ESV Workstation

ES/Dnet User's Manual

EVANS & SUTHERLAND COMPUTER CORPORATION
Salt Lake City, Utah

DOCUMENTATION WARRANTY:

PURPOSE: This documentation is provided to assist an Evans & Sutherland trained BUYER in using a product purchased from Evans & Sutherland. It may contain errors or omissions that only a trained individual may recognize. Changes may have occurred to the hardware/software, to which this documentation refers, which are not included in this documentation or may be on a separate errata sheet. Use of this documentation in such changed hardware/software could result in damage to hardware/software. User assumes full responsibility of all such results of the use of this data.

WARRANTY: This document is provided, and Buyer accepts such documentation, "AS-IS" and with "ALL FAULTS, ERRORS, AND OMISSIONS". BUYER HEREBY WAIVES ALL IMPLIED AND OTHER WARRANTIES, GUARANTIES, CONDITIONS OR LIABILITIES, EXPRESSED OR IMPLIED ARISING BY LAW OR OTHERWISE, INCLUDING, WITHOUT LIMITATIONS, ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. BUYER FURTHER HOLDS SELLER HARMLESS OF ANY DIRECT OR INDIRECT DAMAGES, INCLUDING CONSEQUENTIAL DAMAGES.

ES/Dnet was developed with Ki Research exclusively for Evans & Sutherland Computer Corporation.

Part Number: 517941-500 AA
April, 1990

Table of Contents

1. Introduction

About ES/Dnet

ES/Dnet is a communications package that provides access between the ESV Workstation and VAX systems running a DECnet Ethernet local area network. ES/Dnet allows the ESV Workstation to be a Phase IV DECnet End-Node. With ES/Dnet you can:

- Copy files to or from DECnet nodes
- Do remote login to or from DECnet nodes
- Perform task to task communication
- List a directory residing on a DECnet node
- Rename, delete or print files on a DECnet node
- Submit to batch files on a DECnet node
- Exchange mail with other DECnet nodes

Similarly, from other DECnet nodes, such as a VAX, you can perform all of the above operations on the ESV Workstation.

DECnet is sold in two forms: *End-Node* and *Full-Function*. The major difference between these two forms is routing, which is the ability to read packets at an intermediate node and pass on or route the packets to their destination node. Routing is only available in Full-Function DECnet and currently not available in ES/Dnet.

ES/Dnet does not currently support diskless workstation nodes.

About This Manual

This manual is organized as follows:

- Chapter 1, "Introduction," provides an overview of ES/Dnet and describes the organization of this manual.
- Chapter 2, "User's Information," covers the routine operations of ES/Dnet, focusing on file transfer and remote login.
- Chapter 3, "Programmer's Information," describes the ES/Dnet Network Procedure calls that provide a programming interface to a DECnet local area network.
- Chapter 4, "Network Management," covers the Network Management facilities.

- Chapter 5, "Command Summary," contains a summary of all the ES/Dnet commands in alphabetical order.
- Appendix A, "Troubleshooting," gives solutions for some of the more frequent problems that might occur.
- Appendix B, "Error Codes," contains a list of error codes and their meaning.
- Appendix C, "Modifying UNIX **sendmail.cf**," contains a Tektronix example.
- Appendix D, "Glossary," contains a list of terms used in this document.
- Appendix E, "Installation," contains ES/Dnet installation instructions.

Document Conventions

- *Classic italics* indicate a new term that is being defined.
- CLASSIC SMALL CAPS represent the names of keys on the keyboard.
- **Modern bold** represents any command, function, option, data type, or filename.
- ***Modern bold italics*** represent a variable element which must be filled in with a specific value when actually used.
- `Typewriter normal` represents anything a user sees on the screen, the contents of a file, or sample code.
- **`Typewriter bold`** represents anything typed on a command line in response to a system prompt.

2. User's Information

This chapter describes the ES/Dnet file transfer, remote login, and mail commands.

Introduction

ES/Dnet allows you to transfer files between systems or to login to the remote system (DECnet node) from your ESV Workstation. This is accomplished by the following two ES/Dnet programs: **dap** and **sethost**.

- **dap** is a file manipulation program.
- **sethost** is a virtual terminal program that allows you to log in to a remote node.

In addition, you can send mail to other nodes using **dnamail** or you can modify your **sendmail.cf** to recognize a string that signifies a VAX host and call **-P=dnamail**. See examples of the modified **sendmail.cf** in **dna/examples/unix**.

This chapter is written for users who will be using ES/Dnet for routine file transfer, management, or batch submittal; for remote login; or for sending mail. If you plan to use the ES/Dnet programming interface, see Chapter 3, "Programmer's Information." If you are a system administrator, see Chapter 4, "Network Management." Chapter 5, "Command Summary," contains complete command descriptions for all commands discussed in this chapter.

This chapter consists of the following five sections:

- "Using **dap**" describes file transfer, management, or batch submittal from the ESV Workstation to a DECnet node.
- "File Management from a Remote System" describes file transfer from a DECnet node to the ESV Workstation.
- "Logging In to a Remote System" describes remote login from the ESV Workstation to a DECnet node.
- "Logging In from a Remote System" describes remote login from a DECnet node to the ESV Workstation.
- "Using **dnamail**" describes the mail facility.

Using dap

dap is a file manipulation program that allows the ESV Workstation and the DECnet node to use each others' file systems. The commands include file transfer, file management, general utility, and batch submittal commands.

File Transfer and Management Commands

- The **copy** command is a bidirectional command that allows you to copy files between the ESV Workstation and a remote DECnet node. With the **copy** command, you specify a source and a destination in which either can be the remote file and the other the local file.
- The **delete** command deletes a file at the remote node.
- The **directory** command lists the contents of a remote directory.
- The **get** command retrieves a file from a remote DECnet node and stores the file on the local node.
- The **rename** command renames a file at the remote node.
- The **send** command stores a local file on the remote node.
- The **type** command displays a file to the screen.

General Utility Commands

- The **cd** command changes the local current working directory to the directory you specify.
- The **default** command sets the default access control for the remote node including node name, user name, password, and account name.
- The **exit** command exits **dap** and returns you to your previous prompt.
- The **help** command displays help text which includes descriptions of all of the **dap** commands.
- The **pwd** command lists the local current working directory.
- The **shell** command lets you enter shell commands, or leave **dap** and enter a shell.
- The **show** command displays the current default access control for the remote node.

Batch Submittal Commands

- The **print** command prints a file at the remote node.
- The **submit** command submits a batch file for remote execution.

Syntax Considerations

With ES/Dnet you can access files on any system that is running DECnet protocol. All file specifications for a remote VMS file must be in the standard VMS form of:

```
node::device:[directory.subdirectory]
      file_name.file_type;version
```

device could be a logical name, such as **sys$system** or **USER$1:**, or an actual device, such as **DRA0:**. A **directory** can be one or more subdirectories.

VMS to UNIX Pathname Conversion

If you are familiar with VMS, the following list shows how VMS pathnames are converted into UNIX pathnames.

VMS Pathname	UNIX Pathname
dua1:	**/**
dua1:[dir.sub]fname.ext;#	**/dir/sub/fname.ext**
dua1:[.dir.sub]fname.ext;#	**/dir/sub/fname.ext**
[dir]fname.ext;#	**/dir/fname.ext**
[dir.sub]fname.ext;#	**/dir/sub/fname.ext**
[]fname.ext;#	**/fname.ext**
[.sub]fname.ext;#	**./sub/fname.ext**
fname.ext;#	**fname.ext**

Keep in mind that VMS does not support files which have more than one "." in them.

UNIX to VMS Pathname Conversion

If you are familiar with UNIX, the following list shows how UNIX pathnames are converted into VMS pathnames.

UNIX Pathname	VMS Pathname
fname	**fname;1**
/fname	**hostname::fname;1**
/sub/fname	**node::[sub]fname;1**
./sub/fname	**[.sub]fname;1**

Note: All returned filenames from a VAXStation from the ESV Workstation will have a VMS version number of "1" since some VMS commands can't handle filenames without a version number.

Wildcard Usage

The asterisk wildcard character will match any character, including ".". Therefore, a pattern of "*.*" matches all filenames that contain ".", whereas the pattern "*" matches all filenames. The following sections describe the rules for using wildcards with the various **dap** commands.

Using Wildcards with rename

You can use wildcard specifiers to rename a number of files on the remote system. All filenames are processed as being one of these standard forms:

- **device:[dir_specifier]filename.extension;version**
- **/dir_specifier/filename.extension**
- **dir_specifier:filename.extension**

Wildcard substitution is allowed in the filename, extension, and version fields. The substitution performed is as follows. When an asterisk is found in the filename, extension, and version fields of the new filename, the asterisk will be replaced by the contents of the corresponding field from the old filename. Thus, it is possible to rename classes of files. For example,

Old	New	Result
my_file.c;7	***.c_old;***	**my_file.c_old;7**
my_file.c;7	***.*_old;***	**my_file.c_old;7**

Note: The second example shows how ES/Dnet enhances the VMS substitution logic. Under VMS, you can only use the asterisk by itself in the new field. ES/Dnet allows the asterisk to be combined with other text, allowing more versatility. However, there can only be one asterisk in each of the three fields. Any other asterisks will be left in the field as a literal character.

When performing the substitutions, ES/Dnet copies the device, directory, and version fields from the old filename if they were not specified in the new filename. To rename across directory/device boundaries, the new directory/device paths must be explicitly specified in the new filename. ES/Dnet does not copy filename or extension fields. If not specified in the new filename, they will not be included when communicating with DEC systems.

A version field is considered specified if there is a ";" in the new filename. There does not have to be a value associated with the version, as VMS considers a lone ";" as meaning "newest" version. For example,

Old	New	Result
mine.c	***.ip**	**mine.ip**
mine.c	**xyz.***	**xyz.c**
mine.c	**a*.x*b**	**amine.xcb**
mine.c	**a*b**	**amine.cb**
mine.c	**new.txt**	**new.txt**
mine	**a*b.c*d**	**amineb.cd**
.x	**a*b.c*d**	**ab.cxd**
abc.def;7	***.*;**	**abc.def;**
abc.def;7	***.*;*_1**	**abc.def;7_1**
abc.def;7	***.c**	**abc.c;7**

Using Wildcards with copy, send, and get

This wildcard substitution differs from **rename** in one very important way: device and directory specifiers in the old filename are not copied into the new filename. This is because the old and new filenames belong to entirely different file systems. If a device and/or directory specifier is required in the new filename, it must be specified explicitly. Also, if the

new field is specified as a single asterisk (*), then all fields are left intact. For example,

```
DAP> copy /user/sub/myfile.c vax::* -log
/user/sub/myfile.c copied to vax::myfile.c
DAP> copy /user/sub/myfile.c vax::*. -log
/user/sub/myfile.c copied to vax::myfile.
DAP> copy /user/sub/myfile.c vax::dua1:[hisdir]*.*_old
/user/sub/myfile.c copied to vax::dua1:[hisdir]myfile.c_old
```

Multiple "*" and "%" wildcards are allowed in any part of the filename. Wildcards are not allowed in any directory specifiers. For example,

/dir/sub/fnam*e.* is valid

/dir/s*/fnam*e.* is not valid

Using Wildcards with delete, directory, print, purge, and submit

How you use wildcards with these commands is dependent upon a VMS wildcard capability. For VMS, this means that only one "*" may be used in either the filename, extension, or version fields. When used in the version field, there should be no other characters in the field. Between ES/Dnet systems, multiple "*" are supported as are multiple "." within the filename.

Access Control

VMS access control information is used to limit access to directories and files on the remote system. This access control is defined with:

```
node"username password"::
```

There are four basic ways to include the access control information with your network requests:

- *Explicit access.* You include the access information (*i.e.*, the node name, username, and password) with every request.
- *Proxy access.* An account is set up on the remote system that contains your node name, username, and password mapped to a proxy name. You then use the proxy instead of the node name, username, and password.
- *Explicit proxy.* Only a username is specified with the double quotes. The account specified with the user's default proxy is used.
- *Default access.* The access information is defined in files on the VAX and the ESV Workstation.

If there is no explicit access information in the command line, proxy access is assumed by the remote system. If proxy is not defined on remote system, the default access specification is used.

Establishing Explicit Access

You must get a username and password from the VAX system manager. This username and password pair will allow you access to the files you want to transfer, manage, or submit to batch queues.

Establishing Proxy and Explicit Proxy Access

In order to allow proxy access to the remote host, you must give the remote system manager your node name and user ID name. The remote system manager will assign you a user account that will allow you to use **dap** or other network commands without specifically entering the node name, username, and password.

VAX 5.0 proxy access allows you to stipulate a username without password for each user account you have access to. For example, if you had access to **system** and **dwatts** accounts on the VAX, your default may be **system**, and if **dwatts** was defined for your ES/Dnet system name and username, you could use **dwatts** as the access control information. You would use **dwatts** user account on the VAX.

If a remote system user will be accessing the ES/Dnet system, the ES/Dnet system manager will also need to add proxy access for you in the form:

```
remote_nodeID::remote_username local_username
```

Establishing Default Access

Default access on the remote system is established by the VAX system manager, who uses the NCP command **DEF/SET DEFAULT USER**.

On the ES/Dnet system, you can add default access by editing the **proxy.db** file located in **/usr/etc/esdnet/database/proxy.db** to include:

```
local_username
```

Using setenv to Specify Access Control on UNIX Systems

If you continually access the same remote system, you can use **dap** commands without specifying the remote node and access control each time if you first specify the default access with **setenv**. For example,

```
setenv DAP_DEFAULT 'nodename"username password"::'
```

Now, **dap** will check the **setenv** variable and automatically append it the **dap** commands that allow it, that is all commands other than **copy**.

Note: If you use **setenv** to specify access control, be aware that your username and password will be visible to other users who use the **printenv** command from your account.

Commands Using dap Default Access Control Information

As stated, access control information may consist of username and password for a given node. The **dap** default command allows you to specify access control information and the node for use with other **dap** commands.

All **dap** commands other than **copy** may use access control specified by default command.

With the **get** and **send** commands, if you have defined access with the **default** command or **setenv** command, you do not need to specify the access control. The access control information specified with **default** will be appended to the source filespec with **get**, and to the destination filespec with **send**, **print**, **submit**, **delete**, and **directory**.

Note that **device** need not be included with the file specifications, as the access control implies a default device associated with the specified user password, proxy, or default network accounts.

Getting dap Default Access Information

Entering **show** or **default** without any options displays the current access information, except the password, which is always displayed as **PASSWORD**.

dap Modes of Operation

The **dap** program has two modes of operation: *command line* and *interactive*. In command line mode, you enter the **dap** command with all desired options and arguments, followed by a RETURN. In interactive mode, you enter the **dap** commands, options, and arguments in response to a **DAP>** prompt. In either mode, if you do not enter necessary information, **dap** prompts you for it and waits for input.

Command Line Mode

The format for using **dap** in command line mode is:

```
$ dap command [options] argument1 [argument2]
```

In this format, the command is any one of the **dap** commands; options include any supported option for the command; **argument1** is either the source file or target file/directory; and **argument2** is the destination file. The following list describes the conventions for entering **dap** commands:

- **dap** commands and options are case insensitive; that is, they can be entered in upper case, lower case, or a combination.
- Options are preceded by a hyphen (-), as is standard with UNIX commands.
- Options must be separated by spaces.
- You can enter an option anywhere on the command line.
- File names and directories must be entered as they were specified at the local and remote node.

An example of a **dap** command is:

```
$ dap copy my_file nodex::new_file
```

Entering just **dap copy** will result in the user being prompted for input while in command line mode.

```
$ dap copy
Source: my_file
Destination: nodex::new_file
```

In this example, the user didn't specify any options, but did specify a source and destination file.

Interactive Mode

The format for **dap** in interactive mode is:

```
$ dap
DAP> command [options] argument1 [argument2]
DAP> exit
$
```

Optionally, you can specify only the command in interactive mode and **dap** prompts you for the command options and arguments, as in the following example:

```
$ dap
DAP> copy
Source: my_file
Destination: nodex::new_file
DAP> exit
$
```

dap Commands

The following tables list the **dap** commands and command options. In the command option table, the amount of entry needed to uniquely identify the option is represented in uppercase

Table 2-1. dap Commands

Command	Options	Description
cd	**-log**	Changes the local current working directory.
copy (remote to local)	**- ascii, –append, -bin, –crc, -hash, - log, -nblock, -noattributes, –nocrc, -replace, -statistics, -transmit_buffer**	Copies a remote file to a local file.
copy (local to remote)	**-ascii, -binary, -crc, -cr, -log, -fix, -hash, -nblock, -nocrc, -print, (-print -delete), -statistics -transmit_buffer**	Copies a local file to a remote file.
default	**-clear**	Sets the default access control for the remote node.
delete	**-log**	Deletes a file at the remote node.
directory	**-full, -size, -date, -ufull**	Lists the contents of a remote directory.
exit	(none)	Exits the **dap** program.
get	**-append, -ascii, -bin, -crc, -hash, -log, -nblock, -noattributes, -nocrc, -replace, -statistics, -transmit_buffer**	Copies a file from the remote node.
help	(none)	Displays help text.
print	**-log**	Prints a file at the remote node.
pwd	(none)	Lists the current local directory.
rename	**-log**	Renames a file at the remote node.

Table 2-1. dap Commands (con't.)

Command	Options	Description
send	**-ascii, -binary, -crc, -cr, -fix, -hash, -log, -nblock, -nocrc, -print, (-print -delete), -statistics -transmit_buffer**	Copies a local file to a remote file.
show	(none)	Displays the current default access control for the remote node.
submit	**-log**	Submits a batch file for remote execution.
type	**-log**	Types a remote file.
usage	(none)	Displays usage text.
shell	(shell commands enclosed in double quotes)	Lets you enter shell commands.

Table 2-2. dap Command Options

Option	Description
-Append	Appends a source file to a destination file after retrieving it from a remote system. The destination file must exist prior to using this option.
-ASCII	The file is created on the VAX with **stream-lf** record format and RETURN carriage control record attributes.
-Binary	The file is created on the VAX with fixed length record format. When it is a remote network, no conversion is done on file.
-CLear	Remove an existing default access control.
-CRC	**dap** CRC is applied and calculated.
-CR	The file is created on the VAX with **stream_cr** record format and RETURN carriage control record attributes. (The file must have RETURN as record delimiters.)
-date	Will return file creation date on **dir** command.
-Fix=#	Set the record size of the fixed length file on the destination.

Table 2-2. **dap** Command Options (con't.)

Option	Description
-Hash	Display "#" for each 150 KByte transferred.
-Log	Displays filenames, verifying that a user command was completed.
-NBlock	Sets the size of **dap** reads and writes to the network. The default is 7. The read and write operations are 3644 bytes long. This includes 60 bytes of **dap** overhead and 7*512 bytes of user data.
-NOCRC	Disables **dap** CRC.
-NOAttributes	VAX file record attributes are ignored when creating the file on the local system
-PRint	Copies a file to the remote system and then prints it on the remote system's printer.
-PRint -Delete	Copies a file to the remote system, prints it on the remote system's printer, and then deletes the file once the print operation is finished.
-REplace	Retrieves a remote file and deletes any local file of the same name.
-SIZe	Specifies the number of bytes and blocks of the file on **dir** command.
-STATistics	Displays file transfer statistics such as bytes per second, read and write operations, etc.
shell commands	Any valid shell command enclosed within double quotes.
-TRansmit_buffer	Specifies the depth of the **dap** transmit buffer. The default is 1024.
-ufull	Specifies UNIX is full. Creates the date, size and protection of file on the **dir** command.

copy - Copying a File from a Remote System

The following example retrieves a file from a VAX and stores it on the ESV Workstation. You may replace **old_file** with **my_file**'s contents. No CRC calculation will take place on this transfer.

```
$ dap
DAP> copy -replace -nocrc
Source: nodex"smith 1qazwsx"::my_file
Destination: old_file
DAP>
```

In the following example, the proxy has been defined at the VAX. To copy a file from the VAX to your ESV Workstation, issue the interactive command below. The VAX file is appended to the existing_file.

```
$ dap
DAP> copy  -append
Source: nodex::my_file
Destination: existing_file
DAP>
```

get - Getting a File from a Remote System

The following example uses the **get** command to copy a file from a VAX using access control information. The **-statistics** option generates throughput statistics.

```
$ dap
DAP> get -stat
Source: nodex"smith 1vquartz"::my_file
Destination: local_file
DAP>
          =-=-= Statistical Summary =-=-=
                                          RECV     XMIT    Composite
Number of DAP Physical Messages           5        9        14
Number of DAP Logical Messages            9        13       22
Bytes of User Data                        0        15370    15370
Bytes in DAP Layer                        141      15634    15775
Percent of User Data in DAP Layer         0.00     98.31    97.43
User Data Throughput (Bytes/Sec)          0        5123     5123
DAP Layer Throughput (Bytes/Sec)          47       5211     5258
User Records or Blocks                    0        0          0
================================================================

Logical Link Terminated on 1988-9-28 14:  9:23
Logical Link Established on 1988-9-28 14: 9:20
```

```
Total Transfer Time                          3 Seconds
Negotiated DAP Buffer Size                   3644 Bytes
DAP Convert Buffer Size                      16384 Bytes
Remote Filename                              my_file Local
Filename                                     local_file
```

The following example uses the **get** command with explicit access control (and not the proxy access control). The **-log** option gives the status of both copied and uncopied files and verifies each requested file transfer operation.

```
$ dap
DAP> get -log
Source: vaxapo"nopriv password"::*.a*
Destination: *.vba
test.vba was copied from USER1:[KI]TEST.ACC;1 SYS_EVENT
      (DAP_CLOSE: 0.11) Saturday, 27-Aug-1988 21:10:00
DAP_PE_ERROR_STATUS -- see macro and micro codes below DAP
Status Code - MACRO CODE: 4, FILE_OPEN DAP
Status Code - MICRO CODE: 85, Privilege violation
      (OS denies access)
file copy operation was not successful test1.vba not copied
      from  TEST1.ACF;1
test2.vba was copied from USER1:[KI]TEST2.ACC;1
test3.vba was copied from USER1:[KI]TEST3.ACC;1
```

Note: Wildcard transfers will continue if **dap_pe_error** status errors occur.

send - Sending a File to a Remote System

The following example uses the **send** command to copy a file from the ESV Workstation to a remote node and uses the access control information defined using the default access.

This example uses the **-log** option to list files copied to the remote node. The destination file names will have the default access appended to them because you entered default access information.

```
$ dap
DAP> send -log
Source: *.vba
Destination: *.vbas
./taaaas.vba copied to USER1:[KI]TAAAAS.VBAS;3
./test.vba copied to USER1:[KI]TEST.VBAS;4
./test2.vba copied to USER1:[KI]TEST2.VBAS;4
./test3.vba copied to USER1:[KI]TEST3.VBAS;4
```

default, show - Setting Up, Showing, and Clearing the Default Remote Node Access Control

The following example uses the **default** command to set up default access control. This default access control may be used in all commands except **copy**. Using **default** saves you from having to include remote node access control information each time you use one of the previous commands; the default access control information is used for the remote request.

```
$ dap
DAP> default
rvax"username password"::
DAP>
```

Next, the **show** command displays the default access control.

```
$ dap
DAP> show
rvax"username PASSWORD"::
DAP>
```

If no access control has been defined, ES/Dnet displays **No Default Set**. This indicates that you will have to use the default account on the remote system if proxy is not defined for your local account.

The **show** command and **default** command do not display the password, but instead display **PASSWORD** in place of the actual password.

To clear the default, enter **default -clear**, or enter new access control information.

shell - Executing a UNIX Shell Command

Use the shell command when you want to enter the UNIX shell and issue a UNIX command. This is useful when you want to issue UNIX commands without leaving **dap**. For example,

```
$ dap
DAP> shell "ls -a"
total 228
drwxr-xr-x  8 dand         1024 Mar  3 18:06 .
drwxrwxr-x 37 root         1024 Feb 27 20:40 ..
-rw-r--r--  1 dand          516 Mar  3 13:21 .cshrc
-rw-r--r--  1 dand         1246 Mar 1 08:22 .login
-rw-r--r--  1 dand           33 Mar  3 11:46 .mh_profile
-rw-r--r--  1 dand          281 Feb 28 16:53 .profile
-rw-------  1 dand           39 Mar  1 08:20 .rhosts
```

```
drwxr-xr-x  2 dand          512 Mar  1 08:51 Data_A
drwx------  2 dand          512 Mar  1 16:27 Data_B
drwxr-xr-x  2 dand          512 Mar  1 08:50 Data_C
drwxr-x--x  3 dand          512 Mar  3 11:46 Mail
drwxr-xr-x  2 dand          512 Mar  3 09:14 TEST
drwxr-xr-x  2 dand          512 Mar  3 17:18 Teknicad
-rw-r--r--  1 dand          516 Mar  3 18:06 cshrc.
-rw-r--r--  1 dand        16384 Mar  3 18:22 demo
-rw-r--r--  1 dand          995 Mar  3 17:41 ncp.cmd
-rw-r--r--  1 dand         4142 Mar  3 16:02 trace.trc
-rw-r--r--  1 dand        55808 Mar  3 16:08 vscope.exe
DAP>
```

print - Printing Files

The following example shows how to print a file residing on the remote system. The example shows a wildcard print.

```
$ dap
DAP> print -log
File: VAX"username password"::login.com*;*
USER1:[KI]LOGIN.COM;21 was printed
USER1:[KI]LOGIN.COM;20 was printed
USER1:[KI]LOGIN.COM;18 was printed
```

If you have defined a default access, you can enter **print** and the **filename** to be printed. The **dap print** command will use the default access control information to access the remote node's file specification. For example,

```
DAP> default VAX"username password"::
DAP> print login.com;*
```

Using Print Options with copy and send

You can use the **copy** command or **send** command with options to copy then print files at the remote host.

- The two print options are **-print** and **-print -delete**.
- Use the **copy** command with either option when the file is local to your system and needs to be copied to the remote node.
- Use the **print** command when the file is already located on the remote system. A printer must be available with a print queue defined on the remote system.

The following example shows how to copy a file and print it on the remote system using the **copy** command with **-print** option. It also

uses the **-log** option to display the printed file names as they are queued to the remote system's printer.

```
$ dap
DAP> copy -print -log
Source: test_file.doc
Destination:vaxnet"jones wxc4my"::[test.jobs]hostfile.doc
[test.jobs]hostfile.doc was printed
DAP>
```

The following example sends a file to the remote system, prints it, and then deletes it.

```
$ dap
DAP> send -print -delete
Source: file.c
Destination: vaxpo"username password"::file.c
DAP>
```

Getting Job Status

dap will return a status message when the job has been submitted to the default queue. To find out if the job has completed, you will have to use **sethost** to log in to the remote host and then use whatever command is required by the remote host to check the job's status. For VMS systems, use **SHOW QUEUE SYS$PRINT** if the file is still queued.

submit - Submitting a File for Remote Execution

You can use the **submit** command to submit a file for execution at the remote node. The following example uses a **submit** command with a **-log** option.

```
$ dap
DAP> submit -log
Batch File: vaxnet"jones wxc4my"::[test.jobs]testfile.com
[test.jobs]testfile.com was submitted
DAP>
```

For VMS, you can verify that the job was submitted by looking for a **testfile.log** file in the default directory of the explicit access control account specified or proxy account (if proxy is enabled on remote system). You can also submit a number of files using a wildcard specification. For example,

```
$ dap
DAP> submit -log
Batch File: vaxnet"username password"::login.com;*
```

```
USER1:[KI]LOGIN.COM;21 was submitted
USER1:[KI]LOGIN.COM;20 was submitted
USER1:[KI]LOGIN.COM;18 was submitted
```

After this wildcard submit is complete, you will find the following **.log** files in the default directory:

```
Vax> dir -date login.log
Directory USER1:[KI]
LOGIN.LOG;21 28-AUG-1988 23:33
LOGIN.LOG;20 28-AUG-1988 23:33
LOGIN.LOG;18 28-AUG-1988 23:33
Total of 3 files.
```

If you specify a file on the remote system which you cannot submit, you will get the following error. In this case, it is a privilege violation. (The remote system is a VAX.)

```
$ dap
DAP> submit -log
Batch File: vaxnet"username password"::login.com;19
SYS_EVENT (DAP_CLOSE: 0.11) Sunday, 28-Aug-1988 16:16:09
      DAP_PE_ERROR_STATUS -- see macro and micro codes below
DAP Status Code - MACRO CODE: 4, FILE_OPEN
DAP Status Code - MICRO CODE: 85, Privilege violation
      (OS denies access)
USER1:[KI]LOGIN.COM;19 was not submitted
```

If you had used the **default** command to define access control, you would not need to specify the remote node name, username, or password. **dap** would use the default access control information to access the remote file specified in your **submit** command.

Getting Job Status

dap will return a message to the user when the job has been submitted to the queue. To find out if the job has completed, you will have to use **sethost** to log in to the remote host and then use **Show Queue Sys$Batch** to check the job's status if still queued.

delete - Deleting Files

You can use the **delete** command to delete remote files at the VAX. You can specify the **-log** option to see what file was deleted. The following example shows how to delete a file on a remote system, and includes the **-log** option. The remote system is a VAX.

```
$ dap
DAP> delete -log
```

```
File: vaxnet"jones wxc4my"::[test.jobs]testfile.doc
[test.jobs] testfile.doc was deleted
DAP>
```

If multiple versions of the same file exist on the VAX, you must specify a wildcard or **dap** assumes you mean only the latest version should be deleted. For example,

```
$ dap delete -log vaxnet"username password"::test.vbas
USER1:[KI]TEST.VBAS;6 was deleted
```

To delete all versions of a file, as in the case above, you must use an explicit wildcard, such as **test.vbas;***.

If you have defined default access, you need not specify access control information, which includes node name, username, and password. For example you may enter the commands:

```
DAP> default vaxnet"username password"::
DAP> delete test.vbas;5
```

dap delete command uses the access control information specified with the default command to access specified filename.

rename - Renaming a Remote File

You can use the **rename** command to change the name of a remote file. The following example renames a file and uses the **-log** option. This option displays old and new filenames as the rename operation completes.

```
$ dap
DAP> rename -log
Old Name: vaxnet"jones wxc4my"::[test.jobs]test.o
New Name: new.o
[testfile.jobs]test.o was renamed to user1:[test.jobs]new.o
DAP>
```

The **-log** option allows you to display each old and new filename as the rename operation completes. If you do not specify a wildcard, only the latest version of the remote file will be renamed. For example,

```
$ dap rename -log 1"username password"::*.tst7 *.tst8
USER1:[KI]DEL.TST7;18 was renamed to USER1:[KI]DEL.TST8;1
USER1:[KI]STRM.TST7;18 was renamed to USER1:[KI]STRM.TST8;1
USER1:[KI]CRLF.TST7;18 was renamed to USER1:[KI]CRLF.TST8;1
```

If you have defined access control with the **default** command, you specify the **rename** command with just the old remote filename and new filename, respectively. You do not have to specify access control with

the old remote filename. **dap** will append the access control information and remote node specified with the **default** command.

directory - Getting a Directory Listing on a Remote System

The **directory** command allows you to get a directory listing from a remote system. You can use wildcards in the remote file specification. The following example obtains a directory listing on a VAX.

```
$ dap
DAP> directory
Directory: vaxnet"username password"::*.*
A.VFC;1
ARGS.BIN;1
A_VFC.TST;4
EA0.TRC;1
Total of 4  files
```

Also, you can specify a single filename in a directory command as shown in the following example:

```
$ dap
DAP> directory
Directory: vaxnet"username password"::varcrlf.tst7;1
USER1:[KI]
VARCRLF.TST7;1
Total of 1 file
```

Alternately, you can ask for a **-full** listing on your system's command line.

```
$ dap
DAP> directory -full
Directory: vaxnet"username password"::test.com
SYS$SYSROOT: [SYSMGR]
TEST.COM;1                          File ID:   None
Size:           1/3                 Owner:     [1,4]
Created: 17-MAR-89 11:06:09  Revised: 17-MAR-89 11:06:09 (1)
Expires:  <None specified>  Backup:   <No backup recorded>
Record format:      Stream_lf
Record attributes:  Carriage Control
Journaling enabled:None
File Protection: System:RWED, Owner:RWED, Group:RE, World:
Access Cntrl List:  None
      Total of 1 file  ,       1/ 3 blocks.
```

type

You can use the **type** command to display file to standard out. By entering the following, the file is displayed to the screen.

```
DAP> type login.com
$ncp: = mcr ncp
$sethost: == set host
```

UNIX users can pipe output to more from command mode, not interactive mode.

```
$dap type login.com/more
```

File Management from a Remote System

Commands from a remote system to your ESV Workstation are handled by the ES/Dnet's **d_server**. The **d_server** completes the copy requests from the remote system on your ESV Workstation. The **d_server** network object is defined using the **pconfig** command utility. See Chapter 4, "Network Management," for more details. The **d_server** supports the following commands:

- **copy**
- **create**
- **delete**
- **directory**
- **print**
- **rename**
- **submit**
- **type**

All command options will work with ES/Dnet. Some have no meaning and will be ignored.

You may also **SETDEF** from a VMS system to a directory on the ESV Workstation. For example, entering the following command puts you into the **/UDD/WATTS** UNIX directory.

```
VMS> SET DEF BEAST::[UDD.WATTS]
```

Specifying UNIX Filenames with VMS Syntax

When you are logged in to a remote VAX system and specify a file on the ES/Dnet system, you must use a file specification with either of two formats. You can (1) enclose the UNIX pathname in double quotes, as in:

```
node"user password"::"/smith/jobsdir/newfile"
```

The previous example uses the standard UNIX file specification but encloses it in double quotations to prevent VMS from interpreting the file specification.

Or, you can (2) use square brackets to delimit the path, as in:

```
node"user password"::"[smith.jobsdir]newfile"
```

To use wildcard operations with VAX **copy**, **delete**, **print**, **rename**, **submit**, and **type**, you must use the square bracket syntax.

Note: When using square bracket syntax, the directory names on the ES/Dnet system cannot have a period in them; the **d_server** cannot access directories thus named. To specify the root directory, you must indicate root as:

```
UNIX node"username password"::root:*.*
```

Wildcard Usage

You can use multiple "*" and "%" wildcards in any part of the filename. Wildcards are not allowed in any directory specifiers. For example,

/dir/sub/fnam*e.* is valid
/dir/s*/fnam*e.* is not valid

Note that VMS always expands filename templates to include an extension if one is not specified. That means that if you issue the command,

```
VAX> dir myunx::abc*
```

the VAX will expand the request into **dir myunx::abc*.***, which results in a list of all filenames starting with **abc** and containing a "." in the filename. This means that the file **abcdef** will not match the pattern. This is due to a limitation of VMS in assuming all file systems use "." in filenames.

copy - Copying Files

When logged into the VAX, you can use VMS commands to copy files between a VAX and the ESV Workstation. The following example shows how to copy a local VAX file to your workstation using the double quote syntax.

```
Vax> copy
From: filename.c
To: anode"id password"::"/atuser/test.c"
```

```
Vax>
```

The following sequence accomplishes the same task, but uses the alternate square bracket syntax.

```
Vax> copy
From: filename.c
To: anode"id password"::[atuser]test.c"
Vax>
```

delete - Deleting Files

You can use the **delete** command to delete a file on your ESV Workstation from a remote system. The first **delete** command in this example deletes a wildcard-specified filename; the second is fully specified with no wildcard. Both examples assume a default **/CONFIRM** option has been set up by the VAX system manager.

```
Vax> delete/log ranger"dwatts password"::"*.tst8"
RANGER"dwatts password"::strm.tst8, delete? [N]:y
DELETE-I-
FILDEL,RANGER"dwattpassword"::strm.tst8 deleted (6 blocks)
RANGER"dwatts password"::del.tst8, delete? [N]:y
DELETE-I-FILDEL, RANGER"dwatts password"::del.tst8
deleted (20 blocks)
%DELETE-I-TOTAL, 2 files deleted (26 blocks)

Vax> delete ranger"dwatts password"::"/dwatts/strm.tst7"
RANGER"dwatts password"::"/dwatts/strm.tst8", delete? [N]:y
%DELETE-I-FILDEL, RANGER"dwatts
password"::"/dwatts/strm.tst7"
deleted (6 blocks)
```

The following example accomplishes the same task using the alternate square bracket syntax:

```
Vax> delete ranger"dwatts password"::[dwatts]strm.tst7;*
```

Or you could have set your default to ranger's remote directory, by entering:

```
$ set default ranger"dwatts password"::[udd.dwatts]
```

Once the default is set, you could enter :

```
delete strm.tst7.*
```

Note: Even though UNIX systems have no concept of version number, you must use ";*" to delete a file using the square bracket syntax, as it is required by VMS.

directory - Listing Files

You can get a directory listing for directories residing on your ESV Workstation. You can also use the **/FULL** or **-full** options. For example,

```
Vax> dir ranger"dwatts"::"/circuit/corrigan/*.*"

Directory RANGER"dwatts password"::
"/circuit/corrigan/a_vfc.tyt" "/circuit/corrigan/crash.pad"
"/circuit/corrigan/dap.h" "/circuit/corrigan/dap.pad"
"/circuit/corrigan/forfix.trt"
"/circuit/corrigan/forfix.tst1"
"/circuit/corrigan/modified.bak" "/circuit/corrigan/ncp.out"
"/circuit/corrigan/ncp.outi" "/circuit/corrigan/read.me3"
"/circuit/corrigan/recbinfix1.out"
"/circuit/corrigan/rename.pad"
"/circuit/corrigan/strm.t" "/circuit/corrigan/t.tyt"
"/circuit/corrigan/u.obj" "/circuit/corrigan/vartst7.tst1"
Total of 6 files.
```

Or, alternately, specify the VMS **/FULL** option, as in the following example, which also uses the square brackets.

```
Vax> dir ranger"dwatts password"::"
      [circuit.corrigan]user_data"/full
Directory RANGER"dwatts password"::
[circuit:corrigan]user_data" File ID: None
Size: 2/2 Owner: [0,0]
Created: 3-JUN-1988 20:03 Revised: 27-AUG-1988 20:21 (1)

Expires:  Backup:  File organization: Sequential
File attributes: Allocation: 2, Extend: 0, Global buffer
      count: 0
Version limit: 0, Contiguous best try
Record format: Fixed length 44 byte records
Record attributes: None
Journaling enabled: None
File protection: System:RWED, Owner:RWED, Group:RWED,
      World:RWED
Access Cntrl List: None
Total of 1 file, 2/2 blocks
```

Note: VMS will see only files with extensions as VMS has no concept of files without extensions.

rename - Renaming Files

You can only rename a file remotely if the file is fully specified. For example,

```
Vax> rename ranger"user psswd"::"/old_filename"
"/new_filename"
```

Or, if you want to complete a wildcard rename, you must use the square bracket syntax.

submit - Submitting Files to a Batch Queue

You can submit a file for batch execution by using the VAX **SUBMIT/REMOTE** command. The file must currently exist on the ESV Workstation and must have a batch queue. For example,

```
Vax> submit/remote station"user psswd"::[usr.dand]script.exe
```

print - Printing Files

You can print a file on the ESV Workstation by issuing a VAX **PRINT/REMOTE** command. The file to be printed must currently exist on the ESV Workstation, and a print batch queue must exist also. For example,

```
Vax>print/remote station"user psswd"::[usr.dand.test]results
```

Logging In to a Remote System

The **sethost** command creates a virtual terminal connection from the ESV Workstation to a remote DECnet node. The remote node is specified with the node argument using the node name.

This remote login capability is similar to **rlogin**; however, **sethost** does not attempt to log you in at the remote node. Rather, once **sethost** has accessed the remote node, you then log in to the remote node just as you would if logging on directly. In the case of a VAX system, the VAX's banner and login prompt are displayed, as in:

```
$ sethost vaxnet

Username: jones
Password:(not echoed)
DIGITAL EQUIPMENT VAX SYSTEM
Last interactive login on Thursday, 25-AUG-1988 08:28
Last non-interactive login on Thursday, 25-AUG-1988 08:09
Vaxnet>
```

You are now logged in to the remote VAX and your terminal is considered a VT100. The **sethost** command is described in greater detail in Chapter 5, "Command Summary."

Logging off of the remote system returns you to the ESV Workstation's prompt.

Logging In from a Remote System

You can use the VMS **SET HOST** command to log in to your ESV Workstation from a DECnet node. The **sethost_server /dvta** allows accesses to the ESV Workstation as a virtual terminal. For example,

```
Vax> set host station
Username: dand
Password:
$
```

Using dnamail

You can send mail to network nodes using the **dnamail** command. The following example shows the user interaction for the **dnamail** command.

```
$ dnamail
Node: 1.25
To: dand
Subject: Project meeting
Waiting for connection to peer...
Enter your message below.  Press <EOF> when complete or
<ABORT> to quit.
Don't forget the project meeting today at 3:00.
<EOF>
Waiting for mail sending confirmation...
Mail Sent
$
```

In specifying the node you can use a node name or number. The node number or name can be with or without the "::".

Note: **<EOF>** is the standard UNIX **<EOF>** which is generated by pressing the CONTROL and D keys simultaneously.

3. Programmer's Information

This section describes the *ES/Dnet Network Procedure Call Library* which provides a programming interface to the DECnet local area network. ES/Dnet provides a set of network functions calls permitting user-written applications to:

- Request or accept a connection from a VAX/VMS system,
- Read or write across that connection,
- Close the connection.

This interface allows a user-written cooperating process to exchange data with a user program on a remote VAX node. The user-written programs use these network function calls to interface to the *ES/Dnet agent* (described below).

This chapter contains the following:

- A general discussion of the *ES/Dnet Network Procedure Call Library*
- Descriptions of the Network Procedure calls
- Sample programs using the Network Procedure calls

Network Procedure Calls

/usr/etc/esdnet /dna/lib/libdna.a

The Network Procedure calls are in a library located in ~~/usr/lib/libdna.a~~. In order for an application to execute these calls, the application only has to be linked to the library. The Network Procedure calls available with ES/Dnet include:

NETSTART	Determines if ES/Dnet application was started by an incoming network request.
NOPEN	Requests or accepts a DECnet task-to-task connection.
NCONNECT	Requests a management connection with the ES/Dnet system.
NCLOSE	Requests a termination for a specific connection.
NREAD	Reads data from a connection.
NWRITE	Writes data across a connection.
NIWRITE	Writes interrupt data across a connection.
NIOCTL	Performs management requests for the ES/Dnet system.

Network Procedure Call Format

The following sections describe the Network Procedure calls used to create applications that communicate with other systems across DECnet. These discussions include functional descriptions of each call and list its arguments. Each call is presented in this format:

Name	Displays the name of the call.
Syntax	Shows how the call is defined. It provides information about parameters and types required by the calling program.
Description	Provides a detailed description about the call.
Return Value	Lists the possible return values, including values that indicate only success or failure.
Errors	Describes the returned value that indicates an error has occurred and the external variable error has been set to a code that defines the error.

Variable Types

All number formats are decimal unless otherwise noted. The variable types used in these procedures are defined as:

Variable Type	Description
integer	signed value (at least 16 bits)
word	unsigned 16-bit value
char, byte	unsigned 8-bit value
dword	unsigned 32-bit value
string	null-terminated text string
packet	a structure for each specific type of network information
array of pointers	single dimensional array of pointers
array of bytes	single dimensional array of unsigned 8-bit values

Network Procedure Call Quick Reference

Use the following table for a quick reference for each of the Network Procedure calls.

Table 3-1. ES/Dnet Network Procedure Calls

Call	Arguments	Type	Pass By
NETSTART	**argc**	integer	reference
	argv	array of pointers	reference
	sysnet	array of bytes	reference
NOPEN	**agent_name**	string	reference
	agent_packet	packet	reference
	agent_size	integer	value
	rcv_size	integer	value
	max_data_size	integer	reference
	err_code	integer	reference
NCONNECT	**agent_name**	string	reference
	err_code	integer	reference
NCLOSE	**agent_id**	integer	value
	disc_data	array of bytes	reference
	length	integer	value
	err_code	integer	reference
NREAD	**agent_id**	integer	value
	buff	array of bytes	reference
	buff_size	integer	value
	pend	integer	value
	err_code	integer	reference
NWRITE	**agent_id**	integer	value
	buff	array of bytes	reference
	buff_size	integer	value
	err_code	integer	reference
NIWRITE	**agent_id**	integer	value
	buff	array of bytes	reference
	buff_size	integer	value
	err_code	integer	reference
NIOCTL	**agent_id**	integer	value
	packet	packet	reference
	packet_size	integer	value
	err_code	integer	reference

NETSTART

Syntax:

```
int NETSTART (argc, argv, sysnet)
int  * argc;
char * argv[],
* sysnet;       /* Array of 8+1 bytes */
```

Description:

An ES/Dnet application program uses **NETSTART** to determine if it was started due to an incoming network request. **NETSTART** will return a non-zero value if the application was started by the network and will put an 8-character null-terminated string into **sysnet**. When this happens, the application should set the ES/Dnet access type to **NET_INCOMING** and copy the **sysnet** value into the ES/Dnet agent packet.

If **NETSTART** returns a zero value, the object was invoked by a user and should operate accordingly.

NOPEN

Syntax:

```
int NOPEN (agent_name, agent_packet, agent_size, rcv_size,
max_data_size, err_code)
char      * agent_name;
T_DECNET * agent_packet;
int         agent_size,
rcv_size,
* max_data_size,
* err_code;
```

Description:

NOPEN requests a connection or accepts a connection for a user program to a specific DECnet network object. **NOPEN** returns the **agent_id**, which is used for all future **NREAD**, **NWRITE**, **NIWRITE**, and **NCLOSE** function calls. After the connection has been established, the application can perform reads and writes across the connection.

The **agent_name** identifies the required agent to which the connection is made. The only valid value is **DECNET**. The **agent_packet** is the address of the agent open packet that is specific to the agent being requested. The **agent_size** is the size in bytes of the agent open packet. The **rcv_size** is the maximum size of the user receive data buffer in bytes. The **max_data_size** is the maximum size of the returned agent receive buffer.

The service packet is used when opening a connection to the DECnet server. The **access_type** field identifies whether the open connection request is for an outgoing connection or an incoming connection.

The ES/Dnet agent validates the access control of incoming requests. This is in contrast to the way VMS DECnet handles incoming connection requests when the specified task is already resident.

All strings are null-terminated. The user connect data is a counted byte string; *i.e.*, the first byte identifies the number of bytes that follow. The actual byte values of the connect data are uninterpreted by the ES/Dnet agent and can contain any value. This data is not assumed to be null-terminated and is handled as raw binary 8-bit values.

The typedef of the service packet structure is defined in **ki_deknet.h**.

Return Value:

The call returns the **agent_id** as an integer from 1 to 255. A "-1" is returned for the **agent_id** if there is an error. The **agent_id** is used for all future **NREAD**, **NWRITE**, and **NCLOSE** function calls.

Errors:

A "0" is returned in the **err_code** field if there is no error. The error code returned could be either a system or an ES/Dnet error code.

NCONNECT

Syntax:

```
int NCONNECT (agent_name, err_code)
char *agent_name;
int  *err_code;
```

Description:

NCONNECT creates a management connection with the local ES/Dnet agent. Using this management connection, a program may do the following:

- Read and/or zero various network counters
- Read information from the active node database
- Add entries to the active node database
- Read/zero/add-to the proxy database
- Read/zero/add-to the object database
- Terminate execution of the ES/Dnet agent

Return Value:

The call returns the **agent_id** as an integer from 1 to 255. A "-1" is returned for the **agent_id** if there is an error. The **agent_id** is used for all future **NIOCTL** and **NCLOSE** function calls.

Errors:

A "0" is returned in the **err_code** field if there is no error. The error code returned could be either a system or an ES/Dnet error code.

NCLOSE

Syntax:

```
int NCLOSE (agent_id, disc_data, length, err_code)
int    agent_id;
char * disc_data;
int    length,
* err_code;
```

Description:

NCLOSE terminates the use of a particular connection to a specific ES/Dnet agent. The **agent_id** identifies the agent connection point to be terminated. The **disc_data** is user specified data to be sent to the agent when the link is closed. The **length** is the length of the user data.

Return Value:

The function returns an error code value. It is the same value as **err_code**.

Errors:

A "0" is returned in the **err_code** field if there is no error. The error code returned could be either a system or an ES/Dnet error code.

NREAD

Syntax:

```
int      NREAD (agent_id, buff, buff_size, pend, err_code)
int      agent_id;      /* Connection to read from */
char    * buff;          /* Where to put data       */
int      buff_size,     /* Byte size of buffer     */
         pend            /* T: Wait for data        */
         * err_code;    /* System or ES/Dnet error  */
```

Description:

NREAD waits for data if there is no data currently ready. The function returns either the number of bytes read or a "-1" if the operation could not be performed. If the user buffer is too small, the received data is truncated. The actual function value is the data buffer size and the **err_code** variable is set to **P_TRUNCATE**.

The **agent_id** identifies the agent connection point from where data is to be read. The **buff** is the received data. The **buff_size** is the number of user bytes available in the buffer. **pend** is set to true (non-zero) if the task should wait in the event there is no data available.

The user buffer must be at least 16 bytes to contain any possible interrupt data that may be returned.

Return Value:

NREAD returns the actual number of bytes read. A "-1" indicates a fatal error. Note that a "0" is a valid byte count.

Errors:

A "0" is returned in the **err_code** field if there is no error. The error code returned could be either a system or an ES/Dnet error code.

An error code of **P_IDATA** is returned if interrupt data has been returned to the user instead of normal data.

An error code of **P_DIDATA** is returned if disconnected data has been returned.

NWRITE

Syntax:

```
int       NWRITE (agent_id, buff, buff_size, err_code)
int       agent_id;          /* Connection to read from */
char    * buff;               /*Where to put data        */
int       buff_size,         /* Byte size of buffer      */
           * err_code;       /* System or ES/Dnet error  */
```

Description:

NWRITE writes data to an agent connection point.

NWRITE returns the actual number of bytes written or a "-1" if the operation could not be performed. If the user attempts to write more data than the ES/Dnet agent buffer size, the **err_code** variable is set to **P_TRUNCATE**. The **agent_id** identifies the agent's connection point where data is to be written. The **buff** is the data to write. The **buff_size** is number of user bytes in buffer.

Return Value:

The function returns the actual number of bytes written to the agent connection point. A "-1" indicates a fatal error.

Errors:

A "0" is returned in the **err_code** field if there is no error. The error code returned could be either a system or an ES/Dnet error code.

NIWRITE

Syntax:

```
int     NIWRITE (agent_id, buff, buff_size, err_code)
int     agent_id;
char    *buff;
int     buff_size;
int     *err_code;
```

Description:

NIWRITE writes interrupt data to an agent connection point.

NIWRITE returns the actual number of bytes written or a "-1" if the operation could not be performed. If the user attempts to write more data than the receiver buffer size, the **err_code** variable is set to **P_TRUNCATE**. The **agent_id** identifies the agent's connection point where data is to be written. The **buff** is the data to write. The **buff_size** is number of user bytes in buffer and can be a maximum of 16 bytes.

Return Value:

The function returns the actual number of bytes written to the agent connection point. A "-1" indicates a fatal error.

Errors:

A "0" is returned in the **err_code** field if there is no error. The error code returned could be either a system or an ES/Dnet error code.

NIOCTL

Syntax:

```
int     NIOCTL (agent_id, packet, packet_size, err_code)
int     agent_id,          /* Connection to read from */
        * packet,          /* Control packet          */
        packet_size,  /* Byte size of packet     */
        * err_code;      /* System or ES/Dnet error  */
```

Description:

NIOCTL is used to perform management operations on the local ES/Dnet agent. User programs may read values from the active ES/Dnet databases, optionally zero counters after reading, modify/set items in the active ES/Dnet databases, or remove items from the active ES/Dnet databases. Another use for **NIOCTL** is to perform the **NIO_SELECT** operation. This operation allows a program to wait until data is received on any of up to 16 **agent_id**'s.

The ES/Dnet header file defines the structures that are used with this call. The valid operations are:

NIO_EXECUTOR	Get/Set Executor Characteristics
NIO_NODE	Get/Zero/Set/Clear Node Entries
NIO_OBJ	Get/Set/Clear Object Entries
NIO_PROXY	Get/Set/Clear Proxy Entries
NIO_STATE	Set/Clear Executor State (Start/Stop)
NIO_SCOPE	Get/Set/Clear Scoping
NIO_CIRCUIT	Get/Zero Circuit/Line Counters
NIO_SELECT	Get Selection of **agent_id**'s

Return Value:

The function returns the number of bytes returned in the structure. A "-1" is returned to indicate a fatal error.

Errors:

A "0" is returned in the **err_code** field if there is no error. The error code returned could be either a system or an ES/Dnet error code.

Program Examples

The following examples use the ES/Dnet Network Procedures calls described previously in this chapter. Two sets of programs detail writing and reading data and interpreting data: **CON129**, **OBJ129**, **CON131**, and **OBJ131**. The last program, **showcir**, details the network control interface. This program shows how to gather network statistics with ES/Dnet.

The programs are named in a standard way. The first three characters are either **CON** or **OBJ**. Those that are **CON** are programs that the user would run to initiate a DECnet connection. Those that begin with **OBJ** are DECnet objects that would wait for an incoming connection and then proceed. The last part of the name is a 3-digit number which identifies the DECnet Object Number being used. The numbers that these test programs use are in the "user" range of DECnet object numbers. DEC has object numbers 1 through 127 reserved.

CON129 and OBJ129

This pair of programs demonstrates use of the data channel. **CON129** prompts the user to enter lines of text. These lines of text will be sent to **OBJ129**, which, in turn, displays the received lines on its output file. **CON129** is terminated when the user enters **EOF** (CONTROL-Z or CONTROL-D).

The following example shows an outgoing connection to a network application. This application writes normal data.

CON129 Program

```
/* Outgoing  */
#include <stdio.h>
#include "standards.h"
#include "ki_ESDnet.h"
#list
#define BUFF_SIZE    1024
main (argc, argv)
int    argc;
char * argv[];
{
     /* External Modules */
      extern int NCLOSE(),
     NOPEN(),
     NWRITE();
      /* Local Variables */
     T_DECNET agent_pkt;
     int  agent_id,
          bytes,
          err_code,
          len,
           max_size;
     char   * cp;
     char     buff[BUFF_SIZE],
          node[128],
          user[128],
          pass[128],
          acct[128];
/* Executable Code */
printf ("CON129 started\n");
printf ("Enter node to connect to: ");
cp = gets (node);
if (not cp) then
     {
     printf ("Nothing entered\n");
     return;
     }
endif
printf ("Enter username: ");
cp = gets (user);
if (not cp) then
     {
     printf ("Nothing entered\n");
     return;
```

```
    }
endif
printf ("Enter password: ");
cp = gets (pass);
if (not cp) then
    {
    printf ("Nothing entered\n");
    return;
    }
 endif

printf ("Enter account:  ");
acct[0] = ' ';
cp = gets (acct);

agent_pkt.access_type = OUTGOING;
agent_pkt.ltype = DECNET_TASK;
agent_pkt.rtype = DECNET_OBJ;
agent_pkt.robj_num    = 129;
strcpy (agent_pkt.lname, "OBJ129");
strcpy (agent_pkt.rnode, node);
strcpy (agent_pkt.auser, user);
strcpy (agent_pkt.apass, pass);
strcpy (agent_pkt.aacct, acct);
strcpy (&agent_pkt.auserdata[1], "Data Test");
agent_pkt.auserdata[0] = strlen (&agent_pkt.auserdata[1]);

/* NOPEN */

agent_id = NOPEN ("DECnet", &agent_pkt, sizeof(agent_pkt),
    BUFF_SIZE, &max_size, &err_code);
if (err_code) then
    perror ("CON129: Calling NOPEN0);

    else
        {
        printf ("Agent_id= 0x%04x\n", agent_id);
        printf ("Max size= %d\n",     max_size);

        repeat
        {
        printf ("Enter text: ");
        cp = gets (&buff);
        if (cp) then
```

```
        {
        len = strlen (cp);

/* NWRITE */

        bytes = NWRITE (agent_id, &buff, len, &err_code);
        if (err_code) then
        perror ("CON129 calling NWRITE0);
        elseif (bytes not_equals len) then
        printf ("CON129 only %d bytes written\n", bytes);
        endif
        }
        endif
        }
         until (cp equals 0);

/* NCLOSE */

        NCLOSE (agent_id, "", 0, &err_code);
        }
     endif

    printf ("Execution complete\n");
    }
```

OBJ129 Program

The following example shows an incoming connection from a network application.

```
/* Incoming */

#include <stdio.h>

#include "standards.h"
#include "ki_ESDnet.h"
#list

#define BUFF_SIZE    1024

#define flush_out    fflush(stdout)
#define flush_err    fflush(stderr)

main (argc, argv)
int    argc;
char * argv[];
 {
    /* External Modules */
   extern int   NCLOSE(),
    NOPEN(),
    NWRITE();

    /* Local Variables */
   char     buff [BUFF_SIZE];
   T_DECNET agent_pkt;
   intagent_id,
bytes,
err_code,
len,
max_size;
   char   * cp;

    /* Executable Code */

    printf ("OBJ129 started0);
   printf ("Argc= %d0, argc);

/* NETSTART  */

   if (NETSTART (argc, argv, &agent_pkt.net_in[0])) then
```

```
{
printf ("Started due to incoming connection0);
printf ("SYSNET= '%s'0, agent_pkt.net_in);
agent_pkt.access_type = NET_INCOMING;
}
   else
agent_pkt.access_type = INCOMING;
   endif

   agent_pkt.ltype = DECNET_TASK;
   agent_pkt.lobj_num    = 129;
   strcpy (agent_pkt.lname, "OBJ129");

   printf ("Waiting for connect...");
   flush_out;

/* NOPEN */

   agent_id = NOPEN ("DECnet", &agent_pkt,
  sizeof(agent_pkt), BUFF_SIZE,
   &max_size, &err_code);
   if (err_code) then
perror ("0BJ129: Calling NOPEN0);
   else
{
printf ("*** Connection received ***0);
printf ("Agent_id= 0x%04x0, agent_id);
printf ("Max size= %d0,     max_size);

printf ("Access control:0);
printf ("   Remote Process: '%s'0,   agent_pkt.ruser);
printf ("   Username: '%s'0,   agent_pkt.auser);

printf ("   Password: '%s'0,   agent_pkt.apass);
printf ("   Account:  '%s'0, agent_pkt.aacct);
printf ("Connect Data: %d bytes0,
  agent_pkt.auserdata[0]);

bdump (stdout, &agent_pkt.auserdata[1],
  agent_pkt.auserdata[0], 0);

repeat
   {
   printf ("Waiting for data...");
```

```
    flush_out;

 /* NREAD */

    bytes = NREAD (agent_id, buff, BUFF_SIZE, TRUE,
    &err_code);
   if (not err_code) then
{
printf ("  Received %d bytes\n", bytes);
bdump (stdout, &buff, bytes, 0);
}

   elseif (err_code equals P_IDATA) then
{
printf ("  Received %d bytes of INTERRUPT data\n", bytes);
bdump (stdout, &buff, bytes, 0);
err_code = 0;
}
   else
perror ("0BJ129 calling NREAD0);
   endif
    }
 until (err_code);

 /* NCLOSE */

NCLOSE (agent_id, "", 0, &err_code);
 }
    endif

    printf ("OBJ129 Execution complete\n");
    flush_out;
 }
```

Code Walkthrough of CON129.C

This program first asks for the node that the user wishes to connect to. The user may enter a DECnet node number, like **1.2**, or a node name, like **MVAX1**. The user is then prompted to enter the username, password, and account to be used for access control checking. The account field is not used by VMS or ES/Dnet but is retained for VMS historical reasons.

This information is then copied into the connection packet **agent_pkt** and sets other values that identify the connection being requested. The details follow:

- `agent_pkt.access_type = OUTGOING;`

  (Says that this is an outgoing connection.)

- `agent_pkt.ltype = DECNET_TASK;`

  (Not necessary for this type of connection but says that this program is a "named" task.)

- `agent_pkt.rtype = DECNET_OBJ;`

  (The type of object to connect to: A "numbered" object.)

- `agent_pkt.robj_num = 129;`

  (The DECnet object number to connect to.)

- `strcpy (agent_pkt.lname, "CON129");`

  (The task name of this program.)

- `strcpy (agent_pkt.rnode, node);`

  (The node to connect to.)

- `strcpy (agent_pkt.auser, user);`

  (Username for access control.)

- `strcpy (agent_pkt.apass, pass);`

  (Password for access control.)

- `strcpy (agent_pkt.aacct, acct);`

  (Account for access control.)

- `strcpy (&agent_pkt.auserdata[1], "Data Test");`

  (Connect Data , maximum of 16 bytes)

- `agent_pkt.auserdata[0] = strlen (&agent_pkt.auserdata[1]);`

  (The length of the Connect Data.)

The program issues an **NOPEN()** call to open the DECnet logical link with the DECnet object specified. If the logical link cannot be made, **err_code** is set to a non-zero number, and **CON129** will display the error code and exit.

When the logical link is established, **CON129** displays the ES/Dnet **agent_id**, the maximum size that the local ES/Dnet allows for I/O, and any received connect data.

CON129 enters a loop where it prompts for a line of text to be input. The entered text is written to the remote object. If the user enters an **EOF** (usually CONTROL-Z, but UNIX shells may have this as CONTROL-D), the program exits the loop.

CON129 calls **NCLOSE()** to close the logical link and exits.

Code Walkthrough of OBJ129.C

CON129 connects to this program. When it starts, it prints out information describing its arguments and its environment. The code builds the connection packet to identify itself. A call is made to **NETSTART()** to determine if the program was invoked by a user or automatically by ES/Dnet. If it was started by ES/Dnet, **OBJ129** sets the **access_type** field to **NET_INCOMING** and sets the object number to **129** with the modifier **DN_KEEPALIVE**. An object uses the **NET_INCOMING** access type to connect to the logical link that caused ES/Dnet to start the program. The **DN_KEEPALIVE** modifier comes into play later on subsequent **NOPEN()**s. It causes the **NOPEN()** request to stay around only for a limited period of time before ES/Dnet terminates the **NOPEN()** with an error saying that nobody has requested a new connection.

If **OBJ129** is started by a user, the **access_type** is set to **INCOMING**. This tells ES/Dnet to wait for an incoming connection. The object number **lobj_num** is set to **129** without any modifiers.

OBJ129 completes building the connection packet and calls ES/Dnet via **NOPEN()**. When the incoming connection is received (if this was a **NETSTART**ed object it should return immediately), **NOPEN()** returns. If any errors occur, the program displays the error and exits. For completed connections, more information about the logical link is displayed. The program enters a loop that reads data from the network and displays the received data in hex/ASCII. When **NREAD()** errors, which is the normal condition when the logical link is closed, the loop terminates and **OBJ129** closes the link.

The access type is set to **INCOMING** and the program waits the next connection request. Remember, if this was network started, it

waits only a limited time before ES/Dnet terminates the **NOPEN()** with an error.

Note that objects running with **DN_KEEPALIVE** only receive connections for the same username that is running the object. Objects that do not run with **DN_KEEPALIVE** accept connections for any valid user.

CON131 and OBJ131

These programs are similar to **CON129** and **OBJ129**, except that these demonstrate the usage of the interrupt channel. **CON131** terminates when the user enters an **EOF** (CONTROL-Z or CONTROL-D). **OBJ131** terminates automatically

CON131 Program

```
/* COPYRIGHT 1988 KI RESEARCH, INC.  ALL RIGHTS RESERVED */
/* Outgoing */


#include <stdio.h>

#include "standards.h"
#include "ki_ESDnet.h"
#list

#define flush_out     fflush(stdout)
#define BUFF_SIZE     1024

main (argc, argv)
int     argc;
char * argv[];
{
/* External Modules */
     extern int NCLOSE(),
     NOPEN(),
     NIWRITE();

/* Local Variables */
      char  buff [BUFF_SIZE];

     T_DECNET  agent_pkt;
     int agent_id,
     bytes,
      err_code,
     len,
     max_size;

     char     * cp;
     charnode[128];
     charuser[128],
     pass[128],
     acct[128];

/* Executable Code */

printf ("CON131 started\n");

printf ("Enter node to connect to: ");
```

```
cp = gets (node);
if (not cp) then
     {
     printf ("Nothing entered\n");
     return;
     }
endif

printf ("Enter username: ");
cp = gets (user);
if (not cp) then
     {
     printf ("Nothing entered\n");
     return;
   }
endif

printf ("Enter password: ");
cp = gets (pass);
if (not cp) then
     {
     printf ("Nothing entered\n");
     return;
     }
endif

printf ("Enter account:  ");
acct[0] = ' ';
cp = gets (acct);


agent_pkt.access_type = OUTGOING;
agent_pkt.ltype = DECNET_TASK;
agent_pkt.rtype = DECNET_OBJ;
agent_pkt.robj_num    = 131;
strcpy (agent_pkt.lname, "OBJ131");
strcpy (agent_pkt.rnode, node);
strcpy (agent_pkt.auser, user);
strcpy (agent_pkt.apass, pass);
strcpy (agent_pkt.aacct, acct);
strcpy (&agent_pkt.auserdata[1], "Interrupt Test");
agent_pkt.auserdata[0] = strlen (&agent_pkt.auserdata[1]);

/* NOPEN */
```

```
agent_id = NOPEN ("DECnet", &agent_pkt, sizeof(agent_pkt),
    BUFF_SIZE,
&max_size, &err_code);

if (err_code) then
    perror ("CON131: Calling NOPEN0);
else
    {
    printf ("Agent_id= 0x%04x\n", agent_id);
    printf ("Max size= %d\n",     max_size);

    repeat
     {
     printf ("Enter text: ");
     cp = gets (&buff);
     if (cp) then
         {
         len = strlen (cp);

/* NWRITE */

         bytes = NIWRITE (agent_id, &buff, len, &err_code);
         if (err_code) then
         perror ("CON131 calling NIWRITE");
         elseif (bytes not_equals len) then
         printf ("CON131 only %d bytes written\n", bytes);
         endif
         }
     endif

     }
    until (cp equals 0);

/* NCLOSE */

    NCLOSE (agent_id, "", 0, &err_code);
    }
endif

printf ("Execution complete\n");
}
```

The following program is an example of an incoming connection from a network application.

```
/* Incoming */


#include <stdio.h>

#include "standards.h"
#include "ki_ESDnet.h"
#list

#define BUFF_SIZE    1024

#define flush_out    fflush(stdout)
#define flush_err    fflush(stderr)

main (argc, argv)
int    argc;
char * argv[];
 {
    /* External Modules */
   extern int   NCLOSE(),
    NOPEN(),
    NWRITE();

    /* Local Variables */
   char     buff [BUFF_SIZE];
   T_DECNET agent_pkt;
   intagent_id,
bytes,
err_code,
len,
max_size;
   char   * cp;



    /* Executable Code */

    printf ("OBJ131 started0);
   printf ("Argc= %d0, argc);

/* NETSTART  */
```

```
  if (NETSTART (argc, argv, &agent_pkt.net_in[0])) then
{
printf ("Started due to incoming connection0);
printf ("SYSNET= '%s'0, agent_pkt.net_in);
agent_pkt.access_type = NET_INCOMING;
}
   else
agent_pkt.access_type = INCOMING;
   endif


   agent_pkt.ltype = DECNET_TASK;
   agent_pkt.lobj_num    = 131;
   strcpy (agent_pkt.lname, "OBJ131");

   printf ("Waiting for connect...");
   flush_out;

/* NOPEN */
   agent_id = NOPEN ("DECnet", &agent_pkt,
   sizeof(agent_pkt), BUFF_SIZE,
   &max_size, &err_code);

   if (err_code) then
perror ("0BJ131: Calling NOPEN0);
   else
{
printf ("*** Connection received ***0);
printf ("Agent_id= 0x%04x0, agent_id);
printf ("Max size= %d0,     max_size);

printf ("Access control:0);
printf ("   Remote Process: '%s'0,   agent_pkt.ruser);
printf ("   Username: '%s'0,   agent_pkt.auser);
printf ("   Password: '%s'0,   agent_pkt.apass);

printf ("   Account:  '%s'0, agent_pkt.aacct);
printf ("Connect Data: %d bytes0,
   agent_pkt.auserdata[0]);

bdump (stdout, &agent_pkt.auserdata[1],
   agent_pkt.auserdata[0], 0);

repeat
```

```
    {
    printf ("Waiting for data...");
    flush_out;

 /* NREAD */

    bytes = NREAD (agent_id, buff, BUFF_SIZE, TRUE,
    &err_code);
   if (not err_code) then
{
printf ("  Received %d bytes\n", bytes);
bdump (stdout, &buff, bytes, 0);
}

   elseif (err_code equals P_IDATA) then
{
printf ("  Received %d bytes of INTERRUPT data\n", bytes);
bdump (stdout, &buff, bytes, 0);
err_code = 0;
}
   else
perror ("0BJ131 calling NREAD0);
   endif
    }
 until (err_code);

 /* NCLOSE */

NCLOSE (agent_id, "", 0, &err_code);
 }
    endif

    printf ("OBJ131 Execution complete\n");
    flush_out;
 }
```

showcir Program

This file contains a program that connects to the ES/Dnet agent. It issues **NIOCTL()**s to get information on the circuits.

```
/*************************************************************
     *

COPYRIGHT 1988 KI RESEARCH, INC.  ALL RIGHTS RESERVED

*************************************************************
showcir.c
*************************************************************/

#include <stdio.h>

#include "standards.h"
#include "ki_ESDnet.h"
#include "ki_ESDneterr.h"
#include "ugetswitch.h"

#define say printf

/**********************
* Private Static Data *
**********************/
PRIVATE int full_list  = FALSE,      /* T: Show all
     information  */
known_list = FALSE,      /* T: Show all circuits      */
zero_list  = FALSE;      /* T: Zero counters after reading */


/*******************
* External Modules *
*******************/
extern char * ctime();  /* Generate ASCII Date/Time */
extern long   time();
extern void   copyright_notice(),
  show_timesec_val();


main (argc, argv)
int  argc;
char *argv[];
```

```
{
   /* External Modules */
   extern int NCLOSE(),
  NCONNECT(),
  NIOCTL();

   /* Local Variables */
   long     date_time,
up_time;
   char   * date_time_cp,
    * ex_name;
   intagent_id,
bytes,
err_code,
num_cir  = 0,
ok;
   byte   * ha,
    * sa;
   char     cir_pattern[81];
   T_NIO_EXECUTOR nioex;
   T_NIO_NODE     niond;
   T_NIO_CIRCUIT  niocir;


   /* Executable Code */

   if (argc > 1) then
{
printf ("No command line arguments are allowed\n");
do_usage();
return;
}
   end_if

   strcpy (cir_pattern, "+");
   agent_id = NCONNECT ("DECnet", &err_code);

   if (err_code) then
kperror ("SHOWCIR: Calling NCONNECT\n");
   else

{
nioex.opcode   = NIO_EXECUTOR  NIO_GET;
NIOCTL (agent_id, &nioex, sizeof(nioex), &err_code);
```

```
if (not err_code) then
   {
   sprintf (niond.name, "%d.%d", nioex.area, nioex.node);
   niond.opcode    = NIO_NODE  NIO_GET;
   niond.select_by = SELECT_FIRST;
   NIOCTL (agent_id, &niond, sizeof(niond), &err_code);
   if (err_code) then ex_name = "";
     else ex_name = &niond.name[0];

   time ( &date_time );
   date_time_cp = ctime ( &date_time );
   say ("DECnet circuit counters for local node %d.%d %s on
    %s",
    nioex.area, nioex.node, ex_name, date_time_cp);

   up_time= date_time - nioex.up_time;
   date_time    = nioex.up_time;
   date_time_cp = ctime ( &date_time );
   say ("Up since: %sUp time: ", date_time_cp);
   if (up_time > 601) then show_timesec_val (stdout,
    up_time, FALSE);
    else printf (" %d seconds", up_time);
   end_if
   say ("\n\n");
   }
else
   kperror ("Calling NIOCTL:NIO_EXECUTOR");
end_if
if (zero_list) then niocir.opcode = NIO_CIRCUIT
    NIO_GETNZERO;
   else niocir.opcode = NIO_CIRCUIT  NIO_GET;

niocir.select_by = SELECT_FIRST;
repeat
   {
   strcpy (niocir.name, cir_pattern);     /* Pattern to
    search on */

   NIOCTL (agent_id, &niocir, sizeof(niocir), &err_code);
   if (not err_code) then
{
num_cir++;

printf (
```

```
"Circuit Counters\n\
\tdata_bytes_rcvd: %8ld, data_bytes_sent: %8ld\n\
\tmc_bytes_rcvd:   %8ld, mc_bytes_sent:   %8ld\n\
\tbytes_rcvd:%8ld, bytes_sent:%8ld\n",
niocir.data_bytes_rcvd, niocir.data_bytes_sent,
niocir.mc_bytes_rcvd,   niocir.mc_bytes_sent,
niocir.bytes_rcvd,niocir.bytes_sent);

printf ("\n\
\tdat_frames_rcv:  %8ld, dat_frames_sent: %8ld\n\
\tmc_frames_rcvd:  %8ld, mc_frames_sent:  %8ld\n\
\tframes_rcvd:     %8ld, frames_sent:     %8ld\n",
niocir.frames_rcvd - niocir.mc_frames_rcvd,
niocir.frames_sent - niocir.mc_frames_sent,
niocir.mc_frames_rcvd, niocir.mc_frames_sent,
niocir.frames_rcvd,    niocir.frames_sent);

printf ("\n\
\tsend_1_collision: %4ld, send_multi_coll:   %4ld\n\
\tsend_failure:     %4ld, collision_failure: %4ld,
    no_carrier:     %4ld\n\
\tloss_of_cts:%4ld, dma_underrun:%4ld\n",
niocir.send_1_collision, niocir.send_multi_collision,
niocir.send_failure,     niocir.collision_failure,
    niocir.no_carrier,
niocir.loss_of_cts,niocir.dma_underrun);

printf ("\
\ttoo_many_retries: %4ld, no_sys_buffer:     %4ld,
    no_user_buffer: %4ld\n",
niocir.too_many_retries,
niocir.no_sys_buffer,
niocir.no_user_buffer);
}
   end_if

   niocir.select_by = SELECT_NEXT;
   }
until (err_code);

if (err_code not_equals P_NO_DATA) then
   kperror ("Calling NIOCTL:NIO_CIRCUIT");
end_if

NCLOSE (agent_id, "", 0, &err_code);
```

```
if (num_cir) then printf ("\n%15s Total of %d circuits.\n",
     "", num_cir);
   else printf ("\n%15s No circuits.\n", "");
}
   end_if
}
```

4. Network Management

This chapter provides examples and explanations of the ES/Dnet Network Management facilities, including **pconfig**'s node, object, and LAN device configuration management.

This chapter is divided into these main topics:

- "Using **pconfig**" details the **pconfig** utility and its use in configuring the ES/Dnet endnode.
- "Using the Network Command Facilities" describes the network command utilities, including **shownet**, **showcir**, **showproxy**, **showobj**, **loop**, **mirror**, **updb**, and **dna_stop**. Network command utilities display network configuration and network statistics. They test node connectivity, update the network databases, and halt the network.
- "Using the **scope** Facility" shows you how to gather raw network data packets.

This chapter is written for network managers who are familiar with the networking environment. Some familiarity with DECnet in particular will also be helpful, as this chapter describes how the ES/Dnet system uses ES/Dnet to participate on the DECnet local area network.

Using pconfig

The following sections explain the **pconfig** utility's configuration commands.

Note: Since **pconfig** uses a full screen display, it must be run from **xterm** and not from the boot-up console display.

Database Locations

pconfig allows you to make modifications to databases on your local node. The following databases can be modified with **pconfig**:

executor.db	Contains characteristic information for the local node.
node.db	Contains ES/Dnet node database.
object.db	Contains the ES/Dnet object database.
proxy.db	User created database which contains the ES/Dnet proxy database. Also contains the default ES/Dnet account if one is specified.

These databases are all located in **/usr/etc/esdnet/database**.

pconfig Commands

Table 4-1 summarizes the **pconfig** commands. You can use the **-help** option during invocation to get a description of the available commands and their usage formats.

Table 4-1. pconfig Menu Commands

Menu Command	Description
Exit with Changes Made	Exits **pconfig** and makes changes.
Abort w/o Making Changes	Aborts **pconfig** without making any changes to the database. You will be prompted whether to abort or not.
Restore Initial Values	Restores initial values.
Local Node Maintenance	Defines the Ethernet interface name and the node number for your local node. When ES/Dnet starts, it defines itself to be this node number. This number must be unique across the network and is usually assigned by the network manager. Also modifies local node parameters.
Node Maintenance	Add, modify, list or remove node names and numbers from the network database. If you want to reference a remote node by its name, you must define it using this command.
ES/Dnet Object Maintenance	Lets you stipulate programs that will be started by the ES/Dnet agent by number. The range is 128 to 255 for non-DEC applications and 0 to 127 for DECnet-known objects.
Make Command File	Generates a command file that contains all the definitions for the node and object databases. This command file is made up of NCP format commands. You can edit and add commands to the command file which is created by this command.

Table 4-1. pconfig Menu Commands (con't.)

Menu Command	Description
Use NCP-Like Command	Provides a command line interface instead of the menu interface. The command format is the same as DEC's DECnet utility Network Control Program (NCP). Commands are limited to configuration commands and the **SHOW EXECUTOR** and **SHOW TIMERS** commands.

pconfig Command Options

pconfig has a number of options. The options and their meanings are listed in Table 4-2.

Table 4-2. pconfig Command Options

Command Option	Description
-commands=	Reads commands from the file **ncp.cmd** created in **pconfig**. When you use this switch, it displays the NCP configuration commands as they are executed.
-commands=pathname	Reads commands from the file specified by pathname .
-help	Displays help text.
-ncp	Runs in command line mode instead of screen mode. Accepts commands that are compatible with DEC's NCP program. Gets you the **NCP>** prompt.
-usage	Displays the command usage.
-version	Displays the version number of **pconfig**.
-?	Displays help text.

pconfig Examples

The following examples are intended to help you use the **pconfig** configuration utility. Each example covers a particular user action, such as updating the list of node names and numbers, or adding an object to the ES/Dnet object database. The example shows the program menu with a menu choice indicated, followed by the system's response and any additional screens or output for the example.

Please keep in mind that you can exit **pconfig** at any time if you do not wish to make any changes to the local nodes' network databases. You do so by entering the number 2, for **Abort Without Making Changes**. It is recommended that you try each example shown here and exit without making changes, before actually modifying your configuration.

Starting pconfig

To start **pconfig**, enter **pconfig** after the system prompt.

```
$ pconfig
                    ES/Dnet Configuration Program
Local Node:  1.9    viper  LAN Device Name: ie0
Choose Required Operation: __

      1.Exit With Changes Made
      2.Abort Without Making Changes
      3.Restore Initial Values
      4.Local Node Maintenance
      5.Node Maintenance
      6.ES/Dnet Object Maintenance
      7.Make Command File
      8.Use NCP-Like Command Line Entry
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
```

Local Node Maintenance

Ethernet interface name, local node number and node parameters can be changed by entering **4** on the main menu for **local node maintenance.**

```
          ES/Dnet Configuration Program
Local Node:  1.2    curly          LAN Device Name: ie0
Choose Required Operation: 5
       1. Exit With Changes Made
       2. Abort Without Making Changes
       3. Restore Initial Values
 ==>   4. Local Node Maintenance
       5. Node Maintenance
       6. ES/Dnet Object Maintenance
       7. Make Command File
       8. Use NCP-Like Command Line Entry
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

```
          ES/Dnet Configuration Program
Local Node:  1.9    viper       LAN Device Name: ie0
Choose Required Operation: 4
  1.   Return to Previous Menu

  2.   Define LAN Device Name
  3.   Define Local Node Number

  4.   Hello Timer       [5   ]
  5.   Inactivity Timer [60  ]  9. Max Retransmits      [10  ]
  6.   Incoming Timer   [120 ] 10. Tryhard Threshhold  [2   ]
  7.   Outgoing Timer   [90  ] 11. Executor Buffer Size[1461]
  8.   Keepalive Timer  [300 ] 12. Max Links           [255 ]
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

To change the Ethernet LAN Unit number, input a menu value of 2. The screen will now show the unit number. To modify this value, input a new unit number followed by RETURN. This value is normally set to "la0".

To change the Local Node Number, input a menu value of 3. The screen will now show the Local Node Number. To modify this value, input a new node number followed by RETURN.

There are up to 63 areas in a DECnet network with a maximum of 1023 nodes in each area. The default number for an undefined node is 63.1023. The number to the left of the decimal point is an area number in the range 1 to 63. If you enter an area number outside of this range, an error message will flash on screen. The number to the right of the decimal point is the node number within the area number. The acceptable range for a node number is 1 to 1023. Entering a node number outside this range results in a flashing error message.

The following Local Node Parameters can be modified from this screen:

- The *Hello Timer* regulates how often an endnode hello timer is sent out.
- The *Inactivity Timer* regulates how often a link service inactivity message is sent on an existing link.
- The *Incoming Timer* regulates how long a node will wait for an incoming connection request to be accepted or rejected before breaking the link.
- The *Outgoing Timer* regulates how long the node will wait for a connection request to be rejected or accepted.
- The *Keep Alive Timer* specifies the amount of time a network object will remain alive waiting for additional connection requests.
- *Max Retransmits* is the number of times that ES/Dnet will send or transport a logical link-level message that has not been acknowledged within a particular node delay.
- *Tryhard Threshold* is the limit of retransmits of unacknowledged packets before a designated router is used to complete the retransmission.
- *Executor Buffer Size* specifies in bytes the size of the receive buffer.
- *Max Links* is the maximum number of logical links that can be established to and from an ESV Workstation.

For more information on these parameters, see the discussions of the **SET NODE** and **EXECUTOR** commands in the VAX NCP command utility manual. Please consult your network manager before changing any of these parameters.

To modify a parameter, input its menu number. The current value will be displayed along with the valid range for the value. An arrow will show the current parameter being changed. Input the new value followed by a RETURN.

Maintaining the Node Database

Node Names may be changed by selecting **Node Maintenance** and the **Modify** option. The **Define Local Node Number** selection modifies only the local node number.

```
          ES/Dnet Configuration Program
Local Node:  1.2    curly          LAN Device Name: ie0
Choose Required Operation: 5
      1. Exit With Changes Made
      2. Abort Without Making Changes
      3. Restore Initial Values
      4. Local Node Maintenance
 ==>  5. Node Maintenance
      6. ES/Dnet Object Maintenance
      7. Make Command File
      8. Use NCP-Like Command Line Entry
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Enter **5** for the **Node Maintenance** choice menu. From this menu you can modify, list, add, and remove node names and numbers from the node database on your node.

```
          ES/Dnet Node Maintenance Choice: __
          1.  Return to Previous Menu
    2.  Add      3.   Modify     4.  List     5.  Remove
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

This is the initial maintenance menu. From this menu you can enter numbers **1** through **5**, depending on the action you want to take.

Adding Nodes

The following example uses the menu choice **2** to add a node number (in this case **1.77**):

```
      ES/Dnet Node Maintenance Choice: 2
      1.  Return to Previous Menu
==> 2.  Add      3.  Modify     4.  List      5.  Remove
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Enter node number: 1.77
```

Using this menu you can add any new node names and numbers that you want your local node to communicate with by node name. If you wish to communicate with node 77 by its name (mouse1) you must first enter it into your node's node database using the **add** option.

After you enter the node number and press RETURN, you will see the following screen:

```
      ES/Dnet Node Maintenance Choice: 2
      1.  Return to Previous Menu
==> 2.  Add      3.  Modify     4.  List      5.  Remove
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
AA-00-04-00-4D-04  1.77            ()
Enter node name for this node:
Enter node identification     :
Enter node physical address   : AA-00-04-00-4D-04
```

The **AA-00-04-00-4D-04** is the DECnet address. It consists of two parts and is calculated and displayed for you by **pconfig**. It is comprised of:

1) A hard-coded **AA-00-04-00**.

2) **4D-04** for node number. The lower 10 bits are the node number within an area. The high-order six bits are the area number. Note that DEC places the least significant byte first, most significant byte second.

If the node already exists, the system will tell you. To complete this selection, enter the name of the node. The node name must be no more than six characters; this is a DECnet restriction. You may also specify the node identification. This could consist of what type of machine the node is, and its location and extension to call for troubleshooting assistance. For example,

```
      ES/Dnet Node Maintenance Choice: 2
      1.  Return to Previous Menu
==> 2.  Add      3.  Modify     4.  List      5.  Remove
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
AA-00-04-00-4D-04  1.77            ()
Enter node name for this node: mouse1
Enter node identification     : Production #1
Enter node physical address   : AA-00-04-00-4D-04
```

(If you wanted to exit at this point instead of adding a node, you could have pressed RETURN and not entered a node name.)

When you complete the node information, which consists of the node name and an ID string, the following menu will appear. Please note that you do not have to modify the physical address; this is calculated for you by **pconfig**.

```
      ES/Dnet Node Maintenance Choice: _
      1.  Return to Previous Menu
==> 2.  Add      3.  Modify     4.  List      5.  Remove
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
AA-00-04-00-4D-04  1.77            ()
Enter node name for this node: mouse1
Enter node identification     : Production #1
Enter node physical address   : AA-00-04-00-4D-04
AA-00-04-00-4D-04  Node 1.77    mouse1   (Production #1) added
```

You can use the node ID string to identify the node by location or by function, or by the node's user. The node ID is limited to 32 characters

Modifying Nodes

You can modify an existing entry in the node database by entering the node number or name. The following menu shows node name **mouse1** being entered.

```
      ES/Dnet Node Maintenance Choice: 3
      1.  Return to Previous Menu
2.  Add  ==>  3.  Modify      4.  List       5.  Remove
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Enter node number/name: mouse1
```

Press RETURN to display the information in the node database, as shown in the following menu.

```
      ES/Dnet Node Maintenance Choice: 3
      1.  Return to Previous Menu
2.  Add  ==> 3.  Modify      4.  List       5.  Remove
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Enter node number: 1.77
Enter node name for this node: mouse1
Enter node identification     : Production #1
Enter node physical address   : AA-00-04-00-4D-04
AA-00-04-00-4D-04  Node 1.77    mouse1   (Production #1)
```

You can modify the current information about the node by entering the new information at the beginning of each field. (If you do not want to modify, enter RETURN at the beginning of each field.)

```
      ES/Dnet Node Maintenance Choice: 3
      1.  Return to Previous Menu
2.  Add    ==> 3.   Modify     4.  List       5.  Remove
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Enter node number: 1.77
Enter node name for this node: mouse1
Enter node identification     : Supply Station
Enter node physical address   : AA-00-04-00-4D-04
AA-00-04-00-4D-04  Node 1.77   mouse1  (Production #1)
```

In the previous menu, **Production #1** has changed to **Supply Station**. When you press RETURN after the physical address, the following menu appears:

```
      ES/Dnet Node Maintenance Choice: __
      1.  Return to Previous Menu
2.  Add    ==> 3.  Modify      4.  List       5.  Remove
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Total entries modified: 1.
```

Listing Nodes

Suppose you want to display all the nodes in the database with a node address between 100 and 199, inclusive. Use the VAX wildcards "%" or "*" to specify these nodes. In the following example, the percent symbol is used:

```
      ES/Dnet Node Maintenance Choice: 4
      1.  Return to Previous Menu
   2.  Add     3.  Modify     ==> 4.  List       5.  Remove
   Ethernet Address     Node     Name    Identification
   =================   ======   ======   ==============
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
List which node(s)?: 1.1%%__
```

Pressing RETURN after the wildcard in the previous example will produce a list of all nodes in area 1 that have 1 as the first digit in a 3-digit node number, which includes all nodes between 100 and 199, inclusive. The default entry is "*" which lists all known nodes. The following two screens show the output resulting from the previous example.

```
      ES/Dnet Node Maintenance Choice: 4
      1.  Return to Previous Menu
2.  Add    3.  Modify   ==> 4.  List      5.  Remove
Ethernet Address     Node     Name    Identification
=================   ======   ======   ==============
0A-00-04-00-0A-04    1.10    ranger   (dwatts@Tek)
0A-00-04-00-0B-04    1.11    vise     (corrigan@Tektoo)
0A-00-04-00-0C-04    1.12    sea      (sea)
0A-00-04-00-0D-04    1.13    bull     (winkle)
0A-00-04-00-0E-04    1.14    truth    (truth)
0A-00-04-00-0F-04    1.15    plym     (plymouth)
0A-00-04-00-10-04    1.17    beauty   (VAXstation)
0A-00-04-00-11-04    1.18    august   (Archer)
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Using template: '1.%%'
Press Return to continue:
```

Pressing RETURN displays the rest of the list. The next screen shows the completed command:

```
      ES/Dnet Node Maintenance Choice: __
      1.  Return to Previous Menu
2.  Add    3.  Modify   ==> 4.  List      5.  Remove
Ethernet Address     Node     Name    Identification
=================   ======   ======   ==============
08-00-1E-00-9B-D7    1.19    cuke     (In Lab)
08-00-1E-01-21-A1    1.20    duke     (Tom Tacky)
08-00-1E-00-BE-99    1.21    charm    (Mike Room)
08-00-1E-01-21-A1    1.23    n_23     (n_23)
AA-00-04-00-18-04    1.24    plow     (Jim Hirni)
08-00-2B-06-65-F3    1.30    maxx     (VAX 750)
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Total entries displayed: 14.
```

The Ethernet address is the physical address specified. The physical **pconfig** address can be the default DECnet address (which is supplied by the program) or the hardware's actual address (which you supply). Physical addresses starting without **AA-00-04-00** usually indicate a hardware controller's actual address. This actual address or hardware address is the address the board is shipped with from the factory. The **AA-00-04-00** type number is a DECnet node address or physical address.

Removing Nodes

You can remove 1.10 from the node list by selecting **5** as the **Node Maintenance Choice**, and then entering the node number **1.10** or, as in the following example, by entering the node's name.

```
      ES/Dnet Node Maintenance Choice: 5
      1.  Return to Previous Menu
   2.  Add      3.  Modify      4.  List      ==> 5.  Remove
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Enter node number/name: mouse1
```

Entering the node name **mouse1** gets you the following screen.

```
      ES/Dnet Node Maintenance Choice: 5
      1.  Return to Previous Menu
2.  Add      3.  Modify      4.  List      ==> 5.  Remove
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Using template: 'mouse1'
Remove "AA-00-04-00-4D-04  1.77   mouse1  (Production #1)": n
```

pconfig gives you a second chance to say no. By default, **n** is present because no is the default. You must enter **y** over the **n** and press RETURN in order to remove the node. If you find that you do not want to remove this node after leaving this screen and you have not exited **pconfig**, exit **pconfig** without making changes.

Maintaining Network Objects

A *network object* is an application that resides somewhere on the network. If you want to add, modify, list, or remove objects, you enter **6** and then press RETURN.

```
      ES/Dnet Configuration Program
   Local Node:  1.2    curly       LAN Device Name: ie0
   Choose Required Operation: 6
   1. Exit With Changes Made
   2. Abort Without Making Changes
   3. Restore Initial Values
   4. Local Node Maintenance
   5. Node Maintenance
==>6. ES/Dnet Object Maintenance
   7. Make Command File
   8. Use NCP-Like Command Line Entry
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

You will then see the following menu:

```
      ES/Dnet Object Maintenance Choice: 2
      1.  Return to Previous Menu
==> 2.  Add      3.  Modify      4.  List      5.  Remove
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Enter object name     :
Enter object number   :
Enter object filename:
```

Adding to the List of Known Objects

If you want ES/Dnet to start a network application for you, enter **2**, and then enter the location of the network application, its object name, and its object number. ES/Dnet will start any object from across the network thus named.

You should give your own objects a number greater than 127 because object numbers 1 through 127 are reserved for DEC and ES/Dnet applications such as **d_server** and **sethost**. You must give the exact pathname to the file to be started on behalf of the remote node's request by ES/Dnet. ES/Dnet will start this file when a request for the object is received. Once started, it will wait for more connections based on the keep alive timer value set by the **pconfig -ncp SET KEEPALIVE TIMER <SECONDS>** command. It will wait for any new request with the same access control. Access control is username and password, or a proxy name or default account.

The following screen shows the menu selection to add an object.

```
      ES/Dnet Object Maintenance Choice: 2
      1.  Return to Previous Menu
==> 2.  Add     3.  Modify      4.  List      5.  Remove
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Enter object name     : howmany
Enter object number   : 231
Enter object filename: /bin/howmany
```

The previous examples define an object called **howee** (a user-written network program that returns the number of CAD drawings completed in one hour). The application will be started when a request for the object number comes in from a remote node's network application. The remote node's application must be designed to accept the number of drawings completed as input from the **howee** network application.

Note: Once the object is updated, ES/Dnet must be updated before the new entry in the object database takes effect. Tó update ES/Dnet, use the **updb** command

Modifying an Object

If you want to change the object number, the object name is entered after selecting **3** for **modify**. The following example changes the object number from 231 (as in the previous example) to 234.

```
      ES/Dnet Object Maintenance Choice: 3
      1.  Return to Previous Menu
2.  Add     ==> 3.  Modify      4.  List      5.  Remove
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Enter object name     : howmany
Enter object number   : 234
Enter object filename: /bin/howmany
howee                 234   /bin/howee
```

The following example changes the pathname. You may modify a field entry by typing the new value. Remember, a field remains unmodified if you enter RETURN with the cursor at the beginning of the field. In the next example, you can see that the object filename has changed from **/bin/howmany** to **/bin/howmany_cad**.

```
      ES/Dnet Object Maintenance Choice:3
      1.  Return to Previous Menu
2.  Add     ==> 3.  Modify      4.  List      5.  Remove
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Enter object name     : howmany
Enter object number   : 234
Enter object filename: /bin/howmany_cad
howee                 234   /bin/howmany_cad
```

Note: Once the object is updated, ES/Dnet must be updated before the new entry in the object database takes effect. To update ES/Dnet, use the **updb** command

Listing Objects

You can list the objects in the object database by entering **4** and then pressing RETURN, as shown in the next example.

```
      ES/Dnet Object Maintenance Choice: 4
      1.  Return to Previous Menu
2.  Add      3.  Modify     ==> 4.  List      5.  Remove
Object Name     Number  File
=============   ======  ======================
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
List which object(s)?:*
```

You can list all known objects by entering RETURN at the default wildcard "*". Again, VAX wildcards are supported. The next screen assumes you entered RETURN after the default.

```
      ES/Dnet Object Maintenance Choice: __
      1.  Return to Previous Menu
   2.  Add      3.  Modify     ==> 4.  List      5.  Remove
  Object Name     Number  File
===============  ======  ======================
obj133              133  /etc/dna/bin/obj133
d_server             17  /etc/dna/bin/d_server
obj129              129  /etc/dna/bin/obj129
obj132              132  /etc/dna/bin/obj132
mirror               25  /etc/dna/bin/mirror
dnamail              27  /etc/dna/bin/dnamail
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Total entries displayed: 6.
```

Removing Objects

To remove an object (such as **howmany_cad**), enter the name **howmany_cad**, enter **h*** (gets all **h** objects), or enter the number **234**. In the following example, the object name **howmany_cad** is used to remove the object.

```
      ES/Dnet Object Maintenance Choice: 5
      1.  Return to Previous Menu
      2.  Add     3.  Modify      4.  List     ==> 5.  Remove
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Enter template for objects to remove: howmany_cad
ES/Dnet Object Maintenance Choice: 5
```

pconfig gives you a second chance. You must override the default **n** (for no) with **y** (for yes) in order to remove this object name from this node's object database:

```
      ES/Dnet Object Maintenance Choice: 5
      1.  Return to Previous Menu
      2.  Add     3.  Modify      4.  List     ==> 5.  Remove
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Using template: 'howmany_cad'
Remove "howmany_cad              234  /bin/howmany_cad": y
```

The next example shows the completed screen.

```
      ES/Dnet Object Maintenance Choice:__
      1.  Return to Previous Menu
   2.  Add     3.  Modify      4.  List     ==> 5.  Remove
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Using template: 'howmany_cad'
REMOVED:        "howmany_cad                234
/bin/howmany_cad"
Operation Complete.  1 entries removed.
```

Making a Command File

If you wish to automate **pconfig** or save your current work, this option can create a command file that will run without the menu interface. First, select **7**

```
      ES/Dnet Configuration Program
Local Node:  1.2    viper        LAN Unit Number: 1
Choose Required Operation: 7
      1. Exit With Changes Made
      2. Abort Without Making Changes
      3. Restore Initial Values
      4. Local Node Maintenance
      5. Node Maintenance
      6. ES/Dnet Object Maintenance
==>   7. Make Command File
      8. Use NCP-Like Command Line Entry
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Enter filename to create: ncp.cmd
```

This option allows you to create a command file of the NCP configuration commands necessary to define nodes, objects, and other network information. (The device code command is included. It has no VAX counterpart and is not relevant to ES/Dnet.).

You can now exit **pconfig**. If you enter **ncp.cmd**, you will see the commands necessary to define your existing ES/Dnet databases. You can edit, and then execute the **ncp.cmd** file, to add and remove nodes. The output would display NCP commands as they executed. For example

```
$ pconfig -commands=ncp.cmd
NCP> !
NCP> ! File Generated By ES/Dnet: Version 0.0 On Thu
Jul      28 14
NCP> !
NCP>
NCP> !
NCP> ! First we should let ES/Dnet know what our local
address is:
NCP> Define Executor Address 1.10
NCP>
NCP> !
NCP> !Need to know the LAN Unit Number
NCP> Set LAN Unit Number 0
NCP>
NCP> !
NCP> !And now to define all the nodes:
NCP> !
NCP> Set Node 1.1 Hardware Address AA-00-04-00-01-04
```

```
NCP> Set Node 1.1 Name vaxapo Identification Really is MAXX
NCP>
NCP> Set Node 1.2 Hardware Address AA-00-04-00-02-04
NCP> Set Node 1.2 Name curly Identification Multimax
NCP>
NCP> Set Node 1.8 Hardware Address AA-00-04-00-08-04
NCP> Set Node 1.8 Name encore Identification Micro Vax
NCP>
NCP> Set Node 1.9 Hardware Address 08-00-1E-01-20-D9
NCP> Set Node 1.9 Name viper Identification dwatts @ home
NCP>
NCP> Set Node 1.10 Hardware Address 08-00-1E-01-1D-47
NCP> Set Node 1.10 Name ranger Identification dwatts @ Tek
NCP>
NCP> Set Node 1.11 Hardware Address 08-00-1E-01-2A-BD
NCP> Set Node 1.11 Name vise Identification corrigan@Tek
NCP>
NCP> Set Node 1.12 Hardware Address 08-00-1E-01-22-2B
NCP> Set Node 1.12 Name sea Identification sea
NCP>
NCP> Set Node 1.13 Hardware Address 08-00-1E-00-B5-BD
NCP> Set Node 1.13 Name bull Identification //bullwinkle
NCP>
NCP> Set Node 1.14 Hardware Address 08-00-1E-01-1A-95
NCP> Set Node 1.14 Name truth Identification J Snow
NCP>
NCP> Set Node 1.15 Hardware Address 08-00-1E-00-D2-1A
NCP> Set Node 1.15 Name plym Identification //plymouth
NCP>
NCP> Set Node 1.17 Hardware Address 08-00-2B-06-65-F3
NCP> Set Node 1.17 Name beauty Identification VAXstation
NCP>
NCP> Set Node 1.18 Hardware Address 08-00-1E-01-31-72
NCP> Set Node 1.18 Name august Identification Archer
NCP>
NCP> Set Node 1.29 Hardware Address 08-00-1E-01-29-EA
NCP> Set Node 1.29 Name sjm Identification Scott M
NCP>
NCP> Set Node 1.30 Hardware Address 08-00-2B-06-65-F3
NCP> Set Node 1.30 Name maxx Identification VAX 750
NCP>
NCP> Exit
NCP Execution Complete
```

Using NCP-like Command Line Entry

Selecting **8** will initiate NCP, just as it is on the VAX. The following examples illustrate this.

Show Executor

```
        ES/Dnet Configuration Program
   Local Node:  1.2    viper        LAN Unit Number: 1
Choose Required Operation: 8
        1. Exit With Changes Made
        2. Abort Without Making Changes
        3. Restore Initial Values
        4. Local Node Maintenance
        5. Node Maintenance
        6. ES/Dnet Object Maintenance
        7. Make Command File
==>     8. Use NCP-Like Command Line Entry
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
NCP> show executor
```

The following screen shows the output from the previous **show executor** command.

```
Executor Node:
        DECnet Node Number   1.2
        DECnet Node Name     viper
        Identification       dwatts @ home
        Ethernet Address     08-00-1E-01-20-D9
Lan Device Information:
Unit Number: 0
Timers:
        Hello Timer           15 seconds
        Inactivity Timer      60 seconds  ( 1 minute )
        Incoming Timer       120 seconds  ( 2 minutes )
        Outgoing Timer        90 seconds  ( 1 minute 30 seconds )
        KeepAlive Timer       30 seconds
Max Retransmits:          10
Tryhard Threshold:        2
Executor Buffer Size:     1461 bytes
NCP>
```

- *Max Retransmits* is the number of times that ES/Dnet will send or transport a logical link-level message that has not been acknowledged within a particular node delay.
- *Tryhard Threshold* is the limit of retransmits of unacknowledged packets before a designated router is used to complete the retransmission.

For more information about timers, see the discussions of the **SET NODE** and **EXECUTOR** commands in the VAX NCP command utility manual .

Show Timers

You can display the current value for timers in the database with the **show timers** command. All times are specified in seconds

- The *Hello Timer* regulates how often an endnode hello timer is sent out.
- The *Inactivity Timer* regulates how often a link service inactivity message is sent on an existing link.
- The *Incoming Timer* regulates how long a node will wait for an incoming connection request to be accepted or rejected before breaking the link.
- The *Outgoing Timer* regulates how long the node will wait for a connection request to be rejected or accepted.
- The *Keep Alive Timer* specifies the amount of time a network object will remain alive waiting for additional connection requests.

```
NCP> show timers
Timers:
      Hello Timer            15 seconds
      Inactivity Timer       60 seconds (1 minute)
      Incoming Timer        120 seconds (2 minutes)
      Outgoing Timer         90 seconds (1 minute 30 seconds)
      KeepAlive Timer        30 seconds
```

Show Known Nodes

You can display the local/executor node and a brief list of all other nodes by entering the **show known nodes** command

```
NCP> show known nodes
Executor Node:
      DECnet Node Number    1.2
      DECnet Node Name      viper
      Identification        dwatts @ home
      Ethernet Address      08-00-1E-01-20-D9
20 nodes in use out of maximum of 300
Ethernet Addr     Node Number        Name (Identification)
=================  ===========        ====================
AA-00-04-00-02-04      1.2             curly         (Multimax)
AA-00-04-00-08-04      1.8             encore        (Micro Vax)
08-00-1E-01-20-D9      1.9             viper         (dwatts)
08-00-1E-01-1D-47      1.10            ranger        (dwatts)
08-00-1E-01-2A-BD      1.11            vise          (Tek)
08-00-1E-01-22-2B      1.12            sea           (sea)
08-00-1E-00-B5-BD      1.13            bull          (bull)
```

```
08-00-1E-01-1A-95     1.14          truth        (truth)
08-00-1E-00-D2-1A     1.15          plym         (plymouth)
08-00-2B-06-65-F3     1.17          beauty
(VAXstation)
08-00-1E-01-31-72     1.18          august       (Andy)
AA-00-04-00-18-04     1.24          plow         (Jim Hirni)
08-00-2B-06-65-F3     1.30          maxx         (VAX 750)
AA-00-04-00-4D-04     1.77          mouse1       (Prod#1)
08-00-1B-01-01-5C     1.100         beast        (MV/4000)
08-00-1B-04-07-48     1.101         buster       (MV/2000)
NCP>
```

Show Known Objects

You can show all known objects by entering the **show known objects** command.

```
NCP> show known objects
Object Name       Number  File
===============   ======  ======================
obj133               133  /etc/dna/obj133
fal                   17  /etc/dna/d_server
obj129               129  /etc/dna/obj129
obj132               132  /etc/dna/obj132
mirror                25  /etc/dna/mirror
5 objects displayed
```

Purging an Object

To purge an object from the object database, enter **purge object <node number>**. For example,

```
NCP> purge object 132
REMOVED obj132                132  /etc/dna/obj132
The object has been removed
NCP>
```

Listing NCP Commands

You can get a listing of supported NCP commands by entering **help** at the **NCP>** prompt.

Exiting NCP

You can leave NCP by entering **quit** at the **NCP>** prompt. If you have made changes, **pconfig** will ask whether you want to abort and not save changes. If you enter **n** (for no) it will return you to the **NCP>** prompt.

Then, if you enter **exit** it will automatically update any changes to the database.

```
NCP> quit
No changes made to database. Exiting normally
```

In this case you see an example of exiting NCP when no changes have been made. If you do exit with changes, you will see:

```
Do you really want to abort and not save changes? n
NCP> exit
Updating database with changes
NCP Execution Complete
$
```

Exiting pconfig

There are two ways to exit **pconfig**:

- **exit** with changes made
- **abort** without making changes.

exit

Enter **1** to exit with changes completed. The default response is yes. If you enter **no**, you are returned to the initial screen.

```
      ES/Dnet Configuration Program
Local Node:  1.2    viper         LAN Device Name: ie0
Choose Required Operation: 1
 ==>  1. Exit With Changes Made
      2. Abort Without Making Changes
      3. Define LAN Unit Number
      4. Define Local Node Number
      5. Node Maintenance
      6. ES/Dnet Object Maintenance
      7. Make Command File
      8. Use NCP-Like Command Line Entry
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Do you want to exit with changes made:  Yes__
```

abort

If you made no changes or you made unwanted changes, enter **2**. The default answer is no. You must enter **yes** in order to exit; otherwise, you are returned to the initial screen.

```
      ES/Dnet Configuration Program
Local Node:  1.2    viper        LAN Device Name: ie0
Choose Required Operation: 2
      1. Exit With Changes Made
==>   2. Abort Without Making Changes
      3. Restore Initial Values
      4. Local Node Maintenance
      5. Node Maintenance
      6. ES/Dnet Object Maintenance
      7. Make Command File
      8. Use NCP-Like Command Line Entry
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Do you really want to abort without changes: No__
```

Using the Network Command Facilities

The following discussions provide examples using the Network Command Utilities **shownet**, **showcir**, **showproxy**, **showobj**, **dna_stop**, **updb**, **mirror** and **loop**. Each command's purpose is described, its options are listed, and examples are given of the command and the command's output.

dna_stop

Syntax:

dna_stop

Description:

Use the **dna_stop** command to terminate the ES/Dnet agent. Any active DECnet links will be closed.

Enter **dna_stop** from the command line to stop ES/Dnet. ES/Dnet will stop and **dna_stop** will return a message indicating that ES/Dnet has stopped.

Example:

```
$ dna_stop
Execution complete  DNA_STOP
```

Related VAX Commands:

VAX DECnet may be halted with a command procedure using NCP commands. See the *NCP Utility Manual* for **SET EXECUTOR STATE OFF/SHUT**.

loop

Syntax:

loop [-max=#] [-pattern=##] [-repeat=#]

Description:

loop tests DECnet connectivity up to and including the application layer. Using **loop** you can verify that two programs on different nodes can communicate across DECnet, and that data does not get corrupted. **loop** requests a connection to the mirror object on either ES/Dnet or the DECnet.

loop verifies that the local and remote network software is functioning properly. **loop** tests all seven layers of the network software.

The output from **loop** includes the size in bytes of the available maximum ES/Dnet buffer, the requested connection's (mirror) maximum buffer, **loop**'s maximum buffer, the maximum buffer actually used for this test, the test pattern (which is, by default, a series of 55 alternating 1s and 0s), and the number of times the test will be repeated. The **loop** test is capable of sending messages starting at 256 bytes and going up to the maximum of 4096 bytes.

Options:

-max=#	The maximum length message to send.
-pattern=##	The hexadecimal number to use as the test pattern.
-repeat=#	How many times to repeat each message length.

Examples:

Using loop

In the following example, a maximum size is set at 2048 bytes. (If you want to loop only once, set **-max = 256**.)

```
$ loop -max=2048 viper::
LOOP -- DECnet Loop Test Program
    DECnet Max I/O Size:     4096 bytes
    Mirror Buffer Size:      4096 bytes
    Loop Max Buffer Size:    2048 bytes
    Using Max Buffer size:   2048 bytes
    Test Pattern:            0x55
    Repeat times:            1 per buffer size
```

```
testing  256 bytes...OK!
testing  512 bytes...OK!
testing  768 bytes...OK!
testing 1024 bytes...OK!
testing 1280 bytes...OK!
testing 1536 bytes...OK!
testing 1792 bytes...OK!
testing 2048 bytes...OK!
Loop Test Successful
```

Showing a loop Failure

If the test fails to the remote node, run it to yourself to verify the local node is functioning properly. The following example illustrates a possible failure.

```
$ loop -max=2048 viper::
  LOOP -- DECnet Loop Test Program
    DECnet Max I/O Size:     4096 bytes
    Mirror Buffer Size:      4096 bytes
    Loop Max Buffer Size:    2048 bytes
    Using Max Buffer size:   2048 bytes
    Test Pattern:            0x55
    Repeat times:            1 per buffer size
    testing  256 bytes   only x bytes written
    *** Loop Test Failed ***
```

Performing Similar Operations with dap

Another way to test whether you have connectivity to a remote system or whether you have a properly functioning local node is to perform a **dap** operation such as **directory**. For example,

```
$ dap dir <remote node name> :: *.* (to test remote system
connectivity)
```

Related VAX Commands

In order to loop a message to a node, you can invoke **NCP LOOP NODE <*REMOTE NODE NAME*>**. See the *VAX Network Command Program Utilities Manual* for more details. You can also enter **DIR 0::** to test local node functionality or **DIR <*REMOTE NODE NAME*>::*.*** to test remote node connectivity.

mirror

Syntax:

mirror

Description:

mirror is a DECnet Loopback Object. The ES/Dnet loop program connects to this object as well as to the VMS NCP **LOOP NODE** command. To loop from the VAX, use the NCP **LOOP NODE *<NODE NAME>*** command. The NCP **LOOP NODE** command connects to this mirror object. Usually, you don't bother running this object because ES/Dnet creates a network process to run it automatically when the VAX DECnet's or ES/Dnet's loop program connection request is received.

Because you usually do not need to run this yourself, the output from mirror can be found in the **net_*<link_id>*.log** file. If you suspect that mirror is not being started properly by ES/Dnet, start it from the command line or consult the **net*<link_id>*.log** in the ES/Dnet directory for error messages.

Example:

```
$ mirror
MIRROR - DECnet Loopback Mirror.  Object Number 25
      Waiting for connection
      *** Connection received ***
      Agent_id= 0x0001
      Max size= 4096
      Access control:
            Remote Node:    'viper'
            Remote Process: 'dwatts'
            Username:       ''
            Password:       ''
            Account:        ''
Connect Data: 0 bytes
0000:
MIRROR calling NREAD
Error o150050 (0x0000D028, 53288.)
I/O Terminated By Close Received 34,816 bytes in 5.617
seconds (6,198 bytes/sec, 49,584 bits/sec)
Received 16 messages (2.849 msgs/sec)
```

Related VAX Commands

There is no related VAX command. VAX DECnet only starts **mirror** for the incoming **loop** request.

showcir

Syntax:

showcir [-active] [-zero]

Description:

showcir displays the Circuit/Line Counters for the local ES/Dnet node. These counters can optionally be zeroed after they have been read using the **-zero** option.

The network statistics returned are the same as those returned from the DNA command **show circuit** counters. See Table 4-3 for the Circuit Counter Field Description Table.

Options:

-active	Include only circuits which are active (default).
-zero	Zero counters after reading.

Examples:

The following example uses **showcir** to display network statistical information.

```
$ showcir
DECnet circuit counters for local node 1.9 viper on Fri Sep
2
11:02:01 1988
Up since: Fri Sep  2 10:56:37 1988
Up time:  5 minutes 24 seconds
Circuit Counters
      data_bytes_rcvd       0, data_bytes_sent:       0
      mc_bytes_rcvd:       98, mc_bytes_sent:        97
      bytes_rcvd:          98, bytes_sent:           97
      dat_frames_rcv:       0, dat_frames_sent:       0
      mc_frames_rcvd:       1, mc_frames_sent:        1
      frames_rcvd:          1, frames_sent:           1
      send_1_collision:     0, send_multi_coll:       0
      send_failure:         0, collision_failure:     0,
      no_carrier:           0
      loss_of_cts:          0, dma_underrun:          0
      too_many_retries:     0, no_sys_buffer:         0,
      no_user_buffer    0
Total of 1 circuits.
$
```

To zero the circuit counters, enter **showcir** with the option **-zero**.

```
$ showcir -zero
DECnet circuit counters for local node 1.9 viper on Fri Sep
2 11:01:54
1988
Up since: Fri Sep  2 10:56:37 1988
Up time:  5 minutes 17 seconds
Circuit Counters
      data_bytes_rcvd:          0, data_bytes_sent:          0
      mc_bytes_rcvd:         1960, mc_bytes_sent:         1940
      bytes_rcvd:            1960, bytes_sent:            1940
      dat_frames_rcv:           0, dat_frames_sent:          0
      mc_frames_rcvd:          20, mc_frames_sent:          20
      frames_rcvd:             20, frames_sent:             20
      send_1_collision:         0, send_multi_coll:          0
      send_failure:             0, collision_failure:        0,
      no_carrier:               0
      loss_of_cts:              0, dma_underrun:             0
      too_many_retries:         0, no_sys_buffer:            0,
      no_user_buffer            0
Total of 1 circuits.
```

Understanding Multicast Messages

An Ethernet multicast message (or **MC_**) is a message sent to a group of nodes. This group of nodes must declare the fact that they are willing to receive a list of multicast addresses when the ES/Dnet software comes up.

The multicast messages in the ES/Dnet are:

1) *Router Hello messages*, which tell other routers about each other and who they know to be routers via a list in the router hello message.
2) *Router Updates*, which detail path length and path cost information contained in a router's forwarding and routing databases.
3) *Endnode Hello messages*, which are sent to an endnode to tell routers they are available on the network to receive data.
4) *Loop messages* that are sent by DEC diagnostic and NCP **loop circuit** commands

These messages are regulated by timers which can be set by the user. ES/Dnet transmits two multicast messages: Endnode Hello and the MOP System ID message. ES/Dnet honors information gathered by the VAX/VMS Ethernet product or DEC diagnostics such as Network Interconnect Exerciser or NCP's show/set module configuration

commands. To see how often the **endnode_hello** message is sent, enter **show timers** from the **NCP>** prompt in **pconfig**, or enter **shownet -full**, which displays the hello timer value expressed in seconds.

With smart cache on, ES/Dnet receives two multicast messages, Endnode Hello and Router Hello. With smart cache off, ES/Dnet receives only one multicast message, the Router Hello. As an endnode, ES/Dnet does not need the information in the routing update message.

Related VAX Commands

Related VAX commands include the NCP **SHOW CIRCUIT** and **LINE COUNTER** commands.

Table 4-3. Circuit Counter Field Description

Field	Description
data_bytes_rcvd	The total number of data bytes successfully received. This does not include Ethernet data link headers. This number is the number of bytes in the Ethernet data field, which includes any padding or length fields when they are established.
bytes_sent	The total number of data bytes transmitted.
mc_bytes_rcvd	The total number of multicast data bytes successfully received. This does not include Ethernet data link headers. This number is the number of bytes in the Ethernet data field.
mc_bytes_sent	The total number of multicast data bytes successfully sent. This does not include Ethernet data link headers. This number is the number of bytes in the Ethernet data field. This consists of endnode hello messages.
bytes_rcvd	The total number of user data bytes successfully received. This does not include Ethernet data link headers. This number is the number of bytes in the Ethernet data field, which includes any padding or length fields when they are established. Both multicast and single cast.

Table 4-3. Circuit Counter Field Description (con't.)

Field	Description
bytes_sent	The total number of user data bytes successfully transmitted. This does not include Ethernet datalink headers or datalink generated retransmissions. This number is the number of bytes in the Ethernet data field, which includes any padding or length fields when they are enabled. Both multicast and single cast.
dat_frames_rcv	Of the total, the number of frames that were single-casted.
dat_frames_sent	Of the total, the number of frames that were single-casted.
mc_frames_rcvd	The total number of multicast frames successfully received.
mc_frames_sent	The total number of multicast frames successfully sent. This consists of endnode hello messages.
frames_sent	The total number of frames successfully transmitted. This does not include data link generated retransmissions.
frames_rcvd	The total number of frames successfully received. These are frames that passed hardware filtering.
send_1_collision	The total number of times that a frame was successfully transmitted on the second attempt after a normal collision on the first attempt.
send_multi_coll	The total number of times that a frame was successfully transmitted on the third or later attempt after normal collisions on the previous attempts.
send_failure	The total number of times a transmit attempt failed.
collision_failure	The approximate number of times that collision detect was not sensed after a transmission.
no_carrier	The number of times that a transmit failed due to No Carrier Sense signal during transmission.

Table 4-3. Circuit Counter Field Description (con't.)

Field	Description
loss_of_cts	The number of times that a transmit failed due to loss of the Clear-to-Send signal.
dna_underrun	The number of times that a transmit failed due to a DMA underrun, (*i.e.*, data not supplied from the system bus for transmission quickly enough).
too_many_retries	The number of times that a transmit failed due to an excessive number of collisions (16).
no_sys_buffer	The number of packets that were dropped because there was no ES/Dnet buffer in which to place it.
no_user_buffer	The number of packets that were dropped because there was no user buffer in which to place it. This will always be 0 for ES/Dnet.

showlink [link]

Syntax:

showlink [-active] [-disconnect] [-full] [-known]

Description:

Use the **showlink** command to display link information.

Options:

-active	List only Active items
-disconnect	Disconnect Specified Link(s)
-full	Generate a full information display
-known	List all known items (default)

Example:

```
$ showlink
DECnet status for local node 1.102 ki on Wed Mar  8 15:14:05
1989
Designated Router: *** None ***
Up since: Wed Mar  8 15:13:43 1989
Up time:  22 seconds
Link 7B00 => 0000, LOBJ 129, PID 337, uinfo: 0000, State
OPEN Usr opn
Total of 1 link.


$ showlink -active
DECnet status for local node 1.102 ki on Wed Mar  8 15:14:09
1989
Designated Router: *** None ***
Up since: Wed Mar  8 15:13:43 1989
Up time:  26 seconds
No links.


$ showlink -full
DECnet status for local node 1.102 ki on Wed Mar  8 15:14:14
1989
Designated Router: *** None ***
Up since: Wed Mar  8 15:13:43 1989
Up time:  31 seconds
```

```
Link 7B00 => 0000, LOBJ 129, PID 337, uinfo: 0000, State
OPEN Usr opn
ack_xmt_oth:     0, numoth:      1
ack_xmt_dat:     0, rcv_low:     1, rcv_high: 0,
rcv_pothole:     1 ack_rcv_dat:     0, numlow:      1,
numhigh:      0 numdat: 1, numsent:     0
flowrem_sw:  ON, flowrem_dat:      0, flowrem_int: 1
flowloc_sw:  ON, flowloc_state:    0 timerdat:      0, oth:
0, ack:    0, inact:    0 timercon: 0, out:    0, in:     0,
alive:    0
Total of 1 link.
```

Table 4-4. Link State Table

State	Description
(O) OPEN	The local Session Control has issued an **OPEN** call which created the port.
(CC) CONNECT-CONFIRM	The local Session Control has issued an **ACCEPT** call.
(CI) CONNECT-INITIATE	The local Session Control has issued a **CONNECT-DMT** call.
(RUN) RUNNING	NSP has either received a Connect Confirm message while in the **CONNECT-INITIATE** or **CONNECT-DELIVERED** state or received a Data, Data Request, Interrupt Request, Data Acknowledgement or Other Data Acknowledgement message while in the **CONNECT-CONFIRM** state.
(DI) DISCONNECT-INITIATE	The local Session Control has issued a **DISCONNECT-XMT** call or an **ABORT-XMT** call.
(DIC) DISCONNECT-COMPLETE	NSP has received either a Disconnect Complete message or a Disconnect Initiate message.
(DN) DISCONNECT-NOTIFICATION	NSP has received a Disconnect Initiate message while in the **RUNNING** state.
(CL) CLOSED	The local Session Control has issued a **CLOSE** call.

shownet [node]

Syntax:

shownet [-active] [-full] [-known] [-ncp] [-system] [-zero]

Description:

shownet displays a list of nodes or configuration information. By default, the list contains only those nodes that are currently reachable on the local LAN. The information includes the node type, node number, node name, hops, and next hop to node.

Options:

-active	List only active items (default).
-full	Generate a FULL information display. This includes hardware and current station address and system type if the node is an ES/Dnet node, and a list of the counters associated with that node.
-known	Include all nodes, even if not currently reachable.
-ncp	Generate NCP commands from the list of nodes.
-system	Display node system type, if an ES/Dnet node.
-zero	Zero counters after reading.

Examples:

The following examples illustrate the **shownet** command.

```
$ shownet
DECnet Network status for local node 1.9 viper on Fri Sep  2
11:04:29 1988
Designated Router: *** None ***
Up since: Fri Sep  2 10:56:37 1988
Up time:  7 minutes 52 seconds

Type      Node          Hops  Next Hop to Node
Endnode   1.9    viper    1    (Local)  ->  1.9    viper
Endnode   1.100  buster   1    (Local)  ->  1.100  buster
 Total of 2 nodes.
$
```

If there is a designated router known to the endnode, it will be displayed. The time the local/executor node has been up for will be displayed, as well as the date and time the ES/Dnet software became active. The type, node, hops, and next hop to node information are displayed as well.

To display the system type of the nodes which are using ES/Dnet implementations, enter **shownet** with the option **-system**. The fields that are generated are described in Table 4-5.

Table 4-5. shownet Field Description Table

Field	Description
node type	Endnode, Router, Area Router, Unknown.
node number	The assigned node number in the form area (1-63) node number (1-1023).
node name	The node name associated with the node number.
hops	The logical distance between two adjacent nodes in a network.
next hop to node	Consists of string (Local) ->**##.####** **xxxxxxx**, where **##.####** is the node number of that node and **xxxxxxx** is the name of that node.

The following example uses **shownet** with the **-system** option:

```
$ shownet -system
DECnet Network status for local node 1.9 viper on Fri Sep  2
11:06:00 1988
Designated Router: *** None ***
Up since: Fri Sep  2 10:56:37 1988
Up time:  9 minutes 23 seconds
Type          Node          Hops              Next Hop to Node
Endnode       .9  viper     1  (Local)  ->    1.9    viper

     System:           UNIX
     Total of 1 nodes.
$
```

Use the **shownet** command with the **-full** option to gather configuration information and network statistics. In this example, the **shownet** program refines the following information on a specific node called **viper**.

```
$ shownet -full viper
DECnet Network status for local node 1.9 viper on Fri Sep  2
11:06:50 1988
Designated Router: *** None ***
Up since: Fri Sep  2 10:56:37 1988d
Up time:  10 minutes 13 seconds
Type        Node           Hops                Next Hop to Node
Endnode     1.9  viper     1  (Local)   ->      1.9    viper
System:             UNIX
Station  Address: AA-00-04-00-09-04
Hardware Address: 08-00-1E-01-20-D9
Hello Timer:      15 seconds
On-Cache:         TRUE
                    Received       Transmitted
                    ============   ============
Data Bytes:          0              0
Messages:            0              0
Connects:            0              0
Connects Rejected:   0
Timeouts:            0
Local Flow Offs:     0
Remote Flow Offs:    0
        Total of 1 nodes.
$
```

The statistical information returned includes:

data bytes	Total number of bytes transferred and received.
messages	Total number of messages transmitted and received.
connects (receive)	Requests received from the node for creating a virtual circuit/logical link. Retransmitted connect requests will be included in this total.
connects (transmit)	Requests sent to the node for creating a virtual circuit/logical link. Retransmitted connect requests will be included in this total.
connect rejects	Number of rejections sent to the node rejecting a request to establish a connection i.e., logical link/virtual circuit.
timeouts	The number of times that a message had to be retransmitted to the node

local flow offs	Number of FLOW OFF requests that the local node has sent to the node to stop it from sending further data.
remote flow offs	Number of FLOW OFF requests that the remote node has sent to the local node requesting that no further data be sent.

The station's address is usually the ES/Dnet physical address. The hardware address is the address on the board as it is shipped. The Hello Timer is a timer stated in seconds which regulates how often an endnode hello message is sent out. On-cache true means the node is up and active on the local LAN. You can also use **shownet -full** to display all active nodes.

Related VAX Commands

From DCL on the VAX you may enter **SHOW NET**. The information returned by this DCL command varies depending on the node type. See the VAX/VMS DCL manual for details.

showobj [object]

Syntax:

showobj [-ncp]

Description:

Use the **showobj** command to display the list of known local objects (network applications programs).

Options:

-ncp	Display information in NCP-compatible format.

Example:

```
$ showobj
DECnet status for local node 1.102 viper on Fri Dec 9
12:31:34  1988
Designated Router:  *** None ***
Up since:  Fri Dec  9  09:14:29  1988
Up time:   3 hours 17 minutes 5 seconds
Object Name         Number      File
===========         ======      =============
d_server            17          /etc/dna/d_server
mirror              25          /etc/dna/mirror
dnamaild            27          /etc/dna/dnamaild
Total of 3 objects
$
```

Related VAX Commands

The NCP command **SHOW OBJECT** is a related VAX command

showproxy [proxy]

Syntax:

showproxy

Description:

Use the **showproxy** command to display the proxy database. The proxy database is in **/usr/etc/esdnet/database**.

Example:

```
$ showproxy
DECnet status for local node 1.102 viper on Fri Dec  9
12:31:37 1988
Designated Router:  *** None ***
Up since:  Fri Dec  9 09:14:29  1988
Up time: 3 hours  17 minutes  8 seconds
viper::atuser   atuser
encore::atuser atuser
curly::atuser   atuser
ranger::atuser   atuser
vise::atuser   atuser
beauty::atuser   atuser
buster::atuser   atuser
beast::atuser   atuser
beast::op atuser
ranger::ki   atuser
vise::ki   atuser
beauty::ki   atuser
buster::ki atuser
ranger::sally   atuser
vise::sally   atuser
beauty::sally   atuser
$
```

updb

Syntax:

updb

Description:

updb updates the active databases with information from the disk databases.

The active network databases updated from the databases on disk include:

- Executor
- Proxy
- Node

Note that the active object database is not updated. In order to update the active object database, you must halt ES/Dnet and restart it. In restarting ES/Dnet, your newly activated node's object database gets its object information from the local node's object database, which is on disk. The local node's object database is updated using the **pconfig** or **pconfig -ncp** commands.

If you have made modifications to the local node's disk-based network databases that you want the running/active ES/Dnet to utilize, enter **updb** from the command line.

Example:

The following example uses the **updb** command to return information about database updates.

```
$ updb
There are 19 nodes defined and 5 objects.
Setting Executor Database...
Setting Node Database...
Clearing Proxy Database...
Setting Proxy Database...
Execution complete
```

Using the **scope** Facility

Use the **scope** facility if you encounter serious network problems. **Scope** stores every packet received or transmitted, and **depict** allows you to view the output.

The following example uses **scope** in the background to copy all transmitted and received data packets to the file **etc/dna/scopefiles/scope.trc**.

```
$ send2dna scopefile /usr/etc/esdnet/scopefiles/scope.trc
```

To halt **scope**,type

```
$ send2dna
```

```
$ send2dna > scopefile
```

then enter RETURN. This closes the file.

You can run **scopex** on the file to get minimal breakdown of protocol trace.

5. Command Summary

This chapter describes each ES/Dnet user command.

Table 5-1. ES/Dnet Command Summary

Command	Description
cd	Changes the current working directory on the local node.
copy	Copies a remote file to a local file or vice versa.
dap	Data access protocol utility.
default	Sets the default access control for the remote node.
delete	Deletes a file at the remote node.
directory	Lists the contents of a remote directory.
dna_stop	Causes the ES/Dnet agent to terminate.
dnamail	Sends a mail message in VMS format to a VMS or an ES/Dnet implementation.
exit	Exits the **dap** program.
get	Copies a file from the remote node.
help	Displays help text for individual commands.
loop	Tests DECnet connectivity.
mirror	DECnet Loopback Object.
pconfig	Utility that provides for DECnet Network Management.
print	Prints a file at the remote node.
purge	Purge remote files
pwd	Lists the current working directory on the local node.
rename	Renames a file at the remote node.
send	Copies a file to a remote node.
sethost	Creates a virtual terminal connection from an ES/Dnet node to a remote DECnet node.
shell	Lets you enter shell commands.
show	Displays current default access control for the remote node as defined by **default** or the system variable DAP _DEFAULT.
showcir	Displays the Circuit/Line Counters for the local ES/Dnet node.

Table 5-1. ES/Dnet Command Summary (con't.)

Command	Description
shownet	Displays a list of nodes or configuration information and their characteristics.
showobj	Displays network object information.
showlink	Displays link information.
showproxy	Displays the **/usr/etc/esdnet/database/proxy.db** database.
submit	Submits a batch file for remote execution.
updb	Used to update the active databases.
usage	Displays usage text for individual commands.

All of the commands listed in Table 5-1, with the exception of **dnamail** and **dnamaild**, include the following options:

Table 5-2. ES/Dnet Command Options

Option	Description
-copyright	Display copyright notice.
-help	Display help information for individual commands.
-usage	Display usage information.
-version	Display software version.

cd

Syntax:

cd [-log] ***directory name***

Description:

cd changes the local current working directory to a new working directory specified by the user. Any local directory pathname may be used.

Options:

-log	Display the name of the new working directory.

copy

Syntax:

copy [-append] [-ascii] [-bin] [-crc] [-hash] [-log] [-nblock] [-noattributes] [-nocrc] [-replace] [-statistics] [-transmit_buffer] ***remote source-file local destination-file***

copy [-ascii] [-binary] [-cr] [-crc] [fix=#] [-hash] [-log] [-nblock] [-nocrc] [-print] [-print -delete] [-statistics] [-transmit_buffer] ***local source-file remote destination-file***

Description:

copy copies files between the ESV Workstation and a remote VAX DECnet node.

The first version is a retrieval (remote to local) with ***source-file*** coming from the remote DECnet node and ***destination-file*** going to the local node.

The second version is a store (local to remote) with ***source-file*** coming from the local node and ***destination-file*** going to the remote DECnet node.

Options:

-append	Appends source file to destination file on retrieval from remote system. The destination file must exist.
-ascii	File is created on VAX with **stream_lf** record format and RETURN carriage control record attributes.
-binary	File is created on VAX with fixed length record format of 512 bytes. When it is a remote network, no conversion is done on file.
-cr	File is created on VAX with a **stream_cr** record format and RETURN carriage control record attributes.
-crc	**dap** CRC is applied and calculated (default).
-fix=#	Specify the record size of the fixed length file on the destination.
-hash	Display "#" for every 150 KBytes transferred.
-log	Displays remote and local filenames verifying user command was completed.
-nblock	Sets the size of **dap** reads and writes to the network.

-noattributes VAX file record attributes are ignored in creating the file on local system during copy retrieve or get command.

-nocrc **dap** CRC is not applied. Recommended when transferring source code files.

-print Copies file and then prints it to default remote system's printer.

-print -delete Copies file to remote system, prints it to default printer and then deletes it once print operation has completed.

-replace Deletes existing local file, before retrieving a file from the remote system.

-statistics Displays file transfer statistics such as bytes per second, read and write operations.

-transmit_buffer
Specifies the size of the **dap** transmit buffer. By default this is 1024.

Examples:

Remote to Local from Command Mode

- This is an example of proxy access.

```
dap copy remote_node::rfilename  lfilename
```

- The following is an example using explicit access.

```
dap copy beast"username password"::[tek]test.c test.c
```

- The following is an example of default access.

```
dap copy  13"":user$1:[tek]test.c test.c
```

- In this example, all files beginning with the letters **te** and any file extension are created locally. Proxy access is used.

```
dap copy beast::te*.* *.*
```

- In this example, all files beginning with **te** followed by any two characters and any file_extension are copied locally to the same named files. This example uses proxy access.

```
dap copy beast::te%%.**.*
```

Local to Remote from Command Mode

- The following is an example of explicit access.

```
dap copy test.c beast"username password"::[tek]test.c
```

- The following is an example of default access.

```
dap copy test.c 13""::user$1:[tek]test.c
```

- In this example, all files beginning with the letters **te** and any file extension are stored remotely to the same name. Proxy access is used.

```
dap copy te*.* beast::*.*
```

  In this example, all files beginning with **te** followed by any two characters and any file_extension are stored remotely.

- This example uses proxy access.

```
dap copy te%%.* beast::*.*
```

Remote to Local from Interactive Mode

- This example shows explicit access

```
DAP> copy beast"username password"::[tek]test.c test.c
```

- The following is an example of proxy access.

```
DAP> copy beast::te*.* *.*
```

- In this example using default access, the filenames are displayed to verify to the user that the command was completed.

```
DAP> copy -log 13""::user$1:[tek]test.c test.c
```

- In this example, the **-replace** option is used to delete the existing local file before retrieving the file from the remote system.

```
DAP>copy -log 13""::user$1:[tek]test.c test.c -replace
```

- This example appends the source file to the destination file on the remote system.

```
DAP> copy -log 13""::user$1:[tek]test.c test.c -append
```

Local to Remote from Interactive Mode

- In this example a local file is copied and then printed on the remote system's default printer.

```
DAP> copy -print test.c 13"":::user$1:[tek]test.c
```

- In the following example a local file is copied to the remote system, printed on the remote system's default printer and then deleted once the print operation is completed.

```
DAP> copy -print -delete test.c
13"":::user$1:[tek]test.c
```

- In this example a file is created on the remote system with a **stream_cr** record format and RETURN carriage control record attributes.

```
DAP> copy -cr test.c beast"username
password"::[tek]test.c
```

dap

Description:

dap is a file manipulation program that allows the ES/Dnet system and the DECnet node to share files. The commands include file transfer, file management, general utility, and batch submittal.

Access Control:

VMS access control information limits access to directories and files on the remote system. This access control is defined with:

```
node "username password "::
```

There are three basic ways to include the access control information with network requests:

Explicit access	Include the access information, *i.e.*, the node name, username, and password, with every request.
Proxy access	An account is set up on the remote system that contains your node name and username mapped to a proxy name. Use the proxy instead of the node name, username, and password.
Default access	The access information is defined in files on the VAX and the ES/Dnet system. If there is no explicit access information in the command line, proxy access is assumed by the remote system.

If proxy is not defined on remote system, the default access specification is used.

Modes of Operation:

The **dap** program has two modes of operation: command line and interactive. In command line mode, you enter the **dap** command with all desired options and arguments, followed by a RETURN. In interactive mode, you enter the **dap** commands, options, and arguments in response to a **DAP>** prompt. In either mode, if you do not enter necessary information, **dap** prompts you for it and waits for input.

Syntax Considerations:

With ES/Dnet you can access files on any system that is running DECnet protocols. All file specifications for a remote VMS file must be in the standard VMS form of:

```
node ::device :[directory .subdirectory ]file_name
.file_type ;version .
```

device could be a logical name such as **sys$system** or **USER$1:** or an actual device such as **DRA0:**. A directory can be one or more subdirectories.

Commands:

Table 5-3. dap Utility Command Summary

Command	Description
cd	**-log** Changes the current working directory on the remote node.
copy	**-append, -ascii, -bin, -crc, -fix, -log, -nblock, -noattributes, -nocrc, -replace , -statistics, -transmit_buffer** Copies a remote file to a local file.
copy	**-ascii, -binary, -cr, -crc, -fix, -hash, -log, -nblock, -nocrc, -print, -print -delete , -statistics -transmit_buffer** Copies a local file to a remote file.
default	**-clear** Sets the default access control for the remote node.
delete	**-log** Deletes a file at the remote node.
directory	**-full, -size, -date, -ufull** Lists the contents of a remote directory.
exit	(none) Exits the **dap** program.
get	**-append, -ascii, -bin, -crc, -hash, -log, -nblock, -noattributes, -nocrc, -replace, -statistics -transmit_buffer** Copies a file from the remote node.
help	(none) Displays help text for individual commands.
print	**-log** Prints a file at the remote node.
pwd	(none) Lists the current directory
rename	**-log**

Table 5-3. dap Utility Command Summary (con't.)

Command	Description
	Renames a file at the remote node.
send	**-ascii, -binary, -cr, -crc, -fix, -hash, -log, -nblock, -nocrc, -print, -print -delete, -statistics, -transmit_buffer** Copies a file to a remote node.
shell	(shell commands enclosed in double quotes) Lets you enter shell commands.
show	(none) Displays current default access control for the remote node as defined by default or the system variable DAP_DEFAULT.
submit	**-log** Submits a batch file for remote execution.
type	**-log** Display remote files on the screen.
usage	(none) Displays usage text for individual commands.

Options:

Table 5-4. dap Utility Command Option Summary

Option	Description
-append	Appends a source file to a destination file after retrieving it from a remote system.
-ascii	The file is created on the VAX with **stream_lf** record format and RETURN carriage control record attributes.
-binary	The file is created on the VAX with fixed length record format. When network is remote, no conversion is done on file.
-clear	Clears the current default access string.
-cr	The file is created on the VAX with **stream_cr** record format and RETURN carriage control record attributes. (The file must have RETURN as record delimiters.)
-crc	**dap** CRC is applied and calculated (default).
-date	Will return file creation date on **dir** command.

Table 5-4. dap Utility Command Option Summary (con't)

Option	Description
-fix	Specify the record size of the fixed length file on destination.
-full	On a directory command displays record format, record attributes, owner, protection, etc., in VMS **dir** (Full-Format).
-hash	Display "#" for every 150 KBytes transferred.
-log	Displays filenames, verifying that a user command was completed.
-nblock	Sets the size of **dap** reads and writes to the network. The default is 7. The read and write operations are 3644 bytes long. This includes 60 bytes of **dap** overhead and 7*512 bytes of user data.
-noattributes	VAX file record attributes are ignored when creating the file on the local system.
-nocrc	Disables **dap** CRC.
-print	Copies a file to the remote system and then prints it on the remote system's printer.
-print -delete	Copies a file to the remote system, prints it on the remote system's printer, and then deletes the file once the print operation is finished.
-replace	Retrieves a remote file and deletes any local file of the same name.
-size	Specifies number of bits and blocks of file on **dir** command.
-statistics	Displays file transfer statistics such as bytes per second, read and write operations, etc.
-transmit_buffer	Specifies the size of the **dap** transmit buffer. By default this is 1024.
-ufull	Specifies UNIX is full. Creates date, size, and protection of file on **dir** command.

Options may be placed in any position in the command string.

default

Syntax:

default [-clear] ***access-spec***

Description:

default is used to define the remote access control for all commands except copy. The **copy** command requires you to distinguish between which file is remote and which is local for source. A copy (retrieve) has the remote file as the source. A copy (store) has the local file as source.

The ***access-spec*** specifies the access control for the file transfer as `node "user password"::`. After the default access control has been defined, you only need to enter the remote file-spec with the commands.

The **show** command or the **default** command without argument display the current default access control.

The **default** command should be used from interactive mode.

Options:

-clear Clears any existing defaults.

Example:

From Interactive Mode

```
DAP> default 13"danw wattsd"::
```

delete

Syntax:

delete [-log] ***remote destination-file***

Description:

delete deletes the ***remote destination-file*** at the remote node, such as a VAX DECnet system.

Options:

-log	Displays remote deleted filename verifying user command was completed.

Example:

From Command Mode

This example deletes the file **test.c**. It accesses the remote system explicitly. The **-log** option lets the user know what has taken place.

```
dap delete  13"tek tek"::test.c -log
USER$1:[TEK]TEST.C was deleted
```

From Interactive Mode

In this example, the remote system is accessed by default, and the file **test.c** is deleted silently.

```
DAP> delete beast::test.c    (default access)
```

directory

Syntax:

directory [-date] [-full] [-size] [-ufull]*directory*

Description:

directory generates a directory listing for a remote directory such as a VAX DECnet system.

Options:

-date	Displays creation date of file.
-full	Displays complete file information in a VMS format including protection, size, record attribute and format, ownership, allocation, creation, revised, backup, organization, access control list, and journaling.
-size	Displays size of file.
-ufull	Displays size, date and protection of file.

Example:

The following two examples give a directory listing of the remote system, which is referred to generically.

From Command Mode

```
dap dir remote::[tek]*.*
```

From Interactive Mode

```
DAP> dir remote::[tek]*.*
```

dna_stop

Syntax:

dna_stop

Description:

dna_stop causes the ES/Dnet Agent to terminate. Any active DECnet links are closed.

To stop ES/Dnet from the command line, enter **dna_stop**. ES/Dnet stops, and the **dna_stop** network command utility program returns a message indicating the ES/Dnet has stopped.

dnamail

Syntax:

dnamail [-d] [-r] [-s *subject*] [-t *text_filename*] [-n *node*] [address_list]

nodename

username

subject

message

<EOF>

Description:

dnamail sends a mail message in VMS format to a VMS or an ES/Dnet implementation. It is possible to send a file as well. To send a file, you must use the **-t *text_filename*** option.

Options:

-r	Uppercase remote names
-d	Turn on Debug
-s *subject*	Specify subject
-n *node*	Specify node
-t *textfile*	Specify file to send

Example:

```
dnamail
viper: dnamail
Node: ashton
To: atuser
Subject: Great news!
Enter your message below.  Press <EOF> when complete,
or <ABORT> to quit.
The UNIX mail to VMS mail gateway is now up on our
system.  We can now send mail to each other using
this.
Talk to you later!
Howard
*** EOF ***
viper:
```

exit

Syntax:

exit

Description:

exit exits the **dap** program.

get

Syntax:

get [-append] [-ascii] [-bin] [-crc] [-hash] [-log] [-nblock] [-noattributes] [-nocrc] [-replace] [-statistics] [-transmit_buffer]
remote source-file local destination-file

Description:

get copies files from a remote node such as a DECnet VAX to the ESV Workstation.

Options:

-append	Appends source file to destination file on retrieval from remote system. The destination file must exist.
-ascii	The file is creates on the VAX with stream-lf record format and RETURN carriage control record attributes.
-bin	The file is created on the VAX with fixed length record format. When network is remote, no conversion is done on file.
-crc	**dap** CRC is applied and calculated (default).
-hash	Display "#" for every 150 KBytes transferred.
-log	Displays filenames verifying user command was completed.
-nblock	Sets the size of **dap** reads and writes to the network.
-noattributes	VAX file record attributes are ignored in creating the file on local system during **copy**, **retrieve**, or **get** commands.
-nocrc	**dap** CRC is not applied. Recommended when transferring source code files.
-replace	Deletes an existing local file before retrieving a file from the remote system.
-statistics	Displays file transfer statistics such as bytes per second, and read and write operations.
-transmit_buffer	Specifies the size of the dap transmit buffer. By default this is 1024.

Examples:

In all the following examples, the remote system is referred to generically as remote node.

From Command Mode

- In this example, all files with **te** and any file extension present are created locally with the same names.

```
dap get remotenode::te*.*
```

From Interactive Mode

- This example shows user specified access control using the interactive **dap** default command.

```
DAP> get test.c
```

- This example displays the filenames verifying that the user command was completed.

```
DAP> get -log remotenode::rfilename lfilename
```

- In this example the existing local file is deleted before a file is retrieved from the remote system.

```
DAP> get remotenode::rfilename -replace
```

- In this example the source file is appended to the destination file after retrieval from the remote system.

```
DAP> get remotenode::rfilename -append
```

or

```
DAP> get -append nodeA::test.app newtest.c
```

help

Syntax:

help

Description:

help displays help text for individual commands.

loop

Syntax:

loop [-max=#] [-pattern=##] [-repeat=#]

Description:

loop is a program that tests DECnet connectivity up to and including the application layer. Using the **loop** command you can verify that two programs on different (or the same) nodes can communicate across DECnet protocols. It verifies that data does not become corrupted by comparing what was sent and what was received from **mirror**.

loop requests a connection to the mirror object on ES/Dnet implementations or the VAX DECnet.

The **loop** network command utility program verifies that the local and remote network software is functioning properly. The **loop** network command utility program tests all layers of the network software.

The output from **loop** will include the size in bytes of the available maximum ES/Dnet buffer, the requested connection's (mirror) maximum buffer, **loop**'s maximum buffer, the maximum buffer actually used for this test, the test pattern which is by default a series of alternating 1s and 0s (55), and the number of times the test is repeated. The **loop** test sends messages starting at 256 bytes and going up to the maximum of 4096 bytes.

Options:

-max=#	The maximum length message to send.
-pattern=##	The hexadecimal number to use as the test pattern.
-repeat=#	How many times to repeat each message length.

Example:

In the following example, a maximum size is set at 2048 bytes. If you want to loop only once, set **-max=256**.

```
loop  -max=2048 viper::
```

```
loop -- DECnet Loop Test Program
DECnet Max I/O Size:        4096 bytes
Mirror Buffer Size:         4096 bytes
Loop Max Buffer Size:       2048 bytes
Using Max Buffer size:      2048 bytes
Test Pattern:               0x55
Repeat times:               1 per buffer size
testing  256 bytes...OK!
testing  512 bytes...OK!
testing  768 bytes...OK!
testing 1024 bytes...OK!
testing 1280 bytes...OK!
testing 1536 bytes...OK!
testing 1792 bytes...OK!
testing 2048 bytes...OK!
Loop Test Successful
```

mirror

Syntax:

mirror

Description:

mirror is the DECnet Loopback Object. The loop program connects to this object. To loop from the VAX, use the NCP **loop node *<node name>*** command. The NCP loop node command connects to this mirror object. Usually, you do not need to run this object, because ES/Dnet runs it automatically when it receives the VAX DECnet's or ES/Dnet's loop program connection request.

Because you do not often run this yourself, the output from the mirror network utility command program may be found in the **net_*<link_id>*.log** file. **link.id** is a unique 8-digit hexadecimal number. You may want to run **mirror** from the command line by entering **mirror** at the command line prompt. If you suspect that **mirror** is not being started properly by ES/Dnet, run it from the command line or consult the **net*<link_id>*.log** file in the ES/Dnet directory for error messages.

Example:

```
mirror

MIRROR - DECnet Loopback Mirror.  Object Number 25

Waiting for connection
*** Connection received ***
Agent_id= 0x0001
Max size= 4096
Access control:
Remote Node:       'viper'
Remote Process:    'dwatts'
Username:          ''
Password:          ''
Account:           ''

Connect Data: 0 bytes
0000:
```

```
MIRROR calling NREAD
: Error o150050 (0x0000D028, 53288.)
: I/O Terminated By Close

Received 34,816 bytes in 5.617 seconds
  (6,198 bytes/sec, 49,584 bits/sec)
Received 16 messages (2.849 msgs/sec)
```

The above error is acceptable, because the **loop** application closed the link.

pconfig

Syntax:

pconfig [-commands=] [-commands=*pathname*] [-ncp]

Description:

The **pconfig** utility provides for DECnet Network Management in ES/Dnet. You must be superuser to run **pconfig** .

pconfig Menu:

The following table summarizes the **pconfig** utility's configuration commands.

Exit with Changes Made	Exits **pconfig** and makes changes.
Abort Without Making Changes	Aborts **pconfig** without making any changes to the database. You will be prompted as to whether to abort or not.
Define LAN Device Name	Set Ethernet interface name.
Define Local Node Number	Defines the node number for your local node. When ES/Dnet starts up, it defines itself to be this node number. This number must be unique across the network and is usually assigned by the network manager.
Node Maintenance	Add, modify, list or remove node names and numbers from the network database. If you want to reference a remote node by its name, you must define it using this command.
ES/Dnet Object Maintenance	Lets you stipulate programs that will be started by the ES/Dnet agent by number. The range is 128 to 255 for non-DEC applications and 0 to 127 for DECnet-known objects.

Make Command File	Generates a command file that contains all the definitions for the node and object databases. This command file is made up of NCP format commands. You can edit and add commands to the command file which is created by this command.
Use NCP-Like Command	Provides a command line interface instead of the menu interface. The command format is the same as Digital Equipment Corporation's DECnet utility Network Control Program (NCP). Commands are limited to configuration commands and the **SHOW EXECUTOR** and **SHOW TIMERS** commands.

Options:

-commands=	Reads commands from the file **ncp.cmd**. When you use this switch, it displays the NCP configuration commands as they are executed.
-commands=*pathname*	Reads commands from the file specified by pathname.
-ncp	Runs in command line mode instead of screen mode. Accepts commands that are compatible with DEC's NCP program. Gets you the **NCP>** prompt.

print

Syntax:

print [-log] ***remote-file***

Description:

print prints a remote file at a remote node such as a VAX DECnet system on the default system printer.

Options:

-log	Displays filenames verifying user command was completed.

Example:

In all the following examples, the remote system is referred to generically as remote.

From Command Mode

```
dap print remotenode::[tek]print.file
```

From Interactive Mode

In this example, the printed files are listed to verify that the command was executed.

```
DAP> print remotenode::[tek]print.file -log
REMOTE_FILE:[TEK]PRINT.FILE;21 was printed
```

pwd

Syntax:

pwd

Description:

pwd prints the pathname of the local working (current) directory.

Example:

```
DAP> pwd
```

rename

Syntax:

rename [-log] ***old_filename new_filename***

Description:

rename renames a file on a remote node such as a VAX DECnet system.

Options:

-log	Displays old and new filenames verifying user command was completed.

Example:

In all the following examples, the remote system is referred to generically as remote.

From Command Mode

```
dap rename remote::oldfile.name new_file.name
```

From Interactive Mode

This example displays the filenames to verify that the command was completed.

```
DAP> rename remote::oldfile.name new_file.name -log
```

send

Syntax:

send [-ascii] [-binary] [-cr] [-crc] [-fix=#] [-hash] [-log] [-nblock] [-nocrc] [-print] [-print-delete] [-statistics] [-transmit_buffer]
local source-file remote destination-file

Description:

send copies files from the ESV Workstation to a remote node such as a VAX DECnet system.

Options:

-ascii	File is created on VAX with **stream_lf** record format and RETURN carriage control record attributes.
-binary	File is created on VAX with fixed length record format of 512 bytes.
-cr	File is created on VAX with a **stream_cr** record format and RETURN carriage control record attributes.
-crc	**dap** CRC is applied and calculated (default).
-fix	Specify the record size of remote fixed length file.
-hash	Display '#' for every 150 KBytes transferred.
-log	Displays local and remote filenames verifying user command was completed.
-nblock	Sets the size of **dap** reads and writes to the network.
-nocrc	**dap** CRC is not applied. Recommended when transferring source code files.
-print	Copies file and then prints it to default remote system's printer.
-print -delete	Copies file to remote system, prints it to default printer and then deletes it once print operation has completed.
-statistics	Displays file transfer statistics such as bytes per second, read and write operations.
-transmit_buffer	Specifies the size of the buffer.

Examples:

In all the following examples, the remote system is referred to generically as remote.

From Command Mode

- The following is an example of explicit access.

```
dap send.c beast"username password"::[tek]test.c
```

- This is an example of default access.

```
dap send test.c 13"":::user$1:[tek]test.c
```

- In this example all files beginning with the letters **te** and any file extension are stored remotely to the same name. Proxy access is used.

```
dap send te*.* beast::*.*
```

- In this example all files beginning with **te** followed by any two characters and any file_extension are stored remotely. This example uses proxy access.

```
dap send te%%.* beast::*.*
```

From Interactive Mode

- In this example a local file is copied, then printed on the remote system's default printer.

```
DAP> send -print test.c 13"":::user$1:[tek]test.c
```

- In the following example, a local file is copied to the remote system, printed on the remote system's default printer, then deleted after the print operation is completed.

```
DAP>send -print -delete test.c13"":::user$1:[tek]test.c
```

- In this example, a file with a **stream_cr** record format and carriage return carriage control record attributes is created on the remote system.

```
DAP> send -cr test.c beast"username
password"::[tek]test.c
```

sethost

Syntax:

sethost [-log] [-scriptfile] ***node***

Description:

sethost creates a virtual terminal connection from an ES/Dnet node to a remote DECnet node. The remote DECnet node is specified with the node argument using the DECnet node name.

Logging off the remote VAX terminates **sethost** and the DECnet session.

Options:

-log =*pathname*	Keep a log of session. Default pathname is **sethost.log**
-scriptfile =*name*	Allows user to put a login script in a file. This includes username, password, and any command.

Examples:

This example starts a terminal session on the node `remote::`.

```
sethost remote::
sethost -script=convax.ser remote::
```

where `convax.ser` has

```
shownet.
```

shell

Syntax:

shell [shell commands enclosed in double quotes]

Description:

shell allows you to suspend **dap** and go to a command line level or issue shell commands. To issue **shell** commands from Interactive Mode without leaving the **dap** prompt, enter the desired shell commands in double quotes. If you want to leave the **dap** prompt and go to the **shell** prompt, simply enter **shell** without any argument. From the command line level you can enter any commands accepted by your system. To return to the interactive **dap** mode, **logout** from the shell.

Examples:

From Command Mode

```
shell "ls -l *.*"
```

From Interactive Mode

```
DAP> shell
$ pwd
/usr/dwatts
$ logout
DAP>
```

This can also be entered as follows:

```
DAP> shell "pwd"
/usr/dwatts
DAP>
```

show

Syntax:

show

Description:

show displays the default remote node access control that was defined with the **default** command. The access control specifies the **node"user password"::**. If no default has been assigned, it displays **No Default Set**.

Examples:

In all the following examples, the remote system is referred to generically as remote.

```
DAP> show
remote"danw wattsd"::
```

In this example no default has been set.

```
DAP> show
No Default Set
```

showcir

Syntax:

showcir [-active] [-known] [-zero]

Description:

showcir displays the circuit/line counters for the local ES/Dnet node. After they have been read, these counters may be zeroed using the **-zero** option.

Options:

-active	List only active items (default)
-known	Include all circuits
-zero	Zero counters after reading

Example:

```
showcir  -zero

SHOWCIR

DECnet circuit counters for local node 1.9 viper on
Fri Sep 2  11:-1:54  1988
Up since:  Fri Sep  2  10:56:37  1988
Up time:   5 minutes  17 seconds

Circuit Counters

data_bytes_rcvd:  0,       data_bytes_sent:  0
mc_bytes_rcvd:    1960     mc_bytes_sent:    1940
bytes_rcvd:       1960     bytes_sent:       1940
dat_frames_rcv:   0        dat_frames_sent:  0
mc_frames_rcv:    20       mc_frames_sent:   20
frames_rcvd:      20,      frames_sent:      20

send_1_collision: 0,       send_multi_coll:  0
send_failure:     0,       collision_failure:0,
no_carrier:       0
loss_of_cts:      0,       dma_underrun:     0
too_many_retries: 0,       no_sys_buffer:    0,
no_user_buffer:   0
```

Table 5-5. Circuit Counter Description Table

Field	Description
data_bytes_rcvd	The total number of data bytes successfully received. This does not include Ethernet data link headers. This number is the number of bytes in the Ethernet data field, which includes any padding or length fields when they are established.
bytes_sent	The total number of data bytes transmitted.
mc_bytes_rcvd	The total number of multicast data bytes successfully received. This does not include Ethernet data link headers. This number is the number of bytes in the Ethernet data field.
mc_bytes_sent	The total number of multicast data bytes successfully sent. This does not include Ethernet data link headers. This number is the number of bytes in the Ethernet data field. This consists of endnode hello messages.
bytes_rcvd	The total number of user data bytes successfully received. This does not include Ethernet data link headers. This number is the number of bytes in the Ethernet data field, which includes an padding or length fields when they are established. Both multicast and "single cast."
bytes_sent	The total number of user data bytes successfully transmitted. This does not include Ethernet datalink headers or datalink generated retransmissions. This number is the number of bytes in the Ethernet data field, which includes any padding or length fields when they are enabled. Both multicast and "single cast".
dat_frames_rcv	Of the total, the number of frames that were single-casted.
dat_frames_sent	Of the total, the number of frames that were single-casted.
mc_frames_rcvd	The total number of multicast frames successfully received.

Table 5-5. Circuit Counter Description Table (con't.)

Field	Description
mc_frames_sent	The total number of multicast frames successfully sent. This consists of endnode hello messages.
frames_sent	The total number of frames successfully transmitted. This does not include data link generated retransmissions.
frames_rcvd	The total number of frames successfully received. These are frames that passed hardware filtering.
send_1_collision	The total number of times that a frame was successfully transmitted on the second attempt after a normal collision on the first attempt.
send_,multi_coll	The total number of times that a frame was successfully transmitted on the third or later attempt after normal collisions on the previous attempts.
send_failure	The total number of times a transmit attempt failed.
collision_failure	The approximate number of times that collision detect was not sensed after a transmission.
no_carrier	The number of times that a transmit failed due to No Carrier Sense signal during transmission.
loss_of_cts	The number of times that a transmit failed due to loss of the Clear-to-Send signal.
dna_underrun	The number of times that a transmit failed due to a DMA underrun, (*i.e.*, data not supplied from the system bus for transmission quickly enough).
too_many_retries	The number of times that a transmit failed due to an excessive number of collisions (16).
no_sys_buffer	The number of packets that were dropped because there was no ES/Dnet buffer in which to place it.
no_user_buffer	The number of packets that were dropped due to there not being a user buffer in which to place it. This will always be 0 for ES/Dnet.

shownet

Syntax:

shownet [-active] [-full] [-known] [-system] [-zero] [-ncp] ***node_spec***

Description:

shownet displays a list of nodes or configuration information and their characteristics. By default, the list contains only those nodes that are currently reachable on the local LAN. The characteristics information displayed contains the node type, node number, node name, hops, and next hop to node.

The ***node_spec*** is used to determine which nodes are to have information displayed.

Options:

-full	Generate a full information display.
-active	List only active items (default).
-known	Include all nodes even if not currently reachable.
-system	Display node system type if an ES/Dnet node.
-zero	Zero counters after reading.
-ncp	Displays information in ncp compatible format.

Example:

```
shownet
SHOWNET
DECnet Network status for local node 1.9 viper on Fri
Sep  2 11:04:29 1988
Designated Router: *** None ***
Up since: Fri Sep  2 10:56:37 1988
Up time:  7 minutes 52 seconds

TypeNode     Hops           Next Hop to Node
Endnode      1.9 viper      1 (Local) ->1.9 viper
Endnode      1.100 buster   1 (Local) -> 1.100 buster
Total of 2 nodes.
```

```
shownet  -sys
SHOWNET
DECnet Network status for local node 1.9 viper on Fri
Sep  2 11:06:00 1988
Designated Router: *** None ***
Up since: Fri Sep  2 10:56:37 1988
Up time:  9 minutes 23 seconds

TypeNode     Hops         Next Hop to Node
Endnode      1.9 viper    1 (Local) ->1.9 viper

System:      Tektronix, Inc: ES/Dnet
Total of 1 nodes.
```

```
shownet  -full viper
SHOWNET
DECnet Network status for local node 1.9 viper on Fri
Sep  2 11:06:50 1988
Designated Router: *** None ***
Up since: Fri Sep  2 10:56:37 1988
Up time:  10 minutes 13 seconds

TypeNode      Hops       Next Hop to Node
Endnode       1.9 viper  1 (Local) ->1.9 viper

System:                 Tektronix, Inc: ES/Dnet
Station  Address:       AA-00-04-00-09-04
Hardware Address:       08-00-1E-01-20-D9
Hello Timer:            15 seconds
On-Cache:               TRUE

                        Received        Transmitted
Data Bytes:             0               0
Messages:               0               0
Connects:               0               0
Connects Rejected:      0
Timeouts:               0
Local Flow Offs:        0
Remote Flow Offs:       0

Total of 1 nodes.
```

showobj

Syntax:

showobj [-ncp]

Description:

In order to have ES/Dnet start an object by number, the object must be defined by **pconfig** or **pconfig -ncp**. The object becomes "known" at the time of definition. A number is associated with the file/executable when a remote request to connect to that object is received by the ES/Dnet Agent. The agent looks to see if the object is running and if its access control is the same. If so, the agent connects to this object. If the object is not running, the agent looks to see if it is a known object. If it is a known object as defined by **pconfig**, the agent looks for the file to execute associated with the request connect to that object. If access control is acceptable, the object is started. 0-127 numbers are reserved by ES/Dnet. An example of an ES/Dnet known object is 17 or **d_server**. An example of a user known object is **OBJ134** which assists in testing **NCLOSES** and **NOPENS**.

Options:

-ncp Displays information in NCP compatible format.

Example:

```
showobj
DECnet status for local node 1.102 viper on Fri Dec 9
12:31:34  1988
Designated Router:  *** None ***
Up since:  Fri Dec  9  09:14:29  1988
Up time: 3 hours 17 minutes 5 seconds
Object      Name        NumberFile
-----------------------------------------
d_server    17          /etc/dna/d_server
mirror      25          /etc/dna/mirror
dnamaild    27          /etc/dna/dnamaild
obj129      129         /etc/dna/etc/dna/obj129
obj130      130         /etc/dna/obj130
obj131      131         /etc/dna/obj131
obj132      132         /etc/dna/obj132
obj133      133         /etc/dna/obj133
obj134      134         /etc/dna/obj134
Total of 10 objects
```

showlink

Syntax:

showlink [-active] [-disconnect] [-full] [-known]

Description:

A link, also known as a virtual circuit or connection, is a connection which exists between your node and itself, or your node and a remote node. It is a communication channel that allows you to pass data between the connected node(s). This data could be your own network application exchanging data between your node and VAX applications, or the **dap** program doing a copy to the VAX, or the **sethost** program establishing a remote terminal session. The End Communication Layer, equivalent to the Open System Interconnect (OSI's) Transport Layer, creates and maintains this connection.

Options:

-active	List only active items.
-disconnect	Disconnect Specified Link(s).
-full	Generate a FULL information display.
-known	List all known items (default).

Example:

```
showlink
DECnet status for local node 1.102 ki on Wed Mar  8
15:14:05 1989
Designated Router: *** None ***
Up since: Wed Mar  8 15:13:43 1989
Up time:  22 seconds
Link 7B00 => 0000, LOBJ 129, PID 337, uinfo: 0000,
State OPEN Usr opn
Total of 1 link.
showlink -active
DECnet status for local node 1.102 ki on Wed Mar  8
15:14:09 1989
Designated Router: *** None ***
Up since: Wed Mar  8 15:13:43 1989
Up time:  26 seconds
No links.
```

```
showlink -full
DECnet status for local node 1.102 ki on Wed Mar  8
15:14:14 1989
Designated Router: *** None ***
Up since: Wed Mar  8 15:13:43 1989
Up time:  31 seconds
Link 7B00 => 0000, LOBJ 129, PID 337, uinfo: 0000,
State OPEN Usr opn

ack_xmt_oth:     0, numoth:      1
ack_xmt_dat:     0, rcv_low:     1, rcv_high: 0,
rcv_pothole:     1
ack_rcv_dat:     0, numlow:      1, numhigh:      0
numdat: 1, numsent:     0
flowrem_sw:  ON, flowrem_dat:      0, flowrem_int: 1
flowloc_sw:  ON, flowloc_state:    0
timerdat:      0, oth:    0, ack:    0, inact:    0
timercon: 0, out:    0, in:     0, alive:    0
Total of 1 link.
```

Table 5-6. Link State Table

State	Description
(O) OPEN	The local Session Control has issued an **OPEN** call which created the port.
(CC) CONNECT-CONFIRM	The local Session Control has issued an **ACCEPT** call.
(CI) CONNECT-INITIATE	The local Session Control has issued a **CONNECT-DMT** call.
(RUN) RUNNING	NSP has either received a Connect Confirm message while in the **CONNECT-INITIATE** or **CONNECT-DELIVERED** state or received a Data, Data Request, Interrupt Request, Data Acknowledgement or Other Data Acknowledgement message while in the **CONNECT-CONFIRM** state.
(DI) DISCONNECT-INITIATE	The local Session Control has issued a **DISCONNECT-XMT** call or an **ABORT-XMT** call.
(DIC) DISCONNECT-COMPLETE	NSP has received either a Disconnect Complete message or a Disconnect Initiate message.
(DN) DISCONNECT-NOTIFICATION	NSP has received a Disconnect Initiate message while in the **RUNNING** state.
(CL) CLOSED	The local Session Control has issued a **CLOSE** call.

showproxy

Syntax:

showproxy

Description:

showproxy displays the proxy database. The format of the display is **remotenode::remote user local user**. If the user does not specify explicit or default access control, the user name and node are passed to remote system. The remote node and user name pair are used to do a line search of the proxy database. If the pair exists or meets the wildcard specifications of a pair in the proxy database, the associated local user account is used in starting known objects.

If no access control is specified and no proxy entry exists, access is denied. The **/usr/etc/esdnet/database/proxy.db** is edited by the network manager with your favorite editor.

If ES/Dnet has been running and changes have been made to **proxy.db**, you must execute the command **updb -force** to have **showproxy** reflect these changes.

Example:

In the following example, **viper::atuser** is an instance of the format **remotenode::remote user** and **atuser** is the local user.

```
showproxy

DECnet status for local node 1.102 viper
 on Fri Dec  9  12:31:37  1988
Designated Router:  *** None ***
Up since:  Fri Dec  9 09:14:29  1988
Up since: Fri Dec  9 09:14:29  1988
Up time:  3 hours  17 minutes  8 seconds

viper::atuser   atuser
encore::atuser   atuser
curly::atuser   atuser
ranger::atuser   atuser
vise::atuser   atuser
beauty::atuser   atuser
buster::atuser   atuser
beast::atuser   atuser
```

```
beast::op  atuser
ranger::ki   atuser
vise::ki   atuser
beauty::ki   atuser
buster::ki   atuser
ranger::sally   atuser
vise::sally   atuser
beauty::sally atuser
```

submit

Syntax:

submit [-log] ***remote filename***

Description:

submit submits a file on the remote node such as a VAX DECnet system for execution.

Options:

-log	Displays filenames verifying user command was completed.

Examples:

In all the following examples, the remote system is referred to generically as remote.

From Command Mode

```
dap submit remotenode::[tek]batch.com
```

From Interactive Mode

In this example, the filenames are displayed to verify that the command was executed.

```
DAP> submit remotenode::[tek]batch.com -log
REMOTE.NODE:[TEK]BATCH.COM;21 was submitted
```

updb

Syntax:

updb

Description:

updb updates the active databases with information from the disk databases located in **/usr/etc/esdnet/database**.

usage

Syntax:

usage

Description:

usage displays usage text for individual commands.

A. Troubleshooting

Steps in Troubleshooting

1) Define the "norm" of the network and the nodes — that is, establish a frame of reference in terms of performance (response time and throughput), expected functionality, and each node's system and non-network load on disks, and other I/O and CPU.
2) If a variation from this defined norm occurs, define the problem in terms of how much different is this problem than the norm, and what has changed since it was working correctly. All involved machines i.e., network router and the endnodes, should be checked.
3) Consider all possible solutions and implement the best choice.
4) Verify the "correctness" of solution by observation and successful repetition of failing user action.

Defining the Norm

You can define the norm of your node and network performance by monitoring response time and throughput.

- Use the VAX/VMS monitor system to determine the CPU and I/O loads.
- Determine the average virtual terminal connect time and file transfer times at differing times of day.
- Determine the time it takes to establish and break connections by transferring a zero-length file at different times of the day.
- Determine the average file transfer time using **copy -stat** and **FAL$LOG** at differing times of day.
- Review, on a regular basis, event logging on the VAX and the console system event messages on the workstation.

Defining the Problem

Define the problem in relation to the norm and consider different solutions. For example, if a user complains that a network file transfer did not work or took too long to complete:

- Try a transfer from a known account. Verify that the user has proper quotas, accounts, network access.
- Try the same transfer to another node to see if the same slow transfer occurs.

- Try the same transfer at different time of day.
- Check the VAX's **netserver.log** and the ESV Workstation's **net<*link_id*>.logs**.

Then

- Determine if the transfer time is within the norm for that time of day. Rerun initial norm tests and compare with current results.
- If one transfer works and another doesn't, compare the differences between the two transfer accounts. If there are differences, check the load averages of each system. (Be sure to check both systems. If one system is slow, the transfer will be slow.)
- Use the monitor system on the VAX to see CPU and I/O utilization.

Pay particular attention to response timeouts on VAX and timeouts on the ES/Dnet system. These indicate retransmissions because an **ACK** was not received for a transmitted message.

Compare node counters of the ESV Workstation on the VAX as the VAX sees them and VAX counters as the ESV Workstation sees them. (Note that to get accurate results, both counters should be zeroed before trying the failing transfer.)

If there are wide differences, see if entering **VAX NCP SHOW LINE <*LINE_ID*>** displays any errors in transmit and receive. Check the same via **showcir** on the ESV Workstation.

Verifying Corrections

- If any network parameters are changed, redo the norm tests and save the results. These results are your new norm.
- Retry the transfer that was failing or slow at different times of the day to assure proper functionality.
- If all tests are successful, the problem is fixed, or at least explained.

Network Troubleshooting

- Problem: The **login** command file on the remote host executes in response to every **dap** command. If the **login** command file on the remote host takes too long to execute, it causes the link to be dropped.

  Solution: Reduce the number of commands in the remote host's **login** file.

- Problem: You are denied access on the remote system.

  Solution: Verify that the specified access control information (username, password, proxy, or default) has an associated account . On the remote host, make sure that the world has E or Execute privileges in the **FAL.COM** file and in any other network **.COM** files. For example, enter:

  ```
  set protection=(W:E)FAL.COM
  ```

- Problem: Account quotas are exceeded on the VAX.

  Solution: Since the user's account quotas are used for remote file request *(i.e.*, **copy**), it is important to make sure each network user has adequate quotas. The following quotas should be assigned using the VAX/VMS **AUTHORIZE** utility. (Make sure the VAX network user has **NETMBX**.)

BIOLM 60	Represents the number of outstanding buffered I/O operations allowed.
DIOLM 60	Represents the number of outstanding direct I/O operations allowed.
BYTLM 60000	The executor's pipeline quota is charged against this number of allowed bytes of I/O to be buffered by the VAX/VMS system on behalf of this account.

- Problem: Slow network and **ACK** for each message sent.

  Solution: On the VAX, turn up the Executor's pipeline quota to 12000. This increase insures better network performance as it allows more transport (NSP) messages to be cached without required **ACK**. For example:

  ```
  NCP> DEFINE EXECUTOR PIPELINE 12000
  ```

- Problem: Circuit bouncing on the VAX, **dap** transfer terminating on the ES/Dnet system.

  Solution: On the ES/Dnet system, it may be necessary to turn up the **node_delay** on heavily loaded VAXs and workstations from 2 to 15 seconds, and to turn up the hello timer from 15 to 60. In very heavily loaded systems, it may need to go as high as 60 seconds. For example,

  ```
  $ pconfig/ncp set node delay 15
  $ pconfig/ncp set hello timer 60
  ```

  Tuning **node_delay** reduces the interval between retransmits and increases the interval at which a message is retransmitted if an **ACK** is not received. Keep in mind by increasing this interval the user will not know as quickly that the link has gone down. ES/Dnet currently retransmits up to 10 times. This retransmit factor is not user-setable. On the VAX, you can accomplish reduction of the interval between retransmits by setting delay factor to 127 and delay weight to 6. For example,

  ```
  NCP> define node node_name delay factor 127
  NCP> define node node_name delay weight 6
  ```

  Or, you can turn up hello timer to 60 seconds, as follows,

  ```
  NCP> define circuit circuit_id hello timer 60
  ```

- Improperly configured VAX will not perform well.

  Check the **sysboot** parameters on the VAX. Common **sysboot** parameters are viewed using **SYSGEN** utility. For example,

  ```
  VAX> run $sysgen

  SYSGEN> show Irpcount
  sho Irpcountv
  sho irpcount
  sho irpcountv
  sho srpcount
  sho srpcountv
  sho npagedyn
  sho pagedyn
  sho maxbuf
  SYSGEN>
  ```

  Use the appropriate values specified in DECnet manuals.

B. Error Codes

This appendix lists each error that can be returned from ES/Dnet. This list includes a brief description of the error's meaning and its cause. All codes have the format of **150xxx**, where **150** is the group number. Numbers are in octal (base 8) format.

150001 P_INV_AGENT_ID

Description: Invalid agent ID passed as argument.

Cause: An invalid agent ID has been passed to a ES/Dnet routine.

150002 P_NO_MORE_CHAN

Description: No more channels available for connecting.

Cause: The maximum number of connections has been reached. No more connections can be made until one or more existing connections are closed.

150003 P_DATA_TRUNC

Description: Data in buffer has been truncated to fit buffer.

Cause: The received data has been truncated to fit into the receive buffer. The truncated data is lost. This error also occurs when written data is too large for the ES/Dnet agent to handle. The truncated data is lost.

150004 P_OLD_AGENT_ID

Description: Attempt to use agent ID that has been closed.

Cause: An agent ID corresponding to a connection that has been closed has been attempted to be used in some ES/Dnet command.

150006 P_NO_DATA

Description: No data ready for reading.

Cause: An **NREAD** has been issued without pend and there is no data.

150007 P_CON_CLOSED

Description: Connection to remote agent has been closed.

Cause: The connection to the remote agent has been closed.

150013 P_NO_AGENT

Description: Requested agent is unavailable at this time.

Cause: The requested agent is not currently running.

150014 P_NO_RRESOURCE

Description: Insufficient resources on remote system.

Cause: There are insufficient ES/Dnet resources on the remote system to accept the connection.

150015 P_NO_LRESOURCE

Description: Insufficient resources on local system.

Cause: There are insufficient ES/Dnet resources on the local system to perform the connection.

150016 P_CLOSED

Description: Agent connection closed.

Cause: The specified agent connection has been closed by either the local system or the remote system.

150017 P_INV_CONPKT

Description: Invalid connection packet.

Cause: The packet that was passed to the **NOPEN** routine contained invalid information. Reasons for the packet to be invalid are either (1) the packet size is not correct for the service requested, or (2) information within the packet contains values that are out of range.

150020 P_UNKNOWN_NODE

Description: Unknown node.

Cause: A connect request has been made to a node that is not defined in the node data base.

150021 P_CON_REJECT

Description: Connection request has been rejected by remote object.

Cause: Connection request has been rejected.

150022 P_NO_OBJECT

Description: The requested object is unknown to the remote node.

Cause: No such object at remote node.

150023 P_BAD_ACCESS

Description: Username/password invalid at remote node.

Cause: Access control information was in error.

150024 P_NO_RESPONSE

Description: No response from remote object.

Cause: No response from remote object.

150025 P_NO_CONNECTION

Description: Unable to make connection to remote object.

Cause: Could not make connection to remote object.

150026 P_SHUT_DOWN

Description: Node is shutting down.

Cause: Either the remote or the local node is shutting down DECnet communications.

150027 P_PROTOCOL_ERR

Description: Internal DECnet protocol error.

Cause: Some sort of error has occurred in the DECnet protocol.

150030 P_TOO_BUSY

Description: Remote object is too busy to accept connection.

Cause: The remote object is too busy to accept connections.

150031 P_3RD_DISC

Description: A third party has disconnected the link.

Cause: A third party has disconnected the link.

150032 P_ABORTED

Description: Remote object aborted the logical link.

Cause: The remote object aborted the logical link.

150033 P_INV_NFORMAT

Description: The remote node name was specified incorrectly.

Cause: An invalid format was used when specifying the remote node name.

150034 P_UNREACHABLE

Description: The remote node is not currently available.

Cause: The remote node is not currently on the network.

150035 P_DISCONNECTED

Description: The remote object closed the link.

Cause: The remote node closed the logical link.

150040 P_INV_LINKID

Description: Invalid session control link ID.

Cause: Invalid link ID for session. Internal error.

150041 P_NO_QRESPONSE

Description: No queued response found from agent.

Cause: A response queue was expected but was not found. Internal error.

150042 P_REV_MISMATCH

Description: Agent interface revision incompatibility.

Cause: The user issued an **NOPEN()** to an agent and the interface revision of the user program and the agent are incompatible. Relink the user program with the ES/Dnet libraries.

150043 P_CPROTOCOL_ERR

Description: CTERM protocol error occurred.

Cause: There was a CTERM protocol error. Internal error.

150044 P_NOT_IMPLEMENTED

Description: Requested operation is not yet implemented.

Cause: The requested operation has yet to be implemented.

150045 P_CNIU

Description: DCON connection is not in use.

Cause: The DCON connection specified is not currently in use. Internal error.

150046 P_INVALID_OP

Description: Invalid agent operation attempted.

Cause: The user specified an operation to ES/Dnet that is not one of the allowed values.

150047 P_NO_INCOMING

Description: No incoming connection found for request.

Cause: Either the **net_in[]** field was incorrectly specified in the user application code, or the link was disconnected before the user application could accept the connection.

150050 P_ERITC

Description: I/O terminated by close.

Cause: The requested I/O operation has been terminated due to the logical link closing.

150051 P_BAD_ARG

Description: Argument error.

Cause: An argument was incorrectly specified.

150052 P_TIMEOUT

Description: **NOPEN** terminated due to timeout.

Cause: The **NOPEN** with **DN_KEEPALIVE** or **DN_NETSERVER** modifier timed out. The timer that relates to this is the keepalive timer.

150053 P_INTERNAL

Description: An internal ES/Dnet error has occurred.

Cause: Unknown.

150054 P_IDATA

Description: Interrupt data has been received.

Cause: The remote network object has written interrupt data. There will be at most 16 bytes of this data.

150055 P_BUFF_TOO_SMALL

Description: **NREAD** buffer is too small.

Cause: The **NREAD** buffer must be at least 16 bytes to be able to read any possible Interrupt Data.

150056 P_NOT_GATEWAY

Description: ES/Dnet agent is not a gateway.

150057 P_DI_DATA

Description: Link Closed. Disconnect data received.

150060 P_SIZE_WRONG

Description: Structure size is wrong. May need to be recompiled.

C. Modifying UNIX sendmail.cf

As an option, you can modify the **sendmail.cf** file on the ESV Workstation to make **dnamail** transparent to the user, *i.e.*, have the same interface as **comp**.

Note: If you have made changes to **sendmail.cf** before, the following modifications will be relatively straightforward. If you haven't, we suggest you consult a UNIX system administrator's guide to familiarize yourself with **sendmail**.

The following procedure shows the changes to the **sendmail.cf** file.

Caution: Make a backup of the original **sendmail.cf** file. You should also comment each line you change.

1) In the **/etc/hosts** file, add **dnahost** as an alternate name for the machine you want to be the DECnet relay. For example,

```
192.9.200.1   nova dnahost
```

If you have more than one machine running ES/Dnet, pick one machine to be the relay for machines without ES/Dnet.

2) Create a file **FS/usr/etc/esdnet/dnahosts** to contain a list of all VMS/DECnet nodes you want to send mail to, one node per line. You do not need to have all the VMS machines listed, just the ones you want to send mail to (a listed machine can deliver mail to a machine that you didn't list). For example, a file might contain,

```
rvax
widowvax
voodoo
```

3) Add the following lines somewhere near the beginning of the **sendmail.cf** file:

```
# Major DECnet relay (relayed by #ether)
DSdnahost
# get list of DECnet machines we want to be able to
send mail to FS/usr/etc/esdnet/dnahosts
```

- The **DS** line defines a macro, **$S**, to be the name of the machine that will actually connect to DECnet. Any machine that can't handle DECnet mail forwards any DECnet mail to this machine. If all your machines are on DECnet, you don't need to define the macro. If you already have a **$S** macro (search for **^DS** in **vi**), choose an unused macro instead.

- The **FS** line defines a class, **$S**, that defines the VMS nodes we can send mail to with **dnamail**.

4) Add **dna** as a Trusted User. To do this, look for the following lines in **sendmail.cf**:

```
Troot
Tdaemon
Tuucp
```

Then, add the following line:

```
Tdna
```

5) In the **sendmail.cf** file, look for the following lines (in Ruleset 0):

```
# resolve UUCP domain
R<@$-.uucp>:$+$#uucp$@$1     $:$2@host.uucp:...
```

Then, add the following lines before them:

```
#  If $S (DECnet gateway) is defined, then forward to
#  $S, else resolve to dna mailer.
R$+<@$=S>$?S $#smtp $@$S $:$2!$1 $| $#dna $@$2 $:$1 $.
R$+<@$=S.uucp>$?S $#smtp $@$S $:$2!$1 $| $#dna $@$2
$:$1 $.
```

If you aren't using **S** as the macro for **dnahost**, substitute the letter you are using in the macro.

6) Add the following lines to the **sendmail.cf** file, either at the end of the file or grouped with the other mailer definitions:

```
# DECnet Mailer specification
# Messages processed by this configuration are
# assumed to leave the internet domain.  Hence, they
# may not necessarily correspond to RFC822 in all
# details.
Mdna,    P=:usr:bin:dnamail, F=mnSF, S=14, R=24,
A=dnamail -r  -f $f -n $h $u
S14
# none needed
S24
# none needed
```

7) After making the changes, kill **sendmail**, then restart it. To kill **sendmail**, you must first determine its process number. One way of finding its process number and killing the process, is as follows (in the example, the process number is 38):

```
$  ps -aux | egrep sendmail
root        38  0.0  2.6  108 76 ?  I 0:30   (sendmail)
$  kill 38
```

Now, restart **sendmail**:

```
$  /usr/lib/sendmail -bz
$  /usr/lib/sendmail -bd -q30m
```

After you perform this optional installation procedure, you can use **comp** as the mail interface within the **dap** program.

- To send mail from the ESV Workstation to a remote VMS node, use an address like:

  ```
  vmsnode!user
  ```

- To send mail from a VMS node to the ESV Workstation, use an address like:

  ```
  UNXNOD::"user"
  ```

  The quotes (") prevent VMS mail from converting everything to uppercase. Longer addresses are just as easy:

  ```
  UNXNOD::"node1!node2!..."
  ```

  (In this example, leaving off the quotes causes the "!" to be treated as a comment character).

- To forward mail from the VAX, use three quotes instead of one. For example,

  ```
  SET FORWARD UNXNOD::"""someuser"""
  ```

D. Glossary

active database. This is the in-memory copy of Sequent ES/Dnet network databases. Also known as the *volatile database.* When Sequent ES/Dnet starts up on a node, it copies the permanent databases into the memory of the ES/Dnet Server. These copies are the active databases. If changes are made to the permanent databases while the server is running, the active databases must be updated.

area. A group of nodes in a network that can run independently as a subnetwork.

application program. (1) A program written for or by a user that applies to the user's network. (2) A program used to connect and communicate with stations in a network, enabling users to perform application-oriented activities. See also *network application.*

ASCII. An acronym for *American National Standard Code for Information Interchange.* The standard code, using a coded character set consisting of 7-bit coded characters (eight bits if the parity check is included), is used for information exchange among data processing systems, data communications systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

batch job. A sequence of command procedures than can be executed without user intervention. The procedures are usually held in a batch command file that is placed in a batch queue to await future execution.

circuit. A communications data path between nodes. A circuit operates over a physical Ethernet connection. An Ethernet circuit can provide connections between a number of nodes. Each node is regarded as being adjacent to every other node on the circuit and is equally accessible.

client. A process or node that requests the service of another process or node. See also *server.*

communications controller. A communications device whose operations are controlled by one or more programs stored and executed in the unit. It manages the details of line control and the routing of data through a network. Sometimes called a *network controller.*

CSMA/CD. An acronym for the *Carrier Sense Multiple Access with Collision Detection* access method, which is used by Ethernet networks. This access method is designed to sense and avoid collision of data during transmission. The data link layer uses "carrier sense" to wait for the network to be idle. If it senses the line is free, it starts to transmit. If, during transmissions, the data collides with another transmission, the protocol causes the transmission to stop and jams the

network for a specified amount of time. The controller waits for random exponential intervals of time, retransmitting until it no longer collides with the other transmission. If the controller tries the same transmission 16 times, the packet is dropped, and a network management counter is incremented.

CTERM (protocol). An acronym for *Command Terminal.* CTERM is the term commonly used when referring to the Terminal Foundation Services and Network Command protocols. The Sequent DEC Bridge CTERM protocol allows the local system to log in to a remote node as if the remote node were attached to the local system. This protocol, together with the DAP protocol, is implemented for the User Layer of the DNA architecture. See *CTERM object.*

CTERM object. An ES/Dnet object that invokes the **dvta** server for incoming login requests.

DAP. An acronym for *Data Access Protocol.* DAP is the ES/Dnet protocol that provides an interface for file and record transfer and access between the local ES/Dnet system and a remote node. DAP is also the name of the ES/Dnet network application that provides the mechanism for making file requests (such as file transfer, print, submit, copy) to remote systems. The DAP application uses the DAP protocol.

databases. The databases that store information about the network configuration. The four databases (*executor, node, object,* and *proxy*) are maintained by the NCP utility. See also *permanent databases* and *volatile database.*

data channel. A data link created by an **NOPEN** Network Procedure Call. The **NWRITE** procedure call writes data over this channel. See also *interrupt channel.*

data link. The physical means of connecting one location to another for the purpose of transmitting and receiving data.

DCL. An acronym for *Digital Command Language.* DCL is used to write command procedures for the VAX/VMS system.

DECnet address. The unique address assigned to each node running DECnet Phase IV protocols. It consists of a DECnet area number and inter-area number separated by a period (for example, 1.11). Also referred to as node number.

DECnet physical address. The unique address by which an Ethernet controller is identified to an Ethernet network when running DECnet. It is comprised of a DECnet vendor code, plus the node's DECnet address, converted to a hexadecimal value (with bytes swapped). See also *Ethernet address* and *hardware address.*

default access. Remote node access information, such as remote node number, remote user name, and remote password. You can store this information using the DAP command default so that it doesn't have to be entered with other DAP commands.

directory. One branch in the hierarchical tree structure by which files are organized on a computer.

DNA. An acronym for *Digital Network Architecture*, which is a layered network design for Digital Equipment Corporation's network products. DNA specifications define the relationship between the various software components of DECnet.

DNAMAIL. A program that lets you send mail in VAX/VMS format. The corresponding program, **dnamaild**, converts mail received in VAX/VMS format to the ESV Workstation format.

d_server. A server process that enables remote users to use the DAP services at your local node. See also *incoming requests*.

dvta. A server process that enables remote users to use remote login to a local node. The **dvta** server is invoked by the CTERM object. See also *incoming requests*.

End Communications Protocol. The protocol used to link two nodes at the End-to-End Communication (ECL) layer.

end node. A node than can only send or receive messages from another node; that is, it cannot forward messages intended for nodes on other networks. Also called a *nonrouting node*.

Ethernet address. The unique address by which an Ethernet controller is identified to an Ethernet network. It can be either the DECnet physical address or the hardware address. It consists of six pairs of hexadecimal numbers in the form **XX-XX-XX-XX-XX-XX**.

executor node. The local node in which the ES/Dnet software is running.

hardware address. The Ethernet address hardwired onto every Ethernet controller.

hello timer. A value specifying how often a node sends out hello messages to routers attached to the same Ethernet network as the local node.

inactivity timer. The length of the timeouts for logical link inactivity at the local node. This value is defined in the node's executor database.

incoming requests. Requests from users on remote nodes to a local ES/Dnet node. To support incoming requests, ES/Dnet provides server processes for DAP and CTERM protocols and the loop command. Each server has an associated object.

incoming timer. The length of time an application has to accept an incoming connection request before the connection ID discarded. This value is defined in the node's executor database.

interrupt channel. A data link channel over which priority messages are sent. Priority messages are sent using the **NIWRITE** procedure call, and arrive sooner than any waiting messages that were sent using the **NWRITE** procedure call.

interrupt data. Data, usually control messages, sent over an interrupt channel.

keepalive timer. The length of time a network object activated by a local node remains active while waiting for additional connection requests.

LAN. An acronym for *Local Area Network.* A LAN is a communications network linking a number of devices that are located within a relatively short distance, typically less than a mile.

LAN Device Name. Identifies the workstation slot in which the Ethernet controller resides.

local node. The node on which the user is logged in.

logical Link. A connection at the user level between two processors on different nodes on the network. Also called a *link.* A logical link is a connection at a higher level than a circuit.

logical name. A character string used to refer to VAX/VMS files or devices by other than their specific names.

loop. An ES/Dnet command that allows users to verify that they have configured ES/Dnet correctly. When a remote user issues the loop command on a local node, the mirror object is invoked.

network. Two or more computers connected together by a cable or other medium and running software that allows them to communicate over that medium.

network application. An ES/Dnet program that supports communication for incoming requests to a local node and outgoing requests to a remote node.

network databases. Databases used by ES/Dnet to store addressing and other network information. See *node database*, *executor database*, *object database*, and *proxy database*.

Network Control Program (NCP). A network utility created by DEC to manage DECnet networks. Also an ES/Dnet utility that allows system administrators to configure ES/Dnet databases and set timer values by using a command-line interface. The ES/Dnet utility is similar to the DEC NCP utility.

network procedure calls. A set of eight C-language procedures that allow user-written applications to access DEC Bridge services.

network utility commands. A set of ES/Dnet commands that allows system administrators to manage an ES/Dnet network.

node. Computers connected to a network. In this manual, the terms *node*, *computer*, and *system* are used interchangeably to refer to computers on a network. Each node in a DECnet network can use data, programs, and devices of other network nodes. Each node contains main memory and may have its own disk or share one with another node.

node database. An ES/Dnet database that contains a list of all the ES/Dnet system, VAX and other computers with their DECnet address and name. ES/Dnet uses this database to associate DECnet node names and addresses.

node name. The name of the node associated with a DECnet node number. It can be six characters long.

node number. A unique address that identifies a DECnet node. A node number consists of a node's area number and inter-area number separated by a period. For example, "1.12" is a valid node number.

normal data. Data sent over the data channel, as opposed to the interrupt channel.

object. Objects provide known general-purpose network services. An object is identified by the object number. ES/Dnet supplies the objects **d_server**, **mirror**, and **dvta**.

object database. An ES/Dnet database that contains a list of objects required to run incoming requests. It lists the object, object number, and the pathname of the server associated with the object.

operating system. A program that manages the hardware and software environment of a computer.

outgoing requests. Requests from a local node to a remote node. The DAP application supports outgoing file requests. The **sethost** application supports remote login requests from a local to remote nodes.

pathname. The "location" of a file in a computer's file system. A UNIX pathname consists of directories and subdirectories separated by slashes, and ends with the file name. For example, the file **finance** might have the pathname **/sub/work/finance**. A VAX/VMS pathname for the same file might have the form **VAX:[sub.work]finance;1**.

PCONFIG. The ES/Dnet database configuration program.

permanent databases. Sequent ES/Dnet network databases that contain configuration information. These are created on disk by using the **pconfig** utility. The permanent databases provide initial values for the ES/Dnet when it starts up. See *active database*.

process. An executing program.

protocol. A set of rules governing communication between DNA layers. Programs with equivalent functions in the same layer but residing on different nodes communicate via protocols.

proxy database. An ES/Dnet database containing local user IDs for remote users' incoming requests. If remote users have entries in the proxy database, they do not need to specify a login name or password when they issue a command.

remote node. Another system located on the same Ethernet network as the local node. It can be a VAX/VMS node or an ES/Dnet system that is running ES/Dnet or another computer running ES/Dnet compatible networking software.

router. A network node than can relay messages from one network area to another. See *end node*.

script. A file that contains one or more shell commands. A script allows you to execute a sequence of commands by entering a single command, the script name.

server. A process or node that performs a defined service on behalf of another process or node. See also *client*.

session. The period of time during which a user of a terminal can communicate with an interactive system; usually the time elapsed between login and logout.

SETHOST. An ES/Dnet network application that provides remote login capabilities. The **sethost** application uses the CTERM protocol.

Shell. A command line interpreter program used to invoke operating system utility programs. UNIX supports several different shells.

VAX/VMS. A Digital Equipment Corporation VAX computer running DEC's VMS operating system.

vendor code. A unique number assigned to each Ethernet interface manufacturer by the Xerox Corporation. To ensure that every Ethernet controller board has a unique address, the Xerox Corporation assigns a block of addresses to each manufacturer. The first 32 bits of the address represent the vendor code.

E. Installation

Before Installation

Before you install ES/Dnet, you must have the following information:

- The DECnet *Node Name* and *Node Number* assigned to your ESV Workstation by your DECnet Manager. These must be unique throughout your network.

  Node Name donut Node Number 1.485

- The DECnet *Node Name* and *Node Number* of another VAX/VMS remote machine which has a copy of the current DECnet Database. You need to have an account on this machine to give you access to the DECnet Database. All users should be able to read the DECnet Database. If you can execute the following command on this remote machine, you have access:

  ```
  MCR NCP SHOW KNOWN NODES
  ```

  Remote Node Name KD7806 Remode Node Number 1.2

- The *Factory Ethernet Address* of your ESV Workstation. Type the following to get this address:

  ```
  /usr/etc/factaddr
  ```

  Factory Ethernet Address 08 - 00 - 57 - 08 - 01 - 08 - __ - __

- The ES/Dnet *Software Key Number* (18 digits). Call Ki Research at **1-800-544-8352** to obtain this number. You will be asked for the *System Type* (ESV Workstation) and the *Factory Ethernet Address*.

  Software Key Number 00 00 00 AE F1 93 E6 20 05

General Notes

- You must be logged on the system as **root**.
- Once ES/Dnet is running, it will change the current Ethernet address of the ESV Workstation to conform with the DECnet Network. Any machines that are running TCP/IP (including NFS mounts) will have to update their Address Resolution Protocol (APR) database with the new Ethernet address. The ARP database is generally updated within 15 minutes. However, the following privileged command will cause the entry for the ESV Workstation to be updated:

  ```
  /etc/arp -d ESV_NAME
  ```

where **ESV_NAME** is the name of the ESV Workstation.

- After installation, the path name **/usr/etc/dna/bin** is used to access ES/Dnet commands.

Loading the ES/Dnet Software

Insert the ES/Dnet tape into the tape drive and enter the following command:

```
/usr/pkg/bin/inst
```

Following is an example which shows the loading of the ES/Dnet software.

```
MIPS software package installation

Install package relative to where [/]?  <RETURN>
Please mount the (first, if multiple tapes) distribution
   tape, then press return... <RETURN>

Rewinding the tape... Verifying tape id... ok

Extracting packaging information tree... esdnet1.0

========== selecting subpackages ==========

Install subpackage esdnet (y n) [n]?  y
Selected subpackages:
   esdnet
Is this what you want (y n) [y]?  <RETURN>

========== setting system clock/calendar ==========

The current value of the clock is: Thu Feb 15 17:13:00
   MST 1990
Is the clock correct (y n) [y]?  <RETURN>

========== verifying single-user mode ==========

This system is not presently in a single-user run level.
   Installation of a package can fail if performed at
   this run level.  We recommend that the system be
   brought to a single user run level (using "init S")
   prior to performing the installation.

Are you absolutely sure you wish to continue (y n) [n]? y

========== preserving local files ==========

No preserve list or findmods list for esdnet- preserve
   not executed.

========== verifying disk space ==========
```

```
The system will now be checked to verify that there is
   enough disk space with the current configuration to
   successfully install the package (and any selected
   optional subpackages). For large packages (especially
   operating system packages), this can be time
   consuming...

There is enough space.

========== stripping old links ==========

Stripping links for subpackage esdnet...

======= extracting files from subpackage archives =======

Rewinding the tape...
Verifying tape id... ok
Forward spacing the tape...

Loading subpackage: esdnet...
Forward spacing the tape...
Rewinding the tape...

========== running comply ==========

running first comply pass...
running second comply pass...
There were no comply messages from the second pass.

========== cleaning up old versions ==========

An attempt will now be made to clean up any files left
   over from previous versions of the software which has
   just been installed.

Searching for old versions to remove...

========== restoring preserved user files ==========

No preserve list or findmods list for esdnet- no files
   restored.

========== cleaning up ==========

Remove install tools (y n) [y]? <RETURN>

========== installation complete ==========
```

Installing the Software Key

Using any editor, create the file **/etc/ki_pwd**. Enter the *Software Key Number* on one line of the file, with no spaces before, or between, any of the 18 digits.

If you have already installed the LAT Host Services software, this file will already exist. In this case, enter the ES/Dnet key before the existing key.

Installing the ES/Dnet Software

Run the installation script, which is located in the following file: **/usr/etc/esdnet/dna/bin/install.dna**.

Following is an example which shows the installation of the ES/Dnet software.

Note: The DECnet name is a maximum of six (6) characters. Therefore, it may be different from your system name.

```
ES/Dnet Installation procedure
ln -s /usr/etc/esdnet/dna /usr/etc/dna
ln -s /usr/etc/esdnet/dna/bin/uninstall.dna
   /usr/etc/uninstall.dna
ln -s /usr/etc/esdnet/dna/bin/install.dna
   /usr/etc/install.dna
ln -s /usr/etc/esdnet/dna/bin/config.dna
   /usr/etc/config.dna

To continue, you must know the DECnet node number and
   name that has been assigned for your host and for a
   remote node (which has a DECnet database).  If you do
   not know these, press <ENTER> and rerun this
   installation later by executing '/usr/etc/install.dna'

Do you wish to continue? [Y/N]
y

What is your DECnet node name [keyhole]?
keyhol

What is your DECnet area number [1]?
1

What is your DECnet node number?
59

Defining local node 'keyhol' as DECnet address 1.59

What is the remote DECnet node name?
rd780b

What is the remote DECnet area number [1]?
1

What is the remote DECnet node number?
2

Defining remote node 'rd780b' as DECnet address 1.2

Saving configuration commands in
   /usr/etc/esdnet/dna/database/ncp.cmd

Would you like to see the ncp.cmd file now? [Y/N]
```

```
y
!
! File Generated By ES/Dnet Version 1.4.3
! Generated On Thu Feb 22 18:24:21 1990
!
!
! First we should let DECnet know what our local address
! is:
Define Executor Address 1.59

!
! OK, Now for the Executor Characteristics:
Set Executor Buffer Size 1461
Set Hello Timer 15 Seconds
Set Inactivity Timer 60 Seconds
Set Incoming Timer 120 Seconds
Set Outgoing Timer 90 Seconds
Set Keepalive Timer 300 Seconds

!
!Need to know the LAN Device Name
Set LAN Device Name la0

!
!And now to define all the nodes:
!
Set Node  1.2    Name rd780b
Set Node  1.2    Hardware Address AA-00-04-00-02-04
Set Node  1.2    Identification rd780b

Set Node  1.59   Name keyhol
Set Node  1.59   Hardware Address AA-00-04-00-3B-04
Set Node  1.59   Identification keyhole

!
! And now to define all the objects:
!
Set Object d_server Number 17 File
   /usr/etc/dna/bin/d_server
Set Object mirror Number 25 File /usr/etc/dna/bin/mirror
Set Object dnamaild Number 27 File
   /usr/etc/dna/bin/dnamaild
```

```
If you want to know about options that can be used when
   starting ES/Dnet, you can use the "-help" option.  ie:
   % deknet -help
Would you like to see this help display now? [Y/N].
n
Installation complete
```

To start ES/Dnet, type the following:

```
/etc/init.d/deknet start
```

Downloading the DECnet Database

To download the DECnet Database, run the following script:
/usr/etc/config.dna

The script will ask for the *user name*, *password*, and *node name* of the VAX/VMS machine. Following is a typical example.

```
Enter username for remote node
rbass
Enter password for remote node
(password will not be echoed)
Enter remote node name
rd780b
```

The script will now update the DECnet Database without any more input.

```
Username: RBASS
Password:
Welcome to VAX/VMS version V5.2 on node RD780B
Last interactive login on Thursday, 15-FEB-1990 15:48
Last non-interactive login on Wednesday, 14-FEB-1990
   19:27
(RD780b) $
(RD780b) $
(RD780b) $
(RD780b) $
(RD780b) $ ncp :== $ncp
(RD780b) $ ncp show known nodes
Known Node Volatile Summary as of 15-FEB-1990 16:38:35
Executor node = 1.2 (RD780B)
State                     = on
Identification            = DECnet-VAX V5.2,  VMS V5.2
```

```
Node            State       Active  Delay   Circuit   Next
                            Links                     Node
1.1 (PARK)      unreachable
1.3 (TRAIN)     reachable                   UNA-0     1.3 (TRAIN)
1.4 (CAD780)    reachable                   UNA-0     1.4 (CAD780)
1.306 (SPVAX2)  reachable                   UNA-0     1.35 (ESIMSA)
1.309 (CTVS09)  reachable                   UNA-0     1.309 (CTVS09)
```

```
(RD780b) $ logout

RBASS        logged out at 15-FEB-1990 16:38:46.18

SETHOST for ES/Dnet version 1.4.3

Remote Session Terminated.  Returning control to local
   system
Using command file 'test.cmd'

NCP> !
NCP> ! File Generated By CVTNCP on Thu Feb 15 16:41:18
   1990
NCP> ! Released with ES/Dnet Version 1.4.3
NCP> !
NCP>
NCP> Set Node 1.2 Name RD780B
NCP> Set Node 1.2 ident DECnet-VAX V5.2,  VMS V5.2
NCP> Set Node 1.1 Name PARK
NCP> Set Node 1.3 Name TRAIN

NCP> Set Node 1.306 Name SPVAX2
NCP> Set Node 1.309 Name CTVS09
NCP> *** EOF ***
Updating database with changes

NCP Execution Complete
```

Update the running database with the following command:

/usr/etc/dna/bin/updb