the remote node's service password.

EXAMPLE

This command initiates a down-line load sequence on the node connected to circuit una-1. The node's service password is ffaa.

```
ncp>trigger via una-1 physical address aa-00-03-00-01 password ffaa RET
```

zero circuit

DESCRIPTION

Sets circuit counters to zero for the specified circuit. The executor node maintains these counters for each circuit.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

zero circuit *circuit-id* [counters]

where

circuit circuit-id — Specifies the circuit for which counters are to be zeroed.

EXAMPLE

This command sets circuit counters to zero for circuit una-0.

```
ncp> zero circuit una-0 RET
```

zero executor

DESCRIPTION

Sets node counters associated with and maintained on the executor node to zero.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

zero executor [counters]

zero line

DESCRIPTION

Sets line counters to zero for the specified line. The executor node maintains these counters for each line.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

zero line *line-id* **[counters]**

where

line *line-id*	Specifies the line for which counters are to be zeroed.

EXAMPLE

This command sets the line counters to zero for line qna-0.

```
ncp> zero line qna-0  RET
```

zero node

DESCRIPTION

Sets node counters to zero for specified nodes in the volatile database. The executor node maintains node counters for each node.

RESTRICTION

You must have superuser privileges to execute this command.

SYNTAX

zero { **node** *node-id* / **known nodes** } **[counters]**

where

known nodes	Zeros counters for all known nodes.
node *node-id*	Zeros counters for the specified node.

EXAMPLE

This command sets the node counters to zero for node BOSTON.

```
ncp>zero node boston RET
```

Chapter 3

Error Messages

This chapter outlines the ncp error message format and lists all messages in alphabetical order, giving a short description of each. When possible, you should correct the error condition and retry the command.

3.1 ncp Error Message Format

The ncp error messages have the following format:

ncp [- Listener response]: *error message*[,*error detail*]
[*extra text* or *command echo*]

where

Listener response	Indicates when the error message is being displayed by the Network Management Listener.
error message	Is the reason for the failure.
error detail	Is a detailed explanation of the failure (for certain error messages.
extra text	Gives an additional system-specific explanation of the error condition.
command echo	Is the command in error (if applicable). An arrow points to the illegal condition.

The following example contains the error message, "Unrecognized keyword," and the command echo, "tell boston sho lines." This message indicates an error in the command line.

```
ncp: Unrecognized keyword
tell boston sho lines
```

The command should read, `"tell boston sho known lines."`

3.2 ncp Error Messages

This section lists the ncp error messages in alphabetical order, with a short description of each message.

Bad loopback response

The message returned in a loopback test does not match the message sent. This error is caused by a loopback protocol violation, bad data return, or bad message length return. Check the integrity of the line. Make sure that you are not on a noisy line.

Component in wrong state

The current operational state of the component precludes the requested operation. The error detail identifies the component type. Check to see that the component is in the correct state to perform the requested operation.

Connect failed

A logical link connect has failed for the reason described in one of the following error details:

Access control rejected

The remote node could not understand or would not accept the access control information. Make sure that you have a valid user name and password on the remote system; then try again.

Insufficient network resources

Either the local node or the remote node had insufficient network resources to create the logical link. You can increase the number of maximum links for DECnet–ULTRIX by using the **ncp set/define executor** command, or reduce the number of logical links in use.

Invalid node name format

The format of the specified node name is invalid. Use 1 to 6 alphanumeric characters, including at least one alphabetic.

Invalid object name format

The remote node did not understand the object name format used by **ncp** to identify the Network Management Listener. Contact the person responsible for your network.

Local node shutting down

The local node is shutting down and will not allow any logical link connections. Wait until your system resumes network activity, then try again to connect.

No response from object

The Network Management Listener did not respond; for example, it may have responded too slowly or terminated abnormally. Contact the person responsible for your network.

Node unreachable

No path exists to the remote node. Make sure that DECnet is installed on the remote node. Check to see that the DECnet circuit is running by using the **ncp show circuit status** command. Check the status of the remote node to see if it is up.

Object too busy

The remote nml object had insufficient resources available to accept the connect request. Contact the person responsible for your network.

Remote node shutting down

The remote node is shutting down and will not accept any logical link connections. Wait until the remote system resumes network activity, then try again to connect.

Unrecognized node name

The destination node name does not correspond to any known node address. Enter a valid node name, which consists of from 1 to 6 alphanumeric characters, including at least one alphabetic character. Check to see if the node name is defined in the DECnet database by using the ncp show node or show known nodes command. If the name is not defined, use the ncp set command to define the node.

Unrecognized object

The remote node does not have a Network Management Listener. Contact the person responsible for your network.

File I/O error

An error was encountered while reading or writing a file necessary to the requested operation. The error detail identifies the database where the error occurred. The file may be corrupted or may not exist. Make sure that the DECnet files copied to your system at installation are in the right directories. See *DECnet–ULTRIX Installation* for a list of the files.

File open error

A file necessary for the requested operation could not be opened. The error detail identifies the database where the error occurred. Make sure that the DECnet files copied to your system at installation are in the right directories. See *DECnet–ULTRIX Installation* for a list of the files.

Hardware failure

The hardware associated with the request could not perform the specified operation. Contact the person responsible for your network, or call your Field Service representative.

Incompatible management version

The Network Management Listener version is incompatible with ncp. You must go to the remote system to execute the command.

Invalid file contents

The requested operation could not be performed because the files contained data of an invalid form or value. The error detail identifies the database where the error occurred. Look at the file to see if it is corrupted. See *DECnet–ULTRIX Installation* for a list of the files.

Invalid identification

The identification of the component specified in the requested operation did not have the proper syntax. For example, a device name that should have the syntax dev-*n*, appears as **unas**. The error detail identifies the component type. Make sure that the component name and unit number, if applicable, are correct and configured into the system.

If this message is received on a **loop circuit** command, the following error detail may be included:

Unable to find device

The device specified by the circuit ID does not exist.

Invalid message format

The information sent by **ncp** to a Network Management Listener was improperly formatted or contained an invalid value. Submit an error report to your Digital Software Services representative.

Invalid parameter grouping

The parameters furnished by the user for the requested operation cannot be included in the same command. Check the appropriate **ncp** command description in this manual. Often, use of the **all** parameter in an **ncp** command precludes the use of any other command parameters.

Invalid parameter value

The value of a parameter furnished by the user for the requested operation was not acceptable (for example, a numeric parameter was out of range). The error detail identifies the type of parameter. Check the appropriate **ncp** command description in this manual, and retry the command using a different parameter value. If this message refers to the **length** parameter in a **loop command**, one of the following error details may be included. In each of these cases, the length was more than could be handled, and the maximum length is included with the error message. Retry the command using a shorter length.

Ethernet message size exceeded

The length specified for blocks to be looped exceeds the maximum allowed. (See the **loop** command descriptions in this manual for details.)

Looper size exceeded

The requested length exceeds the buffering capability of the active looper task.

Mirror size exceeded

The requested length exceeds the buffering capability of the network management loopback mirror.

Line communication error

The requested operation failed because of communication errors on the involved line. Make sure that your system is properly attached to the network and that the other node is up and running.

Line protocol error

The requested operation failed because of protocol errors on the involved line. This condition usually implies either incompatible line protocols or protocol-programming errors. It can also be caused by a line hardware error that was not detected by the line protocol. (Line protocol can mean either the Data Link Protocol or the Service Operation Protocol.)

Management program error

The network management software has detected an internal error. Contact the person responsible for your network and/or submit a report to your Digital Software Services representative.

If this error occurs on a loop command, one of the following error details may be provided:

Bad data pattern developed

The software is unable to build a message because of a program error.

Incorrect optional data size on accept

The optional accept data from mir was improperly formatted.

Unable to get physical address from device

A request for a device's Ethernet physical address failed.

Mirror connect failed

The logical link to the network management loopback mirror could not be connected. This error message usually has one of the following error details:

Abort by management

The connection was aborted by a third party, not by the user programs at either end of the connection. Contact the person responsible for your network.

Abort by object

A programming error in the network management loopback mirror caused it to abort the logical link. Submit an error report to your Digital Software Services representative.

Access control rejected

Either the remote node or the network management loopback mirror could not understand or would not accept the access control information. Make sure you have a valid user name and password on the remote node; then try again.

Connection rejected by object

The logical link could not be connected because the network management loopback mirror rejected the connection. This condition usually implies that the loopback mirror is too busy to accept another logical link. Try to connect later.

Disconnect by object

A programming error in the network management loopback mirror caused it to disconnect the logical link. Submit an error report to your Digital Software Services representative.

Insufficient network resources

Either the local node or the remote node had insufficient network resources to connect the logical link. You can increase the number of maximum links for DECnet–ULTRIX by using the **ncp set executor** and **define executor** commands, or reduce the number of links in use.

Local node shutting down

The executor node is in the process of shutting down and is not accepting any more logical link connections. Wait until your system resumes network activity, and then try again to connect.

No response from object

The network management loopback mirror did not respond; for example, it may have responded too slowly or terminated abnormally. Contact the person responsible for your network.

Node unreachable

No path exists to the remote node. Make sure that DECnet is installed on the remote node. Check to see that the DECnet circuit is running by using the **ncp show circuit status** command. Check the status of the remote node to see if it is up.

Object too busy

The remote **nml** object had insufficient resources available to accept the connect request. Contact the person responsible for your network.

Remote node shutting down

The remote node is shutting down and will not accept any logical link connections. Wait until the remote node resumes network activity, then try again to connect.

Unrecognized node name

The destination node name does not correspond to any known node address. Enter a valid node name, which consists of from 1 to 6 alphanumeric characters, including at least one alphabetic character. If the name is not defined in the DECnet database, use the **ncp set** command to define the node.

Unrecognized object

The remote node does not have a network management loopback mirror. Contact the person responsible for your network.

Mirror link disconnected

The logical link from ncp to the network management loopback listener was unexpectedly disconnected. This error message usually has one of the error details listed under "Mirror connect failed." See the error detail for a description of the error and a recommended action.

No room for new entry

The requested operation could not be performed because it required the addition of a new entry in a database that was already full. Contact the person responsible for the remote system.

Operation failure

The requested operation failed. If an error detail is not provided, see the network documentation for the remote system.

Oversized management command message

The ncp command message was too big to be received by the Network Management Listener. Submit an error report to your Digital Software Services representative.

Oversized management response

The message returned by the Network Management Listener was too big to be received by ncp. Submit an error report to your Digital Software Services representative.

Parameter missing

A necessary parameter for the requested operation was omitted. The error detail identifies the type of parameter. Check the appropriate ncp command description in this manual and reenter the command including the necessary parameter.

Parameter not applicable

The user supplied a parameter that is not applicable to the requested operation on the specified component. The error detail identifies the type of parameter. Check the appropriate ncp command description and reenter the command minus the problem parameter.

Parameter value out of range

A numeric parameter value is outside the allowable range. Identify the parameter by referring to the appropriate ncp command description; then reenter the command.

Parameter value too long

The parameter value was too long to be accepted by the Network Management Listener. The error detail identifies the type of parameter. Shorten the parameter value, and then reenter the command.

Privilege violation

The user does not have sufficient privilege to perform the requested operation. Log in as superuser and reenter the command, or contact the person responsible for your network.

Redundant parameter

A parameter has been entered twice in the same command. Eliminate the extra parameter and reenter the command.

Resource error

Network management had insufficient internal resources to perform the requested operation.

On a **loop** command, the following error detail may also be provided:

Unable to check loopback state

The software could not open a socket to see whether the device is in loopback state. Check to see that **dli** is configured into the system.

System-specific management function not supported

The requested operation is ULTRIX system-specific and is not supported by the Network Management Listener. You must perform this operation on the remote node.

Unrecognized command

The Network Control Program (**ncp**) does not support the command entered by the user. Refer to the **ncp** command descriptions in this manual.

Unrecognized component

The component specified by the user does not exist. The error detail identifies the component type. Make sure that the DECnet files copied to your system at installation are in the right directories. See *DECnet–ULTRIX Installation* for a list of the files.

Unrecognized function or option

The requested operation is not implemented by the executor. Refer to the **ncp** command descriptions in this manual.

Unrecognized keyword

One of the keywords in a command is unknown to **ncp**. The command echo flags the unrecognized keyword. Refer to the **ncp** command descriptions in this manual.

Unrecognized parameter type

The command parameter identified in the error detail is not implemented by the executor. Refer to the **ncp** command descriptions in this manual.

Unrecognized value

A parameter value specified by the user is unknown to **ncp**. The command echo flags the faulty parameter value. Refer to the relevant **ncp** command description in this manual.

Chapter 4

Event Messages

The DECnet–ULTRIX Event Logger (evl) is a network management tool that records network activity. The Event Logger can record two categories of event messages: event classes and event types. Event classes relate to specific layers of the DECnet architecture, while event types relate to specific events within an event class.

4.1 Event Classes

DECnet logging events fall into the event classes listed in Table 4-1. Events not logged by DECnet-ULTRIX may be logged by other remote nodes. Check the documentation for the remote system for details.

Table 4–1: Event Classes

Event Class	Description
0	Network Management layer
1[1]	Applications layer
2	Session Control layer
3	End Communications layer
4	Routing layer
5[1]	Data Link layer
6[1]	Physical Link layer
7-479[1]	Reserved for other DECnet products
480-511	Reserved for customer use

[1]Event class not logged by DECnet-ULTRIX.

4.2 Event Message Format

Event messages have the following format:

Event type *class.type*, [*event-text*]

Occurred *dd-mon-yy hh:mm:ss.s* on node *address* [(*node-name*)]

[*component-type component-name*]

[*data*]

where

class	Is the class in which the event occurred (see Table D-1.)
type	Is the specific event type number within that class.
event-text	Is the text describing the event (see Sections D.3 through D.6.)
dd-mon-yy	Is the date (day, month, and year) on which the event occurred.
hh:mm:ss.s	Is the time (hour, minutes, and seconds) at which the event occurred.
address	Is the address of the node at which the event occurred.
node-name	Is the name of the node at which the event occurred.
component-type	Is either line, circuit, or node. If an event is not associated with a particular component, this line is not present.
component-name	Is the name of the component that caused the event.
data	Is event-dependent text that gives more information about the event. Often this text includes the name or address for the component type for which the event applies. It may also provide additional information about the cause of the event.

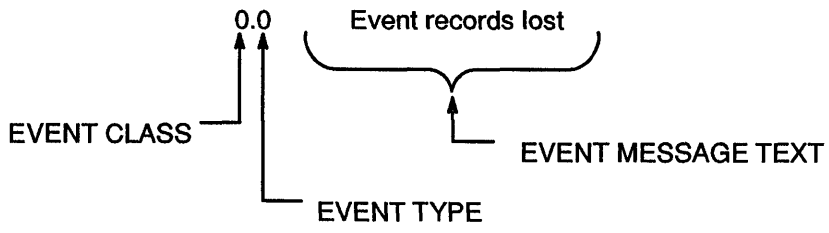
EXAMPLE:

This example shows an event-logging message indicating that an adjacent node is unavailable.

```
Event type 4.18, Adjacency down
Occurred 21-May-87 08:17:11.0 on node 19.12 (PITSBG)
Circuit UNA-0
Adjacency listener receive timeout
Adjacent node = 19.160
```

The following sections list DECnet–ULTRIX event messages by class and type for each layer. Figure D-1 shows the format in which the messages appear:

Figure 4–1: Event Message Format



4.3 Network Management Layer Events

Event	Message
0.0	**Event records lost** Events occurred too rapidly for the event logger to buffer them. This message does not display any event qualifiers.
0.6	**Passive loopback** The software initiated or terminated a passive loopback test on behalf of an adjacent node. This message displays the circuit name to which the event applies and the state of operation qualifier (initiated or terminated).

4.4 Session Control Layer Events

Event	Message
2.0	**Local node state change** The operational state of the local node changed because of an operator command. Note that the transition from **shut** to **off** also happens automatically when the last logical link is disconnected (under normal operation). This message displays the reason for the state change (operator command or normal operation), the old state (**on**, **off**, **shut**, or **restricted**), and the new state (**on**, **off**, **shut**, or **restricted**).
2.1	**Access control failure** The local node rejected a connect request because of invalid access control information. This message displays the name and address of the source node, the object type number and process ID of the source process requesting the connection, the object type number and process ID of the destination process to receive the connect request, and the invalid access control information (user ID, password, and account information).

4.5 End Communications Layer Event

Event	Message
3.2	**Node data base reused** The local node received a connect request from or tried to initiate an outgoing connect to a node for which there is no counter block. All counter blocks have been used, and one of the previously used blocks is available for this new node. This results in the loss of node counters for the node that formerly occupied the database entry. This message displays the address and name of the node for which the database entry was formerly used and the counters for that node.

4.6 Routing Layer Events

Event	Message
4.3	**Oversized packet loss** A packet has been discarded because it was too large to forward to the appropriate adjacent node. Normally, this condition occurs because the adjacent node's buffer is too small or the source node sent a packet that was too large. The latter condition can be handled by setting a smaller segment size at the source node. This message displays the packet header and the name of the circuit over which the packet was to be forwarded. For contents of the packet header, refer to the *DECnet Digital Network Architecture (Phase IV), Network Management Functional Specification.*
4.4	**Packet format error** A packet has been discarded because of a format error in the packet header. Usually, this results from a programming error in packet formatting by the adjacent node. It could also result from a circuit error not detected by circuit protocol. This message displays the first six bytes of the packet (in hexadecimal) and the name of the circuit to which the event applies. For a point-to-point line this message also displays the adjacent node.
4.6	**Verification reject** This message is displayed for point-to-point lines only. It displays the first six bytes of the packet header (in hexadecimal), the name of the circuit, and the adjacent node to which the packet applies. An invalid verification message was received. The password from the remote node does not match the 'circuit receive password' set up in the database.
4.8	**Circuit down** Under normal operating conditions, this event occurs when you set the circuit or executor to the **off** state. It also occurs if there is a hardware problem with the line or device. For point-to-point lines, this event can occur if the node listen timer expires or invalid data is received in the hello message. This message displays the name of the circuit to which the event applies. For point-to-point lines, this message also displays the adjacent node name. In the case of a hardware error, it also displays the following message: **Adjacent node listener received invalid data**
4.9	**Circuit down — operator initiated** This event is logged when the node identification in the hello message from the remote node is not the expected one. This message displays the name of the circuit and adjacent node to which the event applies. This message is displayed for point-to-point lines only.
4.10	**Circuit up** The basic initialization of a remote node has been completed by the device and transceiver. This message displays the name of the circuit to which the event applies, as well as the name and address of the newly initialized node. For a point-to-point line this message also displays the adjacent node.
4.11	**Initialization failure — circuit fault** This event is logged when a verification message is not received within the timeout period. This message is displayed for point-to-point lines only. It displays the circuit, adjacent node, and reason for failure.

Event	Message
4.12	**Initialization failure — software** This message is logged when the local node is unable to send the verification message requested by the remote node due to lack of system resources. This message is displayed for point-to-point lines only. It displays the circuit, adjacent node, packet header, and reason for failure.
4.13	**Initialization failure — operator initiated** The routing layer logs this message when it detects an area mismatch or a version skew during the circuit initialization sequence. This message is displayed for point-to-point lines only. It displays the circuit, adjacent node, packet header, and reason for failure.
4.15	**Adjacency up** For broadcast circuits (UNA and QNA), initialization has occurred with another node on the Ethernet. End nodes log this message for only one node. This message displays the adjacent node number.
4.18	**Adjacency down** The remote node has recycled. This event could result from a remote node restart or an invalid protocol message. The message displays the reason why the adjacent node is down.

4.7 Event Log Summary

Table 4-2 summarizes the events logged by the event logger:

Table 4–2: Event Log Summary

Class	Type	Entity[1]	Standard text	Counters
0	0	none	Event records lost	none
0	1	node	Automatic node counters	Node counters
0	5	node	Node counters zeroed	Node counters
0	8	any	Automatic counters	Counters
0	9	any	Counters zeroed	Counters
2	1	none	Access control reject	Source node, Source process, Destination process, User, Password, Account
3	0	none	Invalid message	Message, Source node
3	1	none	Invalid flow control	Message, Source node, Current flow control
3	2	node	Data base reused	NSP node counters
4	0	none	Aged packet loss	Packet header
4	1	circuit	Node unreachable, packet loss	Packet header, Adjacent node

[1]In this context, entity refers to a component or software module that can generate events.

(continued on next page)

Table 4–2 (Cont.): Event Log Summary

Class	Type	Entity[1]	Standard text	Counters
4	2	circuit	Node out-of-range, packet loss	Packet header, Adjacent node
4	3	circuit	Oversized packet loss	Packet header, Adjacent node
4	4	circuit	Packet format error	Packet beginning, Adjacent node
4	5	circuit	Partial routing update loss	Packet header, Highest address, Adjacent node
4	6	circuit	Verification reject	Node
4	7	circuit	Circuit down, circuit fault	Reason, Adjacent node
4	8	circuit	Circuit down	Reason,Packet header, Adjacent node
4	9	circuit	Circuit down, operator initiated	Reason, Packet header, Adjacent node
4	1	circuit	Circuit up	Adjacent node
4	1	circuit	Initialization failure, line fault	Reason
4	1	circuit	Initialization failure, operator fault, Reason, Packet header, Received version	Reason
4	1	node	Node reachability change	Status
4	1	circuit	Adjacency up	Adjacent node
4	1	circuit	Adjacency rejected	Adjacent node, Reason
4	1	area	Area reachability change	Status
4	1	circuit	Adjacency down	Reason, Packet header, Adjacent node
4	1	circuit	Adjacency down, operator initiated, Reason, Packet header	Adjacent node
5	0	circuit	Locally initiated state change	Old state, New state
5	1	circuit	Remotely initiated state change	Old state, New state
5	2	circuit	Protocol restart received in maintenance mode	None
5	3	circuit	Send error threshold	Circuit counters
5	5	circuit	Select error threshold	Circuit counters
5	6	circuit	Block header format error	Header (optional)
5	1	line	Send failed	Failure reason, Distance

[1]In this context, entity refers to a component or software module that can generate events.

(continued on next page)

Table 4–2 (Cont.): Event Log Summary

Class	Type	Entity[1]	Standard text	Counters
5	1	line	Receive failed	Failure reason, Ethernet header
5	1	line	Collision detect check failed	none

[1]In this context, entity refers to a component or software module that can generate events.

Chapter 5

Network Counters

The network software maintains counters for circuits, lines, and nodes. This chapter lists all counters in alphabetical order within component groups. In some cases, the counters respond to and reflect network events. In other cases, the counters respond to and reflect normal activities such as messages sent and messages received. Individual counter descriptions indicate whether that counter increments when a corresponding event occurs (see Chapter 4 for a description of event messages). Where possible, the description includes reasons why each of the counters might be incremented.

A counter does not display a value greater than its maximum value. When a counter overflows, it locks on the overflow value until it is zeroed. Counter displays with an angle bracket (>) indicate that the counter has overflowed. For example, if the maximum value for a counter is 255, its overflow display is >255. Each of the counter descriptions in this appendix includes the maximum value of the counter.

Each category of counters maintains a timing counter (seconds since last zeroed) that is zeroed when its associated counters are zeroed and starts when they start. In this way, the timing counter logs the seconds since its associated counters were zeroed to provide a time frame for them.

5.1 Circuit Counters

Circuit counters for DECnet–ULTRIX are maintained in the Network Management layer, the Routing layer, and the Data Link layer.

5.1.1 Network Management Layer

Seconds since last zeroed

This counter is zeroed when the other circuit counters are zeroed. It then increments by 1 every second so as to provide a time frame for the other circuit counters. The overflow value is 65,535.

5.1.2 Routing Layer

Circuit down

This counter records the number of times that a circuit was declared down by the executor. The overflow value is 255.

Initialization failure

This counter increments when the circuit could not be initialized by the executor for network use. The overflow value is 255.

Originating packets sent

This counter records the number of packets sent by the executor over the circuit. The overflow value is 4,294,967,295.

Terminating congestion loss

This counter records the number of packets intended for the node that were discarded because the Routing layer could not buffer them. The overflow value is 65,535.

Terminating packets received

This counter records the number of packets received by the executor with the executor as the destination. The overflow value is 4,294,967,295.

Transit congestion loss

This counter records the number of packets received by the executor that were to be routed to another node but were discarded because of heavy traffic on the output circuit. The overflow value is 65,535.

Transit packets received

This counter increments when the executor receives a packet that is to be routed to another node. The overflow value is 4,294,967,295.

Transit packets sent

This counter increments when the executor sends a packet through to another node. The overflow value is 4,294,967,295.

5.1.3 Data Link Layer

Bytes received

This counter increments when a data byte is received on the circuit. The count does not include Data Link Protocol overhead or bytes retransmitted by the Data Link layer. It can be used with the Data blocks received counter to determine the inbound traffic load on the circuit. The overflow value is 4,294,967,295.

Bytes sent

This counter increments when a data byte is sent on the circuit. The count does not include Data Link Protocol overhead or bytes retransmitted by the Data Link layer. It can be used with the **Data blocks sent** counter to determine the outbound traffic load on the circuit. The overflow value is 4,294,967,295.

Data blocks received

This counter increments when a data block is received on the circuit. The count does not include Data Link Protocol overhead. This counter can be used as a statistical base for evaluating the other Data Link layer counters. The overflow value is 4,294,967,295.

Data blocks sent

This counter increments when a data block is sent on the circuit. The count does not include Data Link Protocol overhead or blocks retransmitted by the Data Link layer. It can be used as a statistical base for evaluating the other Data Link layer counters. The overflow value is 4,294,967,295.

Data errors inbound

This counter indicates the number of incoming data errors on the circuit. It can have either of the following qualifiers: negative acknowledgments (NAKs) sent, data field block check error NAKs sent, reply response. The overflow is 255.

Data errors outbound

This counter indicates the number of outgoing data errors on the circuit. It can have any of the following qualifiers: NAKs received, header block check error NAKs received, data field block check error NAKs received, reply response. The overflow is 255.

Local buffer errors

This counter increments when a negative acknowledgment (NAK) is sent. It can have either of the following qualifiers: NAKs sent, buffer unavailable; NAKs sent, buffer too small. The overflow is 255.

Local reply timeouts

This counter increments each time a message is retransmitted because the retry timer for a sent message expired before a positive acknowledgment (ACK) was received from the remote node. The overflow value is 255.

Remote buffer error

This counter increments when a negative acknowledgment (NAK) is received. It can have either of the following qualifiers: NAKs received, buffer unavailable; NAKs received, buffer too small. The overflow value is 255.

Remote reply timeouts

This counter increments each time a message is retransmitted because the retry timer for a sent message expired before a positive acknowledgment (ACK) from your node was received at the remote node. The overflow value is 255.

Selection intervals elapsed

This counter records the number of times that the executor turned a circuit around or selected an adjacent node on both half-duplex and multipoint circuits. This counter is used as a statistical base for the evaluation of the counter for selection timeouts. The overflow value is 65,535.

Selection timeouts

This counter records the number of times that the executor turned a circuit around or selected an adjacent node on both half-duplex and multipoint circuits but the adjacent node failed to respond within the required time. This can be caused by blocks being lost on the circuit in either direction or by too small a value being specified for the executor's response timer. Blocks are usually lost because of a partial, temporary, or total failure of the communications line. This counter can have the following qualifier: No reply to select. The overflow value is 255.

User buffer unavailable

This counter indicates the total number of times that a user buffer was not available for an incoming frame that passed all filtering. User buffers are supplied by users on receive requests. The overflow value is 65,535.

5.2 Line Counters

Line counters for DECnet–ULTRIX are maintained in the Network Management layer and the Data Link layer.

5.2.1 Network Management Layer

Seconds since last zeroed

This counter is zeroed when the other line counters are zeroed. It then increments by 1 every second so as to provide a time frame for the other line counters. The overflow value is 65,535.

5.2.2 Data Link Layer

Blocks sent, initially deferred

This counter indicates the total number of times that a frame transmission was deferred on its first transmission attempt. It is used to measure Ethernet contention with no collisions. The overflow value is 4,294,967,295.

Blocks sent, multiple collisions

This counter indicates the total number of times that a frame was successfully transmitted on the third or later attempt after normal collisions had occurred on previous attempts. The overflow value is 4,294,967,295.

Blocks sent, single collision

This counter indicates the total number of times that a frame was successfully transmitted on the second attempt after a normal collision had occurred on the first attempt. The overflow value is 4,294,967,295.

Bytes received

This counter increments when a data byte is received on the line. The count does not include Data Link Protocol overhead or bytes retransmitted by the Data Link layer. It can be used with the **Data blocks received** counter to determine the inbound traffic load on the line. The overflow value is 4,294,967,295.

Bytes sent

This counter increments when a data byte is sent on the line. The count does not include Data Link Protocol overhead or bytes retransmitted by the Data Link layer. It can be used with the **Data blocks sent** counter to determine the outbound traffic load on the line. The overflow value is 4,294,967,295.

Collision detect check failure

This counter indicates the approximate number of times that a collision detect was not sensed after a transmission. The overflow value is 65,535.

Blocks received

This counter increments when a data block is received on the line. The count does not include Data Link Protocol overhead. It can be used as a statistical base for evaluating the other Data Link layer counters. The overflow value is 4,294,967,295.

Data blocks sent

This counter increments when a data block is sent on the line. The count does not include Data Link Protocol overhead or blocks retransmitted by the Data Link layer. It can be used as a statistical base for evaluating the other Data Link layer counters. The overflow value is 4,294,967,295.

Data overrun

This counter indicates the total number of times that the hardware lost an incoming frame because it was unable to keep up with the data rate. The overflow value is 65,535.

Local station errors

This counter records occurrences caused by a fault in a remote station or by an undetected error on the channel inbound to this station. The overflow value is 255. When this counter has a nonzero value, the type of failure is listed:

Local receive overrun (LOVRN)

The local station experienced a receive overrun and sent out a negative acknowledgment (NAK).

Local receive overrun (LOVR)

The local station experienced a receive overrun but did not send out a NAK.

Local transmit underruns (LUNDR)

The local station experienced a transmit underrun.

Local message header format error (LMHFE)

The local station sent a packet with a bad header for which the remote station sent out a NAK.

Multicast blocks received

This counter indicates the total number of multicast blocks that have been successfully received. The overflow value is 4,294,967,295.

Multicast bytes received

This counter indicates the total number of multicast data bytes that have been successfully received (including bytes in the Ethernet data field but not the Ethernet data link headers). The overflow value is 4,294,967,295.

Receive failure

This counter indicates the total number of blocks received with some data error. (The blocks are data frames that passed either physical or multicast address comparison.) The overflow value is 65,535. When this counter has a nonzero value, the type of failure that has occurred is also listed:

Block check error

The frame failed the cyclic redundancy check (CRC). The problem can be with either the local node or the remote node. The failure can be caused by electromagnetic interference, late collisions, or a faulty hardware controller. Use both circuit-level loopback tests to see whether your node is working correctly.

Framing error

The frame did not contain an integral number of 8-bit bytes. The problem can be with either the local node or the remote node. The failure can be caused by electromagnetic interference, late collisions, or a faulty hardware controller. Use both circuit-level loopback tests to see whether your node is working correctly.

Frame too long

The frame was discarded because it was either longer than the maximum or shorter than the minimum allowable length for the Ethernet. The remote node is sending frame lengths that do not meet the requirements of the Ethernet specification. The problem could be with the DEUNA/DEQNA, the H4000 transceiver, the transceiver cable, the DELNI, or the transmitting node.

Remote station error

This counter records occurrences caused by a fault in a remote station or by an undetected data error on the channel inbound to this station. The overflow value is 255. When this counter has a nonzero value, the type of failure that has occurred is listed:

Remote receive overrun (ROVRN)

The remote station experienced a receive overrun and sent out a negative acknowledgment (NAK).

Remote message header format errors (RMHFE)

The remote station sent a packet with a bad header for which the local station sent out a NAK.

Remote streaming tributaries (RSTR)

The remote station failed to release the channel at the end of the selection interval, or the maximum transmission interval (different for each implementation) is exceeded without releasing the channel.

Send failure

This counter usually indicates the total number of times that a transmit attempt failed. However, this counter can also increment on successful transmissions when a DECOM (broadband transceiver) and DEQNA controller are used together. The overflow value is 65,535. When this counter indicates a failure, the type of failure that has occurred is also listed:

Carrier check failed

The data link did not sense a signal that must accompany transmission of a frame. This condition indicates a failure during transmission because of a problem with the DEUNA/DEQNA, the H4000 transceiver, or the transceiver cable.

Excessive collisions

The maximum number of retransmissions resulting from collisions during transmissions has been exceeded. Transmissions are failing because frames being sent are colliding with frames being transmitted by other nodes. This condition can occur if the network is overloaded or if there is a hardware problem.

Frame too long

Either the DEUNA/DEQNA controller or the transceiver truncated the frame at the maximum buffer size. Either your node tried to send a frame that was too long, or the transceiver cut off the message too soon.

Open circuit

There is a break somewhere along the communications path. If this problem exists on your node only, it is probably a fault with the DEUNA/DEQNA option module, the H4000 transceiver, or the DELNI. In this case, use the loopback tests to isolate the problem. If other nodes report the same problem, the fault is probably with an H4000 transceiver connection, a DELNI connection, or the Ethernet cable itself.

Remote failure to defer

A remote node began transmitting while your node was still actively transmitting. Either there is a problem with the remote node's carrier sense, or there is a weak transmitter on your node. Use the loopback test with transmit assistance to determine whether the remote node can detect a transmission from the assistant node.

Short circuit

There is a short circuit in either the Ethernet cable, the H4000 transceiver, the DELNI, or the DEUNA/DEQNA option module. If this problem exists on your node only, it is probably a faulty DEUNA/DEQNA. In this case, use the loopback tests to isolate the problem.

System buffer unavailable

This counter indicates the total number of times that no system buffer was available for an incoming frame. This can apply to any buffer between the hardware and the user buffers (those supplied on receive requests). The overflow value is 65,535.

Unrecognized frame destination

This counter indicates the number of times that a frame was discarded because there was no enabled portal with the protocol type or multicast address. The count includes frames received for the physical address, broadcast address, or multicast address. The overflow value is 65,535.

User buffer unavailable

This counter indicates the number of times no user buffer was available for an incoming frame that passed all filtering. The user buffer is one supplied by the user on a receive request.

5.3 Node Counters

Node counters for DECnet–ULTRIX are maintained in the Network Management layer and the End Communications layer. Additional counters are kept for the executor node.

5.3.1 Network Management Layer

Seconds since last zeroed

This counter is zeroed when the other node counters are zeroed. It then increments by 1 every second so as to provide a time frame for the other node counters. The overflow value is 65,535.

5.3.2 End Communications Layer

Buffer unavailable

This counter indicates the total number of data segments discarded due to insufficient cache buffering. The overflow value is 65,535.

Connects received

This counter increments when a connect initiation signal is received from the associated node. The overflow value is 65,535.

Connects sent

This counter increments when a connect initiation signal is sent to the associated node. The overflow value is 65,535.

Response timeouts

This counter increments when the associated node fails to respond within the required time. This situation 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 condition normally indicates an overload condition in the network. This should be considered a problem if 2 percent or more of the messages sent are timed out. The overflow value is 65,535.

Total bytes received

This counter increments when bytes are received from the associated node at the logical link level. The count includes bytes from both user messages and logical link protocol messages. The overflow value is 4,294,967,295.

Total bytes sent

This counter increments when bytes are sent to the associated node at the logical link level. The count includes bytes from both user messages and logical link protocol control messages. The overflow value is 4,294,967,295.

Total messages received

This counter increments when a message is received from the associated node at the logical link level. The count includes both user messages and logical link protocol control messages. Furthermore, it includes internal segmentation of user messages by the Network Services layer. The overflow value is 4,294,967,295.

Total messages sent

This counter increments when a message is sent to the associated node at the logical link level. The count includes both user messages and logical link protocol control messages. It also includes the retransmission of a message. The Network Services layer segments the user messages. The overflow value is 4,294,967,295.

User bytes received

This counter increments when user data bytes are received from the associated node at the logical link level. It includes only the user data from data messages and from interrupt, connect, accept, reject, disconnect, and abort functions. The overflow value is 4,294,967,295.

User bytes sent

This counter increments when user data bytes are sent to the associated node at the logical link level. It includes only the acknowledged user data from data messages and from interrupt, connect, accept, reject, disconnect, and abort functions. It does not include retransmissions. The overflow value is 4,294,967,295.

User messages received

This counter increments when a user message is received from the associated node at the logical link level. The overflow value is 4,294,967,295.

User messages sent

This counter increments when a user message is sent to the associated node at the logical link level. The overflow value is 4,294,967,295.

5.3.3 Executor Node Counters

Aged packet loss

This counter increments when a packet is discarded because it has visited too many nodes. The count is the total of all such discards by the executor node. This counter is incremented each time the aged packet loss event occurs. The overflow value is 255.

Node out-of-range packet loss

This counter increments when a packet is discarded because the destination node address was greater than the maximum address defined for the executor. The count is the total of all such discards by the executor node. This counter is incremented each time the node out-of-range packet loss event occurs. The overflow value is 255.

Node unreachable packet loss

This counter increments when a packet is discarded because its destination node was unreachable. The count is the total of all such discards by the executor node. The counter is incremented each time the node unreachable packet loss event occurs. The overflow value is 65,535.

Oversized packet loss

This counter increments when a packet is discarded because it was larger than the circuit buffer size. The circuit buffer size was previously established between the executor node and the adjacent node. The counter is incremented each time the oversized packet loss event occurs. The overflow value is 255.

Packet format error

This counter increments when a packet is discarded because of invalid packet control information. The count is the total of all such discards by the executor node. This counter is incremented each time the packet format error event occurs. The overflow value is 255.

Partial routing update loss

This counter increments when part of a routing update is lost because it contained a reachable node address that exceeded the maximum address defined for the executor node. The count is the total of all such occurrences at the executor node. Only routing nodes keep this counter. This counter is incremented each time the partial routing update loss event occurs. The overflow value is 255.

Peak logical links active

This counter records the number of active logical links between the executor and all nodes (including itself). The overflow value is 65,535.

Verification reject

This counter increments when the executor rejects a verification request from an adjacent node during routing initialization. The count is the total of all such occurrences at the executor node. This counter is incremented each time the verification reject event occurs. The overflow value is 255.

Appendix A

Command Summary

This appendix summarizes the **ncp** commands supported by DECnet–ULTRIX.

You may wish to review the graphic conventions listed in the Preface, especially the use of braces { }, brackets [ ], and parentheses ( ). This summary also uses the following notations:

* = Command cannot be used with the **tell** prefix.

S = Command can be used by someone with superuser privileges only.

A = Command can be used by all users.

S **clear circuit** *circuit-id* [**receive password** | **transmit password**]

S **clear executor** { **identification** | **incoming timer** | **outgoing timer** }

A* **clear executor node**

S **clear** { **known logging** | **logging console** | **logging file** | **logging monitor** } [**name** | **events** *event-list* | **known events** [**circuit** *circuit-id* | **line** *line-id* | **node** *node-id*] [**sink** { **executor** | **node** *node-id* }]]

S clear { node *node-id* / known nodes } { diagnostic file / dump file / hardware address / host / load file / name / secondary loader / service circuit / service password / tertiary loader }

or

clear { node *node-id* / known nodes } all

S clear { object *object-name* / known objects }

S define circuit *circuit-id* { hello timer *seconds* / receive password *password* / service { disable / enable } / state *circuit-state* { off / on } / transmit password *password* }

S define executor { address *node-address* / delay factor *number* / delay weight *number* / gateway access { disable / enable } / gateway user *login-name* / identification *id-string* / inactivity timer *seconds* / incoming proxy { disable / enable } / incoming timer *seconds* / maximum links *number* / maximum node counters *number* / name *node-name* / outgoing proxy { disable / enable } / outgoing timer *seconds* / pipeline quota *number* / retransmit factor *number* / segment buffer size *number* }

S define line *line-id* { controller { loopback | normal } | duplex { full | half } | protocol { ddcmp dmc | ddcmp point | ethernet } }

S define { known logging | logging console | logging file | logging monitor } [name *name* | state { off | on } | [events *list* | known events [[circuit *circuit-id* | line *line-id* | node *node-id*]] [[sink { executor | node *node-id* }]]]]

S define node *node-id* { address *node-address* | diagnostic file *file* | dump file *file* | hardware address *E-address* | host *node-id* | load file *file* | name *node-name* | secondary [loader] *file* | service { disable | enable } | service circuit *circuit-id* | [service] password *service-password* | tertiary [loader] *file* }

S define { object *object-name* | known objects } { accept { deferred | immediate } | default user *login-name* | file *file-id* | number *number* | type { sequenced packet | stream } }

A list { circuit *circuit-id* | known circuits } [characteristics | status | summary]

A list executor [characteristics | status | summary]

A **list** { **line** *line-id* / **known lines** } [**characteristics** / **status** / **summary**]

A **list** { **known logging** / **logging console** / **logging file** / **logging monitor** } [**characteristics** / **status** / **summary** / **events**] [**known sinks** / **sink node** *node-id*]

A **list** { **node** *node-id* / **known nodes** } [**characteristics** / **status** / **summary**]

A **list** { **object** *object-name* / **known objects** } [**characteristics** / **summary**]

S **load node** *node-id* { **address** *node-address* / **from** *load-file* / **host** *node-id* / **name** *node-name* / **physical address** *E-address* / **secondary [loader]** *file* / **[service] password** *service-password* / **tertiary [loader]** *file* / **via** *circuit-id* }

S **load via** *circuit-id* { **address** *address* / **from** *load-file* / **host** *node-id* / **name** *node-name* / **physical address** *E-address* / **secondary [loader]** *file* / **[service] password** *service-password* / **tertiary [loader]** *file* }

S

loop circuit [**node** *node-name* | **physical address** *E-address* | [**help** | { **full** | **receive** | **transmit** } { **node** *node-name* | **assistant node** *node-name* | **physical address** *E-address* | **assistant physical address** *E-address* }]]

[**count** *count* | **length** *length* | **with** { **mixed** | **ones** | **zeros** }]

S **loop line** *line-id* { **count** | **length** | **with** { **mixed** | **ones** | **zeros** } }

A **loop executor** [**count** *count* | **length** *length* | **with** { **mixed** | **ones** | **zeros** }]

A **loop** { **node** *node-id*[*acc-con-info*] } [**count** *count* | **length** *length* | **with** { **mixed** | **ones** | **zeros** }]

S **purge circuit** *circuit-id* [**receive password** | **transmit password**]

S **purge executor** { **identification** | **incoming timer** | **outgoing timer** }

S purge { known logging / logging console / logging file / logging monitor } [name / events *event-list* / known events [circuit *circuit-id* / line *line-id* / node *node-id*] / [sink { executor / node *node-id* }]]

S purge { node *node-id* / known nodes } { diagnostic file / dump file / hardware address / host / load file / name / secondary loader / service circuit / service password / tertiary loader }

or

purge { node *node-id* / known nodes } all

S purge { object *object-name* / known objects }

S set circuit *circuit-id* { hello timer *seconds* / receive password *password* / state { off / on } / transmit password *password* }

or

set circuit *circuit-id* all

S set executor { address *node-address* | delay factor *number* | delay weight *number* | gateway access { disable | enable } | gateway user *login-name* | identification *id-string* | inactivity timer *seconds* | incoming proxy { disable | enable } | incoming timer *seconds* | maximum links *number* | maximum node counters *number* | name *node-name* | outgoing proxy { disable | enable } | outgoing timer *seconds* | pipeline quota *number* | retransmit factor *number* | segment buffer size *number* | state { on | off | shut | restricted } }

or

set executor all

A* set executor node *node-id*[*acc-con-info*]

S set line *line-id* { controller { loopback | normal } | duplex { full | half } | protocol { ddcmp dmc | ddcmp point | ethernet } }

or

set line *line-id* all

S set { known logging | logging console | logging file | logging monitor } { name *name* | state { off | on } | [events *list* | known events [[circuit *circuit-id* | line *line-id* | node *node-id*]] [[sink { executor | node *node-id* }]]] }

S **set { node** *node-id* **}** { **address** *node-address* | **diagnostic file** *file* | **dump file** *file* | **hardware address** *E-address* | **host** *node-id* | **load file** *file* | **name** *node-name* | **secondary [loader]** *file* | **service circuit** *circuit-id* | **[service] password** *service-password* | **tertiary [loader]** *file* }

or

set { node *node-id* **} all**

S **set** { **object** *object-name* | **known objects** } [**accept** { **deferred** | **immediate** } | **default user** *login-name* | **file** *file-id* | **number** *number* | **type** { **sequenced packet** | **stream** }]

or

set { **object** *object-name* | **known objects** } **all**

A **show** { **circuit** *circuit-id* | **known circuits** } [**characteristics** | **counters** | **status** | **summary**]

A **show executor** [**characteristics** | **counters** | **status** | **summary**]

A **show** { **line** *line-id* | **known lines** } [**characteristics** | **counters** | **status** | **summary**]

A **show** { **known logging** | **logging console** | **logging file** | **logging monitor** } [**characteristics** | **status** | **summary** | **events**] [**known sinks** | **sink node** *node-id*]

A **show** { **node** *node-id* / **active nodes** / **known nodes** } [**characteristics** / **counters** / **status** / **summary**]

A **show** { **object** *object-name* / **known objects** } [**characteristics** / **summary**]

A* **tell** *node-id* [*acc-con-info*] *ncp-command*

S **trigger node** *node-id* { **physical address** *E-address* / **[service] password** *hex-password* / **via** *circuit-id* }

S **trigger via** *circuit-id* { **physical address** *E-address* / **[service] password** *service-password* }

S **zero circuit** *circuit-id* [counters]

S **zero executor [counters]**

S **zero line** *line-id* **[counters]**

S **zero** { **node** *node-id* / **known nodes** } **[counters]**

Appendix B

DECnet-Supplied Objects

This appendix lists and defines the DECnet–ULTRIX objects. Table B-1 lists these objects and gives the following information about them: object number, file name, default user, socket type, and accept mode.

Table B–1: Digital-Supplied DECnet–ULTRIX Objects

Object = tell	
Number	= 0
File	= /usr/bin/tell
Default user	=
Type	= Sequenced packet
Accept	= Immediate
Object = DEFAULT	
Number	= 0
File	=
Default user	=
Type	= Sequenced packet
Accept	= Immediate
Object = fal	
Number	= 17
File	= /usr/etc/fal
Default user	= guest
Type	= Sequenced packet
Accept	= Deferred
Object = nml	
Number	= 19
File	= /usr/etc/nml
Default user	= guest
Type	= Sequenced packet
Accept	= Deferred
Object = dterm	
Number	= 23
File	= /usr/etc/dtermd

(continued on next page)

Table B–1 (Cont.): Digital-Supplied DECnet–ULTRIX Objects

Default user	= guest
Type	= Stream
Accept	= Immediate
Object = mir	
Number	= 25
File	= `/usr/etc/mir`
Default user	= guest
Type	= Sequenced packet
Accept	= Deferred
Object = mail11	
Number	= 27
File	= `/usr/etc/mail11dv3`
Default user	= daemon
Type	= Sequenced packet
Accept	= Deferred
Object = dlogin	
Number	= 42
File	= `/usr/etc/dlogind`
Default user	= guest
Type	= Sequenced packet
Accept	= Immediate
Object = dtr	
Number	= 63
File	= `/usr/etc/dtr`
Default user	= guest
Type	= Sequenced packet
Accept	= Deferred

Table B-2 describes the services of Digital-supplied DECnet-ULTRIX objects.

Table B–2: DECnet–ULTRIX Object Descriptions

DECnet Object	Service
tell[1]	Utility that lets you execute commands on a remote DECnet–ULTRIX or DECnet–VAX node.
DEFAULT	You can use the DECnet zero object named to create your own network server programs without modifying the object database. If the DECnet object spawner does not find a match for an object in the object database, it searches for the program by using the path name specified in the **connect** request. The path name can be absolute (the full path name) or relative (the path defined by the user name and password).

[1]Not to be confused with the **tell** prefix used with **ncp** commands.

(continued on next page)

Table B–2 (Cont.): DECnet–ULTRIX Object Descriptions

DECnet Object	Service
	Object numbers are equivalent to object names and can be used on other networks unless the object number is zero. If the object number is zero, you must use an object name.
fal	Allows a process running on any node in a DECnet network to access another node's file system. The **fal** object uses the Data Access Protocol (DAP) to communicate with other nodes.
nml	The process that performs network management services. The **nml** object communicates with other nodes in the network by means of the NICE protocol.
dterm	A homogeneous command terminal service. The **dterm** object uses the TOPS-20 protocol.
mir	The remote loopback mirror, which performs looping services for remote users.
mail11	The DECnet mail service.
dlogin	The heterogeneous command terminal service. The **dlogin** object uses the CTERM protocol, which supports connections from DECnet–ULTRIX to all other DECnet nodes.
dtr	The DECnet Test Receiver that is used with the DECnet Test Sender (**dts**) to test logical links.

Appendix C

Ethernet Addressing

A unique Ethernet address identifies each node on an Ethernet line. You can send a message to any number of nodes on an Ethernet line, depending on the type of Ethernet address you use, physical or multicast.

To configure your network you need not specify the Ethernet address of a node. Whenever you execute the **ncp set executor state on** command, DECnet resets the Ethernet physical address to correspond to the DECnet node address.

Whenever the DECnet software changes the Ethernet address of your Ethernet controller, such as during DECnet startup, it invalidates the existing Internet Ethernet address mapping for the node. The ULTRIX Ethernet driver detects the change in address in 5 minutes. If you want to delete the mapping entry in the address translation table manually, use the **arp -d** command. See the ULTRIX **arp(8)** manual page.

C.1 Ethernet Address Format

Ethernet addresses are represented as six pairs of hexadecimal digits separated by hyphens (for example, AA-00-03-00-67-FF).

Xerox Corporation assigns a block of addresses to a producer of Ethernet interfaces upon application. Thus, every manufacturer has a unique set of addresses to use. Normally, one address out of the assigned block of physical addresses is permanently associated with each interface (usually in a read-only memory). This address is known as the *Ethernet hardware address* of the interface.

NOTE

You can use the **show line** *line-id* **characteristics** command to display the hardware address.

Digital's interface to Ethernet (the DEUNA or DEQNA controller at the node) has the ability to set a different logical address to be used by the interface. This address is known as the *Ethernet physical address*. When a node on the Ethernet initially starts up, the physical address is the same as the Ethernet hardware address. Then, when DECnet turns on a DEUNA or DEQNA device, DECnet constructs a physical address by appending the local node's node address to a constant 8-digit number derived from the block addresses assigned to Digital (AA-00-04-00).

Once the Ethernet physical address has been set to its new value, it is reset to its original hardware address value only when a reset is issued to the DEUNA or DEQNA (for example, when the machine power is shut off).

C.2 Ethernet Multicast Address Types

Ethernet physical addresses and Ethernet multicast addresses are distinguished by the value of the leading low-order bit of the first byte of the address:

- **Physical address.** The unique address of a single node on any Ethernet (low-order bit = 0).
- **Multicast address.** A multidestination address of one or more nodes on a given Ethernet (low-order bit = 1).

There are two types of multicast addresses:

- A **multicast group address** is an address that is assigned to any number of node groups so that they are all able to receive the same message in a single transmission by a sending node.
- A **broadcast address** is a single multicast group address (specifically, FF-FF-FF-FF-FF-FF) to which a message can be sent if it must be received by all nodes on a given Ethernet. (Use a broadcast address only for messages to be acted upon by all nodes on the Ethernet, since all nodes must process them.)

C.3 Ethernet Physical and Multicast Address Values

Digital physical addresses are in the range AA-00-00-00-00-00 through AA-00-04-FF-FF-FF. Multicast group addresses assigned for use in cross-company communications are as follows:

Value	Meaning
FF-FF-FF-FF-FF-FF	Broadcast
CF-00-00-00-00-00	Loopback assistance

Digital multicast group addresses assigned to be received by other Digital nodes on the same Ethernet are as follows:

Value	Meaning
AB-00-00-01-00-00	Dump/load assistance
AB-00-00-02-00-00	Remote console
AB-00-00-03-00-00	All Phase IV routers
AB-00-00-04-00-00	All Phase IV end nodes
AB-00-00-05-00-00	Reserved for future use
through	
AB-00-04-FF-FF-FF	

Index

M

N

O

P

R

S

T

Z

HOW TO ORDER ADDITIONAL DOCUMENTATION

DIRECT TELEPHONE ORDERS

In Continental USA
call 800–DIGITAL

In Canada
call 800–267–6215

In New Hampshire
Alaska or Hawaii
call 603–884–6660

In Puerto Rico
call 809–754–7575

ELECTRONIC ORDERS (U.S. ONLY)

Dial 800–DEC–DEMO with any VT100 or VT200
compatible terminal and a 1200 baud modem.
If you need assistance, call 1–800–DIGITAL.

DIRECT MAIL ORDERS (U.S. and Puerto Rico*)

DIGITAL EQUIPMENT CORPORATION
P.O. Box CS2008
Nashua, New Hampshire 03061

DIRECT MAIL ORDERS (Canada)

DIGITAL EQUIPMENT OF CANADA LTD.
940 Belfast Road
Ottawa, Ontario, Canada K1G 4C2
Attn: A&SG Business Manager

INTERNATIONAL

DIGITAL
EQUIPMENT CORPORATION
A&SG Business Manager
c/o Digital's local subsidiary
or approved distributor

Internal orders should be placed through the Software Distribution Center (SDC),
Digital Equipment Corporation, Westminster, Massachusetts 01473

*Any prepaid order from Puerto Rico must be placed
with the Local Digital Subsidiary:
809–754–7575 x2012

DECnet–ULTRIX
NCP Command Reference
AA–PB62A–TE

READER'S COMMENTS

What do you think of this manual? Your comments and suggestions will help us to improve the quality and usefulness of our publications.

Please rate this manual:

	Poor				Excellent
Accuracy	1	2	3	4	5
Readability	1	2	3	4	5
Examples	1	2	3	4	5
Organization	1	2	3	4	5
Completeness	1	2	3	4	5

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

_____

_____

_____

_____

General comments:

_____

_____

_____

_____

Suggestions for improvement:

_____

_____

_____

_____

Name _____ Date _____

Title _____ Department _____

Company _____ Street _____

City _____ State/Country _____ Zip _____

DO NOT CUT – FOLD HERE AND TAPE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

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

POSTAGE WILL BE PAID BY ADDRESSEE

digital™

Networks and
Communications Publications
550 King Street
Littleton, MA 01460–1289

DO NOT CUT – FOLD HERE

CUT ON THIS LINE