10703

DNLS PRELIMINARY REFERENCE GUIDE

Augmentation Research Center
Stanford Research Institute
Menlo Park, California 94025

PREFACE

This document is essentially a collage of DNLS documentation culled from various sources. It does not pretend to be definitive, but should suffice to equip the new DNLS user with a working command vocabulary and an orientation to the display mode.

Related documents:

For information about TENEX and the Executive Command set:

TNLS User Guide (7470.)

This document contains many features common to both TNLS and DNLS that are not documented here, e.g., EXEC Commands, a subset of the Output Processor Directives, and Error Messages.

For information about user programs and content analysis:

L10 Programming Guide (9246.)

This document is intended as an introduction to writing user programs and content analyzer patterns. It assumes a degree of sophistication in DNLS usage.

For information about the Journal:

NIC Journal User Guide (7635.)

This document describes the features of the current Journal System. The Journal may be used only through TNLS.

For hardcopy formatting directives:

Output Processor User Guide (6978.)

This document contains a summary of all current Output Processor Directives. Novice users are urged to consult the Output Processor Section of the TNLS User Guide (see -- 7479,) before attempting this document.

For the latest DNLS information:

Folklore Branch of DNLS Status File (nls.status,1)

Users are urged to consult the first branch of this file for information about new DNLS commands, changes, etc.

For creating NLS files offline:

DEX User Guide (9934.)

This document contains a description of the Deferred Execution System (DEX) which may be used to prepare DNLS files offline for subsequent online editing.

CONTENTS

SYNTAX CONVENTIONS

The following conventions are used in the syntax expressions throughout this guide.

NAMED CHARACTERS

Special characters such as Command Accept, Command Delete. Carriage Return, etc. are referred to by names (CA, CD, CR, etc.) in uppercase letters.

Commands are shown in lower case. Most DNLS commands require that only the first character of each command word be typed.

COMMANDS

Commands are shown in lower case. Most DNLS commands require that only the first character of each command word be typed.

PARAMETERS

Values to be supplied by the user are in shown uppercase. The names of these parameters will not cause confusion with the uppercase named characters.

SYSTEM OUTPUT

Text output by the system as a command is entered is shown in lower case letters enclosed in square brackets ([]). Brackets are also used to clarify the command, e.g. the command Insert Statement requires only that the user types "is". However, this is shown as "i[nsert]s[tatement]" in the syntax representation for this command, even though over the Network, some sites do not receive these characters.

QUANTITY

In cases where any number of entities might be supplied by the user, the entity is preceded by the dollar sign character ($).

OPTIONS

Many DNLS commands operate on a variety of entities. These choices are shown in a vertical column. The general syntax of the command applies to all choices except where specified elsewise.

CA

CA means "command accept;" this is done by pressing either CA key on the keyboard, or the right-hand button on the mouse.

LIT

"LIT" means any string of characters input from the keyboard or keyset.

VIEWPSEC

The term VIEWSPEC in a syntax equation means that VIEWSPECS may be set. Viewspecs are explained in Section 5 (see -- 10708,) of this document.

BUG

BUG means the selection of an entity (statement, word, etc.) on the display.

Section 1. DNLS ENVIRONMENT 1

THE CONSOLE AND ITS DEVICES 1a

The DNLS console is essentially a set of devices mounted in or on one or more pieces of furniture. There are several styles of consoles involving different types of furniture, but the component devices are always the same: the display, mouse, keyboard, and keyset. 1a1

THE DISPLAY 1a1a

When DNLS is not running, the display simulates a Teletype -- whenever a Teletype would issue a carriage return and type a new line, the text on the display is moved upwards one line and the new line of text appears at the bottom. 1a1a1

When DNLS is running, the screen is specially formatted. The elements of the format are described here very briefly -- more elaborate descriptions are to be found in appropriate sections of this document. 1a1a2

Feedback Area 1a1a3

The feedback area occupies the top two or three inches of the screen. It is divided up into five areas, each of which contains a specific type of feedback information that tells the user what is going on and the state of his operations. 1a1a3a

VIEWSPEC Feedback Area 1a1a3b

In the upper left hand corner are two lines that indicate the current status of certain parameters called VIEWSPECs, which govern the way in which text is displayed in the text area of the screen. Most of the time this information is displayed in small characters, but during certain commands the characters are displayed in a larger size, which is a signal to the user that VIEWSPEC parameters may now be changed by entering code letters. 1a1a3b1

Command Feedback Line (CFL) 1a1a3c

At the center of the feedback area is the command feedback line or CFL. This displays the name of the current command, such as "Insert Word." If there is no current command, the words "Command Reset" are displayed. Whenever the name of a command is in the CFL, that command is either in progress or "ready to go." 1a1a3c1

Under the CFL an up-arrow may be displayed at certain times and various positions. In principle, the meaning of this arrow is as follows: 1a1a3c2

When the arrow appears under a word of the command name, it means that the word will be "set" by any character that the user enters. 1a1a3c2a

For example, if the CFL reads "Delete Word" with the arrow under "word," the user may enter a "w" to confirm the command "Delete Word," or he may enter some other character to get a different command, e.g., he may enter a "c" to get "Delete Character". 1a1a3c2a1

On the other hand, if the CFL reads "Insert Character" with the arrow under "Insert," then any character the user enters will change "Insert" to something else and advance the arrow to the word "Character." If the user now enters an "r" the CFL will change to "Replace Character" with the arrow under "Character." 1a1a3c2a2

Additionally, when the arrow is under the first word of the command name, it generally means that the user may either enter a character to change the word, or he may go ahead and execute the command. 1a1a3c2b

When the arrow is under the second (or third) word of the command, the command has not been completely specified and the user MUST enter a character to either set a new word or confirm the one that is there. 1a1a3c2b1

As a convenience, when the user wishes to confirm a second or third word that is already there, he may use the special CA (command accept) character. 1a1a3c2b1a

Unfortunately, there are several inconsistencies and ambiguities in this scheme, which the new user will discover as he goes along. 1a1a3c3

Address Area 1a1a3d

This area, which is usually blank, occupies the space just to the right of the CFL. This area is used generally to display file or statement names or numbers to the user during the execution of various comands. 1a1a3d1

Date/Time Area 1a1a3e

This area is at the upper right-hand corner of the screen, and displays the current date and time. It is updated only when the display is recreated by some command (as this happens frequently during DNLS use, the time displayed is generally quite accurate). 1a1a3e1

Display Area 1a1a4

The remainder of the screen is the Display area, used mainly for displaying the user's working text -- i.e., part of the contents of some set of files, formatted according to VIEWSPECs. 1a1a4a

a. Literal-Input Feedback 1a1a4b

When the user is typing in new text, the top of the text area is cleared as needed and the new text appears there as it is being typed in. When the string of new text is completed, the display is recreated with the new text in place in the file as indicated by the user 1a1a4b1

b. The Cursor 1a1a4c

The cursor is used during the execution of commands for selecting operands from the text by pointing to them followed by entering the special character CA (command accept) from the mouse or the keyboard. 1a1a4c1

Whenever such a selection is permitted, the cursor appears as an uparrow (this condition is referred to as "armed cursor"). 1a1a4c1a

When a cursor selection is not permited, the cursor appears as a plus sign (this condition is referred to as "disarmed cursor"). 1a1a4c1b

THE KEYBOARD 1a1b

The keyboard closely resembles a conventional typewriter keyboard. It has upper- and lowercase characters. 1a1b1

The keyboard has the usual complement of characters. plus the following special characters (none of which can be used in text, as they all have special effects as soon as they are typed). 1a1b2

CA (Command Accept) 1a1b3

This character is used in many places in DNLS commands. In general, it causes DNLS to accept something specified or to do somethng that has been requested. It may be thought of as an "affirmative" or a "confirmation." 1a1b3a

It is basically used to terminate literal typein, select an operand from the screen, or give final confirmation for a command. 1a1b3a1

CD (Command Delete) 1a1b4

This special character is used to abort a command. 1a1b4a

CENTERDOT 1a1b5

This is used to repeat certain commands (such as the "Insert Statement" command) without having to respecify all of the parameters, 1a1b5a

BC (Backspace Character) 1a1b6

Each time this key or the Control A key is pressed one character is deleted from current input. 1a1b6a

BW (Backspace Word) 1a1b7

Each time this key or the Control A key is pressed one word (i.e. one visible) is deleted from current input. 1a1b7a

THE KEYSET 1a1c

The keyset has one key for each finger of the left hand. The keys are struck in combinations called "chords," and each chord corresponds to a character or combination of characters from the keyboard. There are 31 possible chords; beyond this, two of the buttons on the mouse may be used to control the "case" of the keyset, giving alternative meanings to each chord. There are four cases used, for a total of 124 possible combinations. 1a1c1

A simple binary code is used, and has proved remarkably easy to learn. Two or three hours' practice are usually sufficient to learn the most commonly used chords and develop reasonable speed. 1a1c1a

The keyset was developed to increase the user's speed and smoothness in operating DNLS. It was found that users normally keep the right hand on the mouse, because the great majority of command operations involve a pointing action; efficient use of the keyboard, however, requires the use of both hands, and shifting the right hand (and the user's attention) to the keyboard is distracting and annoying if it must be done for each two- or three-letter command mnemonic. 1a1c1b

Use of the keyset permits the user to keep his right hand on the mouse and his left on the keyset, reverting to the keyboard only for entry of long strings of text (typically five or more characters of text). 1a1c1b1

Originally, the keyset exactly duplicated the keyboard in function; in the development of DNLS, however, certain control functions have been made two-stroke operations from the keyset where they would be three- or four-stroke operations from the keyboard. Nevertheless, it is still possible to operate all of the features of DNLS without using the keyset; thus the beginner may defer learning the keyset code until he has gained some degree of mastery over the rest of the system. 1a1c1c

THE MOUSE 1a1d

The mouse is a rounded box-shaped object, about four inches on its longest side, which is moved by the right hand. It is mounted on two wheels and a pivot point, and rolls on any flat surface. The wheels drive potentiometers which are read by an A/D converter, and the system causes a tracking spot (or cursor) to move on the screen in correspondence to the motion of the mouse. 1a1d1

The user specifies locations in the displayed text by pointing with the mouse/cursor combination. This eliminates the need for specifying a location by entering a code of some kind. Use of the mouse is very easily learned and soon becomes unconscious. 1a1d1a

On top of the mouse are three special control buttons, whose uses are described below. 1a1d1b

The three buttons on the mouse are used as follows. 1a1d2

1. Right-hand Button 1a1d3

When pushed and released without any intervening input, this button gives a CA (command accept). 1a1d3a

2. Center Button 1a1d4

When pushed and released without any intervening input, this button gives a CD (command delete). 1a1d4a

When it is held down while a string of characters is entered from the keyset, this button causes the characters to be interpreted uppercase -- see the latter part of this section. 1a1d4b

3. Left-Hand Button 1ald5

When pushed and released without any interveneing input, this button gives a backspace, causing the last input character (in a literal type-in) to be thrown away. 1ald5a

A backspace made during the process of a cursor selection causes the last selection made to be cancelled. 1ald5a1

When it is held down while a string of characters is entered from keyset, This button causes the characters to be interpreted as Case 2 input (i.e., letters come out as numbers or punctuation marks). 1ald5b

4. Left-hand and Center Buttons Together 1ald6

When pushed and released without any intervening input, this combination gives a backspace-word, causing the last input word (in a literal type-in) to be thrown away. 1ald6a

When it is held down while a LIT is entered from keyset, this combinations causes the LIT to be interpreted as CASE 3 input (i.e., letters are interpreted as VIEWSPEC control codes. See Section 5 -- 10708,). 1ald6b

MOUSE AND KEYSET, CODES AND CASES 1b

Mouse					
Buttons:	000	010	100	110	
Case:	-0-	-1-	-2-	-3-	1b1
Keyset Code					1b2
O O O O X	a	A	!	show one level less	1b3
O O O X O	b	B	"	show one level deeper	1b4
O O O X X	c	C	#	show all levels	1b5
O O X O O	d	D	$	show top level only	1b6
O O X O X	e	E	%	current statement level	1b7
O O X X O	f	F	&	recreate display	1b8
O O X X X	g	G	'	branch show only	1b9
O X O O O	h	H	(	g off	1b10
O X O O X	i	I	)	show content passed	1b11
O X O X O	j	J	@	i or k off	1b12
O X O X X	k	K	+	show content failed	1b13
O X X O O	l	l	-	show plex only	1b14
O X X O X	m	M	*	show statemnt numbers	1b15
O X X X O	n	N	/	hide statemnt numbers	1b16
O X X X X	o	o	↑	frozen statement windows	1b17
X O O O O	p	p	0	frozen statement off	1b18
X O O O X	q	q	1	show one line more	1b19
X O O X O	r	R	2	show one line less	1b20
X O O X X	s	S	3	show all lines	1b21
X O X O O	t	T	4	first lines only	1b22
X O X O X	u	U	5	inhibit refresh display	1b23
X O X X O	v	V	6	normal refresh display	1b24
X O X X X	w	W	7	all lines, all levels	1b25
X X O O O	x	X	8	one line, one level	1b26
X X O O X	y	Y	9	blank lines on	1b27
X X O X O	z	Z	=	blank lines off	1b28
X X O X O	,	<	[	(nothing)	1b29
X X O X O	.	>	]	(nothing)	1b30
X X O X O	;	:	←	(nothing)	1b31
X X O X O	?	\	ALT	centerdot	1b32
X X O X O	SP	TAB	CR	(nothing)	1b33

Section 2. FILES 1

FILE STRUCTURE 1a

INTRODUCTION 1a1

When working in DNLS, one is at all times constructing, studying, or modifying a file. DNLS files have a hierarchical, tree, or outline structure. 1a1a

```
---------------------------------------------------------    1a1b
   0 ...                                                     1a1b1
   1 ...
      1a ...
      1b ...
         1b1 ...
         1b2 ...
         1b3 ...                                             1a1b2
   2 ...                                                     1a1b3
   3 ...
      3a ...
      3b ...
      3c ...
         3c1 ...
      3d ...
         3d1 ...
         3d2 ...
            3d2a ...
            3d2b ...
            3d2c ...                                         1a1b4
   4 ...
      4a ...
      4b ...                                                 1a1b5
   5 ...
      5a ...
         5a1 ...
         5a2 ...
            5a2a ...
      5b ...                                                 1a1b6
                                                             1a1c
    FIGURE 1.  Hierarchical File Structure
---------------------------------------------------------    1a1d
```

It would be difficult to overstate the importance of this structure in the design of DNLS; it is correspondingly important for the user to understand the structure and its terminology. 1a1e

In the remainder of this discussion of file structure. note that every statement is headed by a string of digits and letters. These strings are the statement numbers associated with the file structure; they have been suppressed from the rest of the document, but are printed here as an example. Also, the reader is invited to observe the way this document is formatted; the indentation of statements reflects "level" in the structure. 1a1f

1a2 OVERALL FILE STRUCTURE 1a2

1a2a Every DNLS file is made up of STATEMENTS, entities which may contain any sort of text (every paragraph and heading in this document is a statement). 1a2a

1a2a1 Every DNLS file has an ORIGIN STATEMENT or "zero statement". (The origin statement has been omitted from the printout of this document). The origin statement is a "0th-level" statement (the only one in the file). 1a2a1

1a2a2 The statements immediately below the origin statement in the outline are "1st-level" statements (all section titles in this document are the 1st-level statements). 1a2a2

1a2a3 The statements immediately below the 1st-level statements are 2nd-level statements, and so forth to arbitrary depth. 1a2a3

1a3 STATEMENT NUMBERS 1a3

1a3a Every statement has a unique "statement number." This is a string of alternating fields of numbers and letters. The statement number is a primary means of addressing parts of the file in DNLS commands. 1a3a

1a3a1 The first field always contains a number. 1a3a1

1a3a2 The number of fields is equal to the level of the statement. Properly speaking, the origin statement should have no statement number, since its level is 0; for convenience, however, the statement number "0" is assigned to it. 1a3a2

1a3a3 The statement number (and its following space) is NOT part of the text of the statement; it is associated with the position of the statement in the file and is subject to change when the file structure is modified by adding, deleting, or moving statements. 1a3a3

1a3b When necessary, the @ character is used in the letter fields of statement numbers as an "alphabetical zero." Thus the 26 letters and the @ can be used to form a sequence: a, b, c, ... x, y, z, a@, aa, ab, ac, ... az, b@, ba, bb, ... 1a3b

1a4 PRIMARY RELATIONSHIPS BETWEEN STATEMENTS 1a4

1a4a The following relationships between statements are defined: SUBSTATEMENT, SOURCE, SUCCESSOR, AND PREDECESSOR. These are best defined by examples, with reference to Figure 1 on the first page of this section. 1a4a

1a4a1 SUBSTATEMENT and SOURCE refer to the relationships between statements at different levels. 1a4a1

1a4a1a Statements 1, 2, and 3 are substatements of the origin statement. Statement 1a is a substatement of Statement 1. Statements 1b1, 1b2, and 1b3 are substatements of Statement 1b. 1a4a1a

1a4a1a1 Any statement may have any number of substatements. 1a4a1a1

1a4a1a2 All first level statements are substatements of the origin statement. 1a4a1a2

1a4a1a3 Given the number of a statement, the number of a substatement is obtained by adding a field to the end of the last number. 1a4a1a3

1a4a1b SOURCE is the inverse of substatement. Statement 1b is the source of Statements 1b1, 1b2, and 1b3. Statement 3c is the source of Statement 3c1. 1a4a1b

1a4a1b1 Every statement has just one source (except the origin statement, which has no source). 1a4a1b1

1a4a1b2 Given the number of a statement, the number of the source is obtained by removing a field from the end of the first number. 1a4a1b2

1a4a2 SUCCESSOR and PREDECESSOR refer to the relationships between statements of the same level. 1a4a2

1a4a2a Statement 2 is the SUCCESSOR of Statement 1. Statement 3d2 is the successor of Statement 3d1. 1a4a2a

1a4a2a1 Not every statement has a successor. The origin statement has no successor. No statement has more than one successor. A statement and its successor always have the same level and the same source. A successor specification with a statement having no succeeding statement of the same level and source refers to the statement itself. 1a4a2a1

1a4a2a2 Given the number of a statement, the number of the successor is obtained by incrementing the last field of the first number. 1a4a2a2

1a4a2b PREDECESSOR is the inverse of successor. Statement 1a is the predecessor of Statement 1b. 1a4a2b

1a4a2b1 Not every statement has a predecessor. The origin statement has no predecessor. No statement has more than one predecessor. A statement and its predecessor always have the same level and the same source. A predecessor specification with a statement having no preceding statement of the same level and source refers to the statement itself. 1a4a2b1

1a4a2b2 Given the number of a statement, the number of the predecessor is obtained by decrementing the last field of the first number. 1a4a2b2

1a5 STRUCTURAL ENTITIES MADE UP OF STATEMENTS 1a5

1a5a Given these primary relationships -- source, substatement, predecessor, and successor -- we can define the following STRUCTURAL ENTITIES: STATEMENT, BRANCH, PLEX, and GROUP. 1a5a

1a5a1 STATEMENT has already been explained. 1a5a1

1a5a2 A BRANCH consists of a specified statement, plus all its substatements, all their substatements. etc. In the illustration, Branch 1 consists of Statements 1, 1a, 1b, 1b1, 1b2, and 1b3. Branch 1a consists of Statement 1a alone. Branch 4 consists of Statements 4, 4a, and 4b. 1a5a2

1a5a2a Branch 0, in any file, contains the entire file. 1a5a2a

1a5a3 A PLEX is made up of a specified branch, plus all the other branches that have the same source. Plex 1a and Plex 1b are the same; each consists of Branches 1a and 1b. Plex 3a consists of Branches 3a, 3b, 3c, and 3d; Plex 3b and 3c, and 3d are the same as Plex 3a. 1a5a3

1a5a4 A GROUP is a contiguous subset of a plex. It is identified by two branches, which must be in the same plex, and consists of those two branches plus all branches lying "between" them in the same plex. Group 3d2c, 3d2c consists of Branches 3d2a, 3d2b, and 3d2c.

1a5a4

1a6 SECONDARY RELATIONSHIPS BETWEEN STATEMENTS 1a6

1a6a We can now define the following relationships: HEAD, TAIL, END, UP, DOWN, NEXT, and BACK. 1a6a

1a6a1 The HEAD of a specified statement is the first statement at the same level that has the same source. The head of Statement 3d2c is Statement 3d2a. The head of Statement 5a2 is Statement 5a1. The head of Statement 3a is Statement 3a itself. 1a6a1

1a6a1a Head pertains only to members of the same plex. 1a6a1a

1a6a2 The TAIL of a specified statement is the last statement at the same level that has the same source. The tail of Statement 3d2b is Statement 3d2c. The tail of Statement 4a is Statement 4b. The tail of Statement 3c1 is Statement 3c1 itself. 1a6a2

1a6a2a Tail pertains only to members of the same plex. 1a6a2a

1a6a3 The END of a specified statement is the "last" statement in the branch defined by the specified statement. The end of Statement 3 is Statement 3d2c. The end of Statement 3c is Statement 3c1. 1a6a3

1a6a4 UP refers to the statement that is one level higher than the current statement and precedes the current statement. For example, statement 3 is up from statement 3c. 1a6a4

1a6a5 DOWN refers to the statement following the current statement that is one level lower. For example, statement 4a is down from statement 4. 1a6a5

1a6a5a Any down specification with a statement having no following statement at a lower level refers to the statement itself. Thus, excess d specifications are ignored. 1a6a5a

1a6a6 NEXT refers to the statement immediately following the current statment regardless of level or of source. For example, statement 4b is next to statement 4a; statement 5 is next to statement 4b. 1a6a6

1a6a7 BACK refers to the statement immediately preceding the current statement regardless of level and source. For example, 4b is back from statement 5. 1a6a7

FILE CONTENT

1b

FILE NAMES 1b1

The names of files in TENEX/DNLS are of the following form: 1b1a

<DIRECTORY>FILENAME.EXTENSION;VERSION # 1b1a1

where 1b1b

DIRECTORY = 1-39 alphanumeric characters, 1b1b1

excluding control characters, non-printing characters, period (.), and semicolon (;). This element is a TENEX user name and is required only when a user references a file belonging to a directory other than his own (or the one to which he is currently connected). 1b1b1a

FILENAME = 1-39 alphanumeric characters, 1b1b2

excluding control characters, non-printing characters, period (.), and semicolon (;) 1b1b2a

EXTENSION = 1-39 alphanumeric characters, 1b1b3

excluding characters control, non-printing characters, period (.), and semicolon (;) 1b1b3a

VERSION # = a numeric value (1 to 131071) 1b1b4

The length of the entire filename (including the delimiters . and ;) must not exceed 39 characters. Otherwise, there are no restrictions on the length of any field within the total filename. . 1b1c

TYPES OF FILES 1b2

There is a variety of types of files that are generated within DNLS. When a user enters DNLS for the first time, he is automatically assigned a file by DNLS. The file is empty except for a dummy origin statement which contains his identification string as a filename, an extension name "DNLS" and version number 1; this file is referred to an the user's "initial file". Within DNLS itself, files are created by using the Output File and Output Device commands, see File commands described in the latter part of this section. 1b2a

At this point it is necessary to identify the types of files used by the DNLS user. Although the user may use any identifier as an extension name, the convention generally followed by the DNLS user group is to identify the type of the file by the extension name where: 1b2b

DNLS = an DNLS file 1b2b1

PC = a partial copy file created by DNLS when the file is edited in any way 1b2b2

(NO.)= a sequential file for hardcopy output where NO. is the number of copies generated when the file is printed 1b2b3

One of these extension names is automatically supplied by the system whenever the user fails to specify extension name in a command, depending on the operation being performed. 1b2c

DNLS FILES 1b2d

An DNLS file is a file which may be edited or viewed in DNLS. DNLS files are created within DNLS in two ways: when the user enters DNLS for the first time. a file bearing the users identification string as its filename is created by the system; and when the user issues the Output File command and specifies a new file. 1b2d1

PARTIAL COPY FILES 1b2e

Whenever an DNLS file is modified a partial copy file is automatically created by the system for that file. Partial copy files have an extension name "PC" and may be used only in conjunction with an DNLS file. That is, the user may not load, copy, etc. a partial copy file. 1b2e1

When a user attempts to modify an DNLS file, he is actually working on the partial copy associated with that file. Modifications are actually made to an DNLS file only by operations which merge to it the contents of its partial copy. 1b2e2

When a partial copy exists for a particular file, the file is considered "locked", i.e. no other partial copy may be made for the file. This feature prevents other users from modifying the file. A file remains locked until the user updates, outputs, or unlocks the file via the commands described in the latter part of this section. 1b2e3

SEQUENTIAL ACCESS FILES 1b2f

The hardcopy devices used by the system require sequential files, i.e., files that are processed as a sequence of characters. Any file that is to be output at a terminal requires processing by the Output Device command which essentially takes a DNLS file and copies it into a sequential file for processing on a specific device. If the user, when issuing the Output Device command allows the system to 'create' an extension name for the sequential file, the extension name will be some number depending on the number of copies of the file desired by the user. 1b2f1

SYSTEM CREATION OF FILES 1b3

The TENEX system automatically creates files for the user under a variety of circumstances. 1b3a

NEW FILENAME 1b3a1

When the user enters the DNLS system for the first time DNLS automatically creates a file for him with the name "user's identification string.DNLS;1". 1b3a1a

When the user makes changes to a file in the DNLS subsystem, the system automatically creates a partial copy file for the opened file. This file contains the changes made to the original file. With the DNLS command Update File, the user can cause the system to add the changes back into the original and delete the partial copy. The system lists partial copies in the user's file directory as separate files with a new file name that it creates in the form (USERNAME)FILENAME.PC;#. 1b3a1b

NEW EXTENSION NAMES 1b3a2

If when the user issues the Output File command in DNLS, he enters a unique (to his directory) FILENAME followed by a CA. The system will automatically assign the file the extension name "DNLS". Similarly, when the user issues the Output Device command, the system automatically assigns the file the extension name "TXT". 1b3a2a

NEW VERSION NUMBERS 1b3a3

If, when the user outputs a file from DNLS, ne enters a FILENAME that exists in his directory, the system will automatically assign the file the next higher version number. 1b3a3a

USER CREATION OF FILES 1b4

The user may create a new DNLS file by using the Update or Output command; text files are creted by using the Output Device command. These commands are described in the next part of this section. 1b4a

INFORMATION IN THE ORIGIN STATEMENT OF A FILE 1b5

The origin statement of a named file begins with the filename, the date and time of the last modification to the file (or date of creation if it is unmodified), and the identification string of the user who modified or created it (ending with a semicolon). As explained below, this information is automatically maintained by the system. 1b5a

Example:
<SMITH>FILE.DNLS;22, 24-MAY-71 11:50 SSS ;7, 19-14:48
SSS 1b5a1

FILE MANIPULATION AND INPUT/OUTPUT COMMANDS 1c

LOAD FILE 1c1

The load file command causes the file specified to be opened and made available to the user for work in the DNLS subsystem. 1c1a

l[oad] f[ile] FILENAME CA 1c1a1

Where FILENAME = the name of the file to be opened. 1c1a2

If the user enters only the name field of FILENAME, extension DNLS and the highest version number, are the default values for the remaining fields. If the file belongs to another user's directory, FILENAME must include the directory name enclosed in anglebrackets. 1c1a2a

When this command is executed, any file and any associated partial copy currently open is automatically closed before the the file specified in the load file command is opened. 1c1b

If the file being loaded has an associated partial copy, the partial copy is also opened. 1c1b1

The user may open a file from another user's directory by prefacing FILENAME with <other user's name>. However, if the file has an associated partial copy created by the other user, the file will be "locked" to further changes by anyone but the other user (the file may be read only). In this case, the user may either request the other user to unlock the file, or he may copy the file (in EXEC) so that he has a copy in his own directory. However, when the file is copied in EXEC, the partial copy that causes the file to be locked is not also copied. 1c1b2

The file being opened must be an DNLS file. 1c1b3

The user may also access files by using links, 1c1b4

Example: 1c1c

l f myfile CA 1c1c1

causes the system to open the most recent version of the file myfile.nls in the current user's directory. 1c1c1a

```
1 f <smith>rate.nls;3                                   1c1c2
```

causes the system to open a file named "rate.nls;3" belonging to the directory SMITH. 1c1c2a

UPDATE FILE 1c2

The update file command causes the system to merge the contents of the current DNLS file with its current partial copy. The file created by this merge can either be written onto a new version of the same file, or written over the old version of the file. 1c2a

```
u[pdate file  new version  FILENAME] CA
              o[ld version FILENAME] CA                 1c2a1
```

Note: in general, updating to a new version is "safer" than updating to an old version. In the event of a system crash during an update to an old version, that version may be "lost" (along with its partial copy). If a crash should occur during an update to a new version. the original version and partial copy are not affected even though the new version may be lost. 1c2b

When updating to an old or new version, the current partial copy is automatically deleted (but not expunged) by the system. 1c2c

Instead of incorporating the partial copy into the current file, the user may delete all changes made to the file since the last update or output operation by using the Execute Unlock command which deletes the current partial copy. 1c2d

Example: If the current file is APPLE;DNLS.4 1c2e

```
u o CA                                                  1c2e1
```

causes the current file to remain APPLE.DNLS;4 1c2e1a

```
u  CA                                                   1c2e2
```

causes the current file to be changed to APPLE.DNLS;5 1c2e2a

OUTPUT FILE 1c3

The Output File command causes the system to copy the content of the currently open file and its associated partial copy to the filename specified. 1c3a

o[utput] f[ile] FILENAME CA 1c3a1

Where FILENAME = the name of the file to be created. 1c3b

If only the name field of FILENAME is supplied, the system creates a file having the extension name "DNLS" and assigns it the next highest version number. 1c3b1

The origin statement of the destination file will contain FILENAME, the current date and time, and the identification string of the user who is creating the file. 1c3c

The contents of the currently open file and its partial copy are then copied into the named file. Finally, the named file is opened and the currently open file is closeadand its partial copy is automatically deleted (but not expunged) by the system. Thus the Output File command always leaves you with the named file open. 1c3d

The difference between output File and Update File is that the file being created by Output File is ordered internally to provide more efficient access and storage. 1c3d1

An attempt to perform an output operation using the same filename and version nuaber as the current file will cause the system to issue the message: 1c3d2

FILE BUSY 1c3d2a

and the command will not be executed. 1c3d3

When this command is executed, any partial copy associated with the file being output is deleted (but not expunged). 1c3d4

Example: if there is a file APPLE.DNLS;4 1c3d5 o f apple CA   creates a file APPLE.DNLS;5 1c3d5a

OUTPUT/UPDATE LOCKED FILE 1c4

When an Output or Update File is done on a locked file, the user must have write privileges for the directory to which the original file belongs (even if the user is putting the new file in another directory). If the user doesn't have write privileges, the message "No write access to <DIRECTORY>" is issued. The Output/Update is not executed. 1c4a

EXECUTE UNLOCK 1c5

The Execute Unlock command deletes the contents of the partial copy associated with the current file. In effect the file is restored to its status immediately following the last update or output operation on the file. 1c5a

e[xecute] u[nlock] CA [FILENAME really ?] CA 1c5a1

Where FILENAME = the name of the current file 1c5b

An extra CA is required to terminate this command to decrease the chance of executing this command by mistake. 1c5c

If the user attempts an Execute Unlock command on a file that is not locked, the system will issue the message: "This file is not locked". 1c5d

If the file is locked by someone else, system will issue message "You do not have this file locked". 1c5e

If the user does not have write privileges for the directory in which the specified file resides, the system will issue the message: "No write access to <DIRECTORY>". 1c5f

EXECUTE RESET 1c6

The execute reset command creates a partial copy that voids the contents of the current file. 1c6a

e[xecute] r[eset] CA [really ?] CA 1c6a1

This command is essentially equivalent to deleting plex 1 of a file. 1c6b

Like the Execute Unlock Command, this command requires an extra terminating CA to decrease the chance of executing this command by mistake. (Should this command be executed by mistake, the Execute Unlock command may be used to restore the original file, but not the partial copy.) 1c6c

EXECUTE FILE VERIFY 1c7

The execute file verify command causes the system to check for any problems in the current file that would render it unacceptable for processing by DNLS (e.g. structural inconsistancy). 1c7a

```
e/xecute/ f/ile verify/  CA
```
1c7a1

In response, the system will print: 1c7b

```
FILE VERIFY IN PROGRESS
```
1c7b1

If no errors are detected, the above message will go away. Otherwise, it issues the message: 1c7c

```
BAD FILE -- TYPE CA
```
1c7c1

This message indicates that the system found an error in the file structure. To recover the file use the following procedure: 1c7d

1. Issue the Execute Quit command, enter NLS, and attempt to load the file. 1c7d1

2. Execute File Verify. If still bad continue to next step. 1c7d2

3. Check partial copy file. Issue the Execute Unlock command to delete the current partial copy of the file. 1c7d3

4. Execute File Verify. If still bad continue to next step. 1c7d4

5. If at this point the error message persists for the file, the only recourse is to return to an earlier version of the file. Go to EXEC, delete the current version, reenter NLS and load a previous version of the file. 1c7d5

NULL FILE 1c8

A new command, Null File, has been added to TNLS and DNLS. It requires a file name, and will create an empty file of that name. Upon completion of the command the user is left with the CM / display start at the origin of this new file. 1c8a

```
n/ull file/ FILENAME CA
```
1c8a1

If a file with the specified name already exists, then the message "File already exists; CA to proceed" is typed. Confirmation (a CA) causes DNLS to create a new, empty version of the file. Any other character is interpreted as a new command. 1c8b

EXECUTE OWNERSHIP OF FILE 1c9

The Execute Ownership of File command enables the user to change the default directory asssociated with all link specifications in a file. 1c9a

```
e/xecute/ o/wnership  of file/ DIRECTORY NAME CA
```
1c9a1

EXECUTE FILE STATUS 1c10

The Execute Status File command causes the system to display status information about the current file. 1c10a

```
e/xecute/ st/atus/ f/ile/ CA
```
1c10a1

When this command is executed the following information is displayed in the upper left portion of the screen: 1c10b

- the filename 1c10b1
- whether the file is locked or not 1c10b2
- the default directory for links 1c10b3
- number of statements in the file 1c10b4
- the creation date of the first version of the file 1c10b5

- the creation date of the current version of the file 1c10b6
- number of structure pages 1c10b7
- number of data pages 1c10b8
- number of total pages 1c10b9
- percentage of words used 1c10b10

EXECUTE LINK STACK STATUS 1c11

This command causes the system to display the current link stack in the upper left portion of the screen. 1c11a

```
e[xecute] st[atus] l[ink stack] CA
                                                 1c11a1
```

OUTPUT DEVICE PRINTER FILE 1c12

The Output Device Printer File command causes the system to convert the current file from its random file format to a sequential format and to process it so that it may be listed at the line printer. 1c12a

The default procedure for the execution of this command causes the system to output the current file to a file of the same filename in the directory <PRINTER>. The system then asks the user how many copies are to be generated. This number becomes the extension field of the sequential file name. This procedure eliminates the need (if appropriate) for the user to copy the file to the line printer in the EXEC for each hardcopy required of the file. Alternatively, the user may refuse the default filename and subsequent automatic listing by typing in a filename whose directory is his own, another user's, i.e. any directory but <PRINTER>, This causes the system to create a sequential file in the specified directory which may be subsequently listed by copying it to "LPT:" or some file name in the directory <PRINTER" at the EXEC level. This procedure also requires the user to specify number of copies of the file. 1c12b

```
o[utput] d[evice] p[rinter file FILENAME] CA...
                                 FILENAME CA
...[copies?] NUMBER CA
[output processor in progress]                   1c12b1
```

When this command is executed, the current DNLS file and its partial copy are printed at the terminal. 1c12c

This processor may be interrupted at any time by issuing the interrupt Control O. 1c12d

The file is printed beginning with the statement currently at the top of the display area. To print an entire file, the file must be displayed starting at statement 0. 1c12e

The user may control the format of the output from within the file by using the directives described in the Output Processor Guide (7477,). Output format may also be controlled by setting the viewspecs discussed in Section 5 (see -- 10708,) of this document prior to issuing the Output Device command. 1c12f

Section 3. ADDRESSING IN DNLS - JUMPING AND LINKS 1

JUMPING 1a

DNLS files may, of course, contain a great deal more text than can be displayed on the screen, just as a document may contain more than one page of text. An DNLS file is thought of as a long "scroll." The process of moving from one point in the scroll to another, which corresponds to turning pages in hard copy, is called "jumping." There is a very large family of Jump commnds. 1a1

The basic Jump command is Jump to Item. The user specifies it by entering 'j or "ji", and then either selects some statement with the cursor (using the mouse) or types in SPACE followed by the name or number of a statement. The selected statement is moved to the top of the screen. 1a1a

Most of the Jump commands reference the hierarchical structure of the text. Thus Jump to Successor brings to the top of the display the next statement at the same level as the selected statement; Jump to Predecessor does the reverse; Jump to Up starts the display with the statement of which the selected statement is a substatement, and so forth. 1a1b

The Jump to Name command uses a different way of addressing statements. If the first word of any statement is enclosed in parentheses (this is the system default -- the user can change the delimiter characters), the system will recognize it as the "name" of the statement. Then, if this word appears somewhere else in the text, the user may jump to the named statement by pointing to the occurrence of the name, or by typing the name. 1a1c

This provides a cross-referencing capability which is very smooth and flexible; the command Jump to Return will always restore the previous display, so that the user may follow name references without losing his place. 1a1c1

It is also possible to jump to a statement by typing its statement number. 1a1d

JUMP COMMANDS 1b

Jump to Origin 1b1

The display start is positioned to the origin statement. 1b1a

```
j/ump to/ o/rigin/ VIEWSPEC CA                                1b1a1
```

Jump to Item 1b2

The display start is positioned to the selected statement. Note that the i in the command specification may be omitted. 1b2a

```
j/ump to item/ i/tem/  BUG VIEWSPEC CA
                       NULL                                   1b2a1
```

Jump to Up 1b3

The display start is positioned to the source statement of the selected statement 1b3a

```
j/ump to/ u/p/  BUG VIEWSPEC CA                               1b3a1
```

Jump to Down 1b4

The display start is positioned to the first substatement of the selected statement 1b4a

```
j/ump to/ d/own/ BUG VIEWSPEC CA                              1b4a1
```

Jump to Successor 1b5

The display start is positioned to the successor of the selected statement 1b5a

```
j/ump to/ s/uccessor/ BUG VIEWSPEC CA                         1b5a1
```

Jump to Predecessor 1b6

The display start is positioned to the predecessor of the selected statement. 1b6a

```
j/ump to/ p/redecessor/ BUG VIEWSPEC CA                       1b6a1
```

Jump to Head 1b7

The display start is positioned to the first statement in the plex where the selected statement is found. 1b7a

```
j[ump to] h[ead] BUG VIEWSPEC CA                    1b7a1
```

Jump to Tail 1b8

The display start is positioned to the last statement in the plex where the selected statement is found. 1b8a

```
j[ump to] t[ail] BUG VIEWSPEC CA                    1b8a1
```

Jump to End of Item 1b9

The selected statement determines a branch, and the last statement in that branch is placed at the top of the display. 1b9a

```
j[ump e[nd of] i[tem] BUG VIEWSPEC CA
                NULL                                1b9a1
```

LINKS 1c

A "link" is a string of text, occurring in an ordinary file statement, which indicates a cross-reference of some kind. It may refer to another statement in the file, or to a statement in some other file, possibly belonging to another DNLS user. Using links is similar to the Load File command except that it is quicker and allows the user to reference any location in the file. Using links also enables the user to embed precise cross-references in a file for subsequent on-line reading. 1c1

The text of the link is both human-readable and machine-readable, and the command Jump to Link permits the user to point to the link with the mouse and immediatly see the material referenced. 1c2

In general, the syntax of the link is: 1c3

(directory,filename,address:viewspec) 1c3a

directory = 1c3b

the directory associated with the filename. If not specified, the current user's directory is assumed unless the Declare Default Directory command (see Section 2. -- 10705,) was used to specify another directory. 1c3b1

filename = 1c3c

the name of the file to be accessed (i.e., the name field only). If filename is omitted, the system assumes that the link refers to a location in the current file. 1c3c1

address = 1c3d

a statement number or name indicating the exact location in the file which appear as the first statement on the display. If address is not specified, the system assumes the origin statement of the file. 1c3d1 viewspecs = 1c3e

a series of view specifications, or format codes which control the way the file will appear when accessed through the link. If not specified, the system uses the viewspecs i effec when the link is executed. Viewspecs are discussed later in this document (see Section 5 -- 10708,). 1c3e1

Links are usually delimited by right parentheses. However, they may also be delimited by angle brackets ("<"and ">") or preceded by two dashes ("--"). Also, right and left delimiters may be used in any combination. e.g. a link may begin with the chracters "--" and end with a left parethesis. 1c4

An example of a link is (Smith, Plans, Longrange:ebtng). 1c5

The first item in the link indicates that the refernced file belongs to a user named Smith; the second is the name of the file; the third is the name of a statement in the file (a statement number may also be used); and the string of characters following the colon controls the VIEWSPECs to set up a particular view of the material. 1c5a

The use of interfile links permits the construction of large linked structures made up of many files, and study of these files as if they were all sections of a single document. 1c5b

Other examples include: 1c6

(see -- 7000,) 1c6a

<3> 1c6b

(myfile,:x) 1c6c

RETURN JUMPS 1d

General 1d1

The commands "Jump to Return" and "Jump to File Return" permit the user to return automatically from any jump to a previous view. Thus links may be freely used without the danger of losing one's place. 1d1a

The Intrafile Return Ring 1d2

All jumps made within a file (except jumps made with "Jump to Return" and "Jump to Ahead") are recorded in an ordered list called the Intrafile Return Ring. The ring may have up to five entries, each of which records a display start position and a set of display parameters -- i.e. the information needed for complete reconstruction of a view, assuming that no editing takes place. 1d2a

The list is a ring in the sense that its ends are joined; i.e. the first entry is also the list successor of the last entry. A pointer indicates the "current" entry, i.e., the entry containing information for the current view. Each new jump (except "Return" and "Ahead") causes a new entry to be made ahead of the current entry, and the pointer is moved to the new entry. 1d2b

The command "Jump to Return" causes the pointer to be moved back one entry and the display is recreated from the new "current" entry. No changes are made in the entries themselves. 1d2c

The command "Jump to Ahead" causes the pointer to be moved forward one entry, and the display is recreated from the new "current" entry. No changes are made in the entries themselves. 1d2d

It will be seen that because of the ring structure of the list, repeated use of "Jump to Return" or "Jump to Ahead" will eventually bring the user back to the starting point. 1d2e

The user may "step" through the ring by issuing either the Jump to File Return or the Jump to File Ahead command and entering a Space character instead of the confirming CA when the name of the next file in the ring is displayed on the screen. The user may continue hitting the Space character in response to each filename displayed on the screen until any particular file is found whereupon entering a CA in response to the desired filename will cause the system to execute the return or ahead. 1d2f

It should also be remembered that each new entry in the ring always goes just ahead of the "current" entry, and that an old entry may be overwritten in the process. 1d2g

The "Jump to File Return" and "Jump to File Ahead" Commands 1d2h

These commands are exactly analogous to the corresponding intrafile jump commands. "Jump to File Return" moves the pointer back one entry and creates a new display from the information in the new "current" entry, and "Jump to File Ahead" does the reverse. 1d2h1

Section 4. EDITING AND COMPOSITION

1

COMPOSITION 1a

Composition is simply the creation of new text material as content for a file. 1a1

In the simplest case, the user gives the command "Insert Statement" by typing "is". He then points (using the mouse) to an existing statement; the system displays a new statement number which is the logical successor, at the same level, as the statement pointed to. The user may change the level of this number upward by typing a "u" or downward by typing a "d". The new statement number is changed accordingly by the system. 1a2

The user then types the text of the new statement from the keyboard. On the screen, the top part of the text-display area is cleared and characters are displayed here as they are typed. When the statement is finished, the user hits a CA (command accept) button on the keyboard or mouse, and the system recreates the display with the new statement following the one that was pointed to. 1a3

New material may also be added to existing statements by means of commands such as Insert Word, Insert Text, and others. Properly speaking, these operations are for modification rather than composition, and are discussed below. 1a4

EDITING 1b

A large repertoire of editing commands is provided for file modification. These commands operate upon various kinds of text entities. Within statements, they may operate upon single characters, words, and arbitrary strings of text defined by pointing to the first and last characters, 1b1

This set of commands is not restricted to operation within one statement at a time; for example, a word may be moved or copied from one statement to another. 1b1a

The editing functions also operate at the structural level, taking statements or sets of statements as operands. A number of special entities have been defined for this purpose: for example, a "branch" consists of some specified statement, plus all of its substatements, plus all of their substatements, etc. A branch can be deleted, moved to a new position in the structure, etc. 1b2

COMMANDS 1c

INSERT 1c1

Insert Character 1c1a

LIT is inserted immediately after the selected character 1c1a1

i[nsert] c[haracter] BUG LIT CA 1c1a1a

Insert Word 1c1b

LIT is inserted after the selected word, with an intervening SPACE. 1c1b1

i[nsert] w[ord] BUG LIT CA 1c1b1a

Insert Visible 1c1c

LIT is inserted after the selected visible, with a SPACE between. 1c1c1

i[nsert] v[isible] BUG LIT CA 1c1c1a

Insert Link 1c1d

The link is inserted after the selected visible, with a SPACE between. 1c1d1

i[nsert] l[ink] BUG LIT CA 1c1d1a

LIT and the visible selected by BUG are both required to have the syntax of a valid link (see Section 3 of this document -- 10706). 1c1d2

Insert Number 1c1e

LIT is inserted after the selected visible, with a SPACE between. 1c1e1

i[nsert] n[umber] BUG LIT CA 1c1e1a

Insert Text 1c1f

LIT is inserted after the selected character. This command is identical to the insert character command. 1c1f1

i[nsert] t[ext] BUG LIT CA 1c1f1a

Insert Invisible 1c1g

LIT is inserted immediately after the selected invisible. 1c1g1

```
i/nsert/ i/nvisible/ BUG LIT CA                   1c1g1a
```

Insert Statement 1c1h

LIT becomes the text of a new statement or set of statements, following the selected statement at a level determined by the LEVADJ. 1c1h1

```
i/nsert/ s/tatement/ BUG $LEVADJ SPACE LIT CA
                         NULL    CA        CDOT    1c1h1a
```

LEVADJ = 1c1h2

any number of up or down level specifications (u or d respectively) which indicates that the statement to is be inserted x levels higher or lower than the statement specified by BUG. u and d may also be preceded by an integer value indicating the number of levels up or down. This specification may include both u's d's . which cancel out each other on a one-to- one basis. 1c1h2a

CDOT = 1c1h3

"center dot" character means continue insert command. This option allows the user to continue inserting statements at the same and/or other levels. When this delimiter is used, the syntax for inserting subsequent statements is the same as though the user had entered the Insert command up to and including the first CA; the system expects the user to enter a level specification and/or LIT. 1c1h3a

When a new statement is inserted into a file, all statements following the place of insertion are automatically renumbered by the system as necessary. 1c1h4

The maximum number of characters allowed per statement is approximately 2000. Every statement consists of at least one character. 1c1h5

After this command is executed the CM is positioned to the first character of the most recently inserted statement. 1c1h6

Section 5. VIEW CONTROL OPERATIONS

1

VIEWSPECS 1a

INTRODUCTION 1a1

In DNLS the user is at all times "viewing" a file. Certain parameters are in effect at all times which control the precise nature of the view a user has of a file. These parameters are called viewspecs and several of the DNLS commands documented in this Reference Guide allow their specification as part of the execution of the command. 1a1a

Generally speaking, the most common and important use of viewspecs is to cause some of the statements in the file (or part of the file) to be ignored (not displayed) for various reasons. Thus, for example, certain important viewspecs have the effect of ignoring all statements that are below a specified level in the hierarchical file structure. 1a1b

When the user first enters DNLS, all of the viewspecs are automatically preset to standard values. Whenever the user issues a viewspecs command or certain others as noted in this document, he has the option of changing any of the viewspecs by typing special one-letter codes. 1a1b1

VIEWSPEC CONTROL 1a2

VIEWSPECs may be controlled in four ways; during certain commands such as Jump or Load, with the View Set command, in a link or from the keyset in Case 3. (The viewspecs may also be set from the keyboard with the right-hand and center buttons on the mouse down, i.e. in Case 3 position.) 1a2a

During the Jump and Load commands (and a few others), there is a point where the VIEWSPECs in the upper left-hand corner of the display become large, indicating that all VIEWSPECs are accessible to change. They may then be changed by typing the codes in from the keyboard or keyset as upper- or lower-case letters. 1a2a1

The View Set command may be used to achieve exactly the same effect without doing anything else. 1a2a2

A link may contain a string of VIEWSPEC codes, preceded by a colon, as the last element in the parentheses. 1a2a3

Case 3 may be used to set all of the VIEWSPECs that are not capital letters, as shown in the table of keyset codes. This may be done at any time. 1a2a4

Note that the chord for each VIEWSPEC corresponds to the appropriate lower-case letter in Case 0. 1a2a4a

After VIEWSPECs have been given in this fashion, it is necessary to hit Chord 00110, Case 3 for "new view," (or otherwise cause the display to be recreated), before the new VIEWSPECs will become effective. 1a2a4b

VIEWSPEC DEFINITIONS 1a3

INTRODUCTION 1a3a

There are two types of viewspecs. The first type includes the Level and Line specifications whose value may range from 1 to ALL. 1a3a1

The remaining viewspecs are ON/OFF switches for various DNLS features. Each is controlled by a pair of one-letter codes, one of which turns the feature ON and the other of which turns it OFF. Note that some of these codes are capital letters; it is important to distinguish between capital and lower-case viewspec codes, because they have different effects. 1a3a2

LEVELS VIEWSPEC 1a3b

The Levels viewspec specifies how many levels of the file structure are to be displayed. Initially, level is set to its standard value of ALL. 1a3b1

DNLS displays only statements whose level is equal to or higher than the current level specification. This viewspec also affects the output in the Output Device command and restricts the effect of the Substitute and Assimilate commands. 1a3b2

d sets L to 1
c sets L to ALL
a sets L to L-1
b sets L to L+1
e sets L relative 1a3b2a

(i.e. L is set to the level of the first statement to be displayed by the command, i.e. the statement specified in the command.) For example, if a "jump to item" specified a statement whose statement number was "5a2", only first, second, and third level statements would be displayed. 1a3b2a1

where L = current level specification 1a3b3

Note: it is possible to set the Levels viewspec to 0 by use of the a viewspec. However, this setting is meaningful only if the origin statement is displayed. When the Levels viewspec in is effect, only the origin statement is displayed. 1a3b4

LINES VIEWSPEC 1a3c

The lines viewspec is a value from 1 to ALL which allows the user to specify how many lines of each statement are to be displayed. The lines viewspec is preset to ALL; if the user changes it to, for example, 3, only the first three lines of any statement will be displayed. 1a3c1

The codes for setting the lines viewspec are as follows: 1a3c2

t sets T to 1
s sets T to ALL
q sets T to T-1
r sets T to T+1 1a3c2a

LINES AND LEVELS VIEWSPECS 1a3d

In addition, to the viewspecs for lines and levels there are two extremely useful codes that affect both levels and lines: 1a3d1

x sets levels and lines to 1
w sets levels and lines to ALL 1a3d1a

STATEMENT NUMBERS ON/OFF (Codes m/n) 1a3e

Normally, when a statement is displayed, its statement number is not printed at the beginning of the first line. Statement numbers may be seen by using the viewspec "m". 1a3e1

m turns statement numbers ON
n turns them OFF. 1a3e1a

The standard setting for this viewspec is OFF (n). 1a3e2

STATEMENT NAMES ON/OFF (Codes C/D) 1a3f

Normally, when a statement is displayed, its statement name (if any) is visible. 1a3f1

C turns statement names ON
D turns them OFF 1a3f1a

The standard setting for this viewspec is ON (C). 1a3f2

BLANK LINES BETWEEN STATEMENTS ON/OFF (Codes y/z) 1a3g

The viewspec code "y" causes DNLS to put blank lines between statements. This makes the display more readable. 1a3g1

y turns blank lines ON
z turns them OFF. 1a3g1a

The standard setting for this viewspec is OFF (z). 1a3g2

INDENTATION OF STATEMENTS ACCORDING TO LEVEL ON/OFF (Codes A/B) 1a3h

DNLS normally indents according to level when it displays statements. This can be suppressed by the viewspec "B", causing all statements to be displayed flush at the left margin. 1a3h1

A turns indenting ON
B turns indenting OFF 1a3h1a

The standard setting for this device is ON (A). 1a3h2

CREATE NEW VIEW (Code F) 1a3i

The VIEWSPEC code f has a special effect; instead of setting a parameter, it acts as a "command," causing the display to be recreated and putting into effect any parameter changes that have been made since the last time the display was recreated. 1a3i1

AUTOMATIC DISPLAY RECREATION (Codes u/v) 1a3j

Certain commands cause the display to recreated when executed. The user may defer display recreation (i.e., until the user issues a command which specifically recreates the display, such as Jump to item. or issuing the "f" viewspec) by using the Viewspec "v". This feature is useful when the user is performing a repetitious series of insert statements, Xset commands, etc. However, caution should be exercised when using this viewspec as the user may unintentionally affect statements previously moved, inserted, etc. while this viewspec is in effect. 1a3j1

u causes the display to be automatically recreated 1a3j1a

v inhibits automatic display recreation 1a3j1b

The normal setting is u (recreate display) 1a3j1c

DISPLAY MODE BRANCH-ONLY/NORMAL/PLEX-ONLY (Codes g/h/i) 1a3k

When the display mode is BRANCH-ONLY, DNLS looks for the end of the branch defined by the display-start statement. If it comes to the end of the branch, it ends the display there. Thus, in effect, it displays only one branch (of course, the branch may not fit on the display, in which case the BRANCH-ONLY mode makes no difference for that view). 1a3k1

Similarly, when the display view is PLEX-ONLY the display is restricted to the plex defined by the display-start statement. 1a3k2

Normally, DNLS keeps putting more statements on the display until the screen is full or the end of the file is reached. 1a3k3

g sets view to BRANCH-ONLY 1a3k3a

h sets it to NORMAL 1a3k3b

l sets it to PLEX-ONLY 1a3k3c

The default setting is normal (h). 1a3k3d

This viewspec affects Output Device, Output Quickprint, Output Sequential File, and Substitute commands. 1a3k4

FROZEN STATEMENT DISPLAY ON/OFF (Codes O/P) 1a3l

If this feature is ON, any statements that have been frozen with previous "Freeze" commands (see Section 4 -- 10707,) are displayed at the top of the screen. Below the last frozen statement is a dotted line, followed by as much of the normal display as will fit. 1a3l1

o turns frozen statements ON 1a3l1a

p turns frozen statements OFF. 1a3l1b

The standard setting is OFF (p). 1a3l1c

VIEW SET COMMAND 1a3m

The View Set command enables the user to use the viewspec features of DNLS at any time (i.e. besides during link, output device, jump to, substitute, etc. operations). 1a3m1

v[iew set] VIEWSPECS CA 1a3m1a

where VIEWSPECS = any series of valid viewspec codes 1a3m2

Viewspecs activated by the View, Jump to, etc. commands remain in effect until deactivated by their opposites in subsequent commands, or until the user leaves DNLS. 1a3m3

VIEWSPEC DISPLAY AREA AND DEFAULTS VIEWPSECS 1a4

The current settings of six VIEWSPECs are displayed on two lines in the upper left-hand corner of the screen. 1a4a

The top lines shows "L" and "T", which appear either as numbers or as the word "ALL." 1a4a1

The second line shows four VIEWSPECs: 1a4a2

g, l, or h for branch-only, plex only, or normal mode 1a4a2a

i, j, or k for content-analyzer on, off, or reversed 1a4a2b

The use of content analyzer patterns and the viewspecs which effect them are described in the L10 Programming Guide (see -- 9246,). 1a4a2b1

m or n for statement numbers on or off 1a4a2c

u or v for recreate or defer recreate 1a4a2d

MULTIPLE DISPLAY AREAS 1b

Ordinarily, in DNLS, the user has one "view" of a file. There are a set of commands which, however, enable the user to expand the number of views he may have of the same and/or other files. This feature is governed by the Goto Display area subsystem which consists of the following command set. 1b1

GOTO DISPLAY AREA CONTROL 1b1a

This command allows the user to execute commands which control the number of views the user may have of files. 1b1a1

g[oto] d[isplay area control] 1b1a1a

Once the user enters the sequence of characters "g d", DNLS expects any of the following subcommands. 1b1a2

HORIZONTAL SPLIT 1b1a3

This command splits the display horizontally. 1b1a3a

h[orizontal split] BUG CA 1b1a3a1

The display is split where the BUG occured horizontally (into an upper and lower segment) at the bugged location moving the image of the original display area to the upper or lower segment depending on whether the cusor is above or below the bugged position when the final CA is input. 1b1a3b

No display area will be created which is smaller then two lines by 20 columns (using the character size of the original display area). 1b1a3b1

VERTICAL SPLIT 1b1a4

This command splits the screen vertically. 1b1a4a

v[ertical split] BUG CA 1b1a4a1

The display area is split where the BUG occured vertically (into a left and right segment) at the bugged location moving the image of the original display area to the left or right segment depending on whether the cursor is to the left or right of the bugged position when the final CA is input. 1b1a4b

No display area will be created which is smaller then two lines by 20 columns (using the character size of the original display area). 1b1a4b1

MOVE BOUNDARY 1b1a5

This command enables the user to move view area boundaries. 1b1a5a

m[ove boundary] BUG1 BUG2 CA 1b1a5a1

The selected boundary (BUG1) is moved to the new position (BUG2). A boundary will not be moved passed a boundary of a neighbor. A boundary is moved for all display areas for which it is a boundary. Any resulting display area which is smaller than two lines by 20 columns will be deleted. 1b1a5b

FORMAT DISPLAY AREA/CHARACTER SIZE 1b1a6

This command allows the user to change the image size of the character on the display. 1b1a6a

f[ormat display area] c[haracter size] NUMBER CA 1b1a6b

The current character size of the display area which currently contains the cursor is displayed, and the user may type a number (0, 1, 2, 3) for a new character size. The final CA causes the character size to be changed. The horizontal and vertical increments are automatically adjusted. Different display areas may simultaneously have different character sizes. 1b1a6c

CLEAR DISPLAY AREA 1b1a7

The bugged display area is cleared, i.e. the image is erased, the return and file return rings are released, and the association of a file with that display area is removed. The display area itself is not deleted. 1b1a7a

c/lear display area/ BUG CA 1b1a7a1

One may freely edit and jump using several display areas. The position of the cursor is used to resolve ambiguities. 1b2

For example, If one executes a Jump command, the position of the cursor when the final command accept is entered determines in which display area the new image is to appear. 1b2a

Also, If one changes viewspecs using the leftmost two buttons of the mouse, the viewspecs of the display area containing the cursor when the buttons go down are used as the initial values and are displayed in the viewspec area. When the buttons are released, the display area containing the cursor receives the new viewspecs. 1b2b

Section 6. DNLS/EXEC

1

INTRODUCTION 1a

The only EXECUTIVE command documented here is the DNLS access command. All other commands of interest to the DNLS user are documented in the TNLS User Guide (see -- 7470,). 1a1

ACCESSING DNLS 1b

In order for the user to enter DNLS, he must use the EXECUTIVE command DNLS. 1b1

```
@nls CR
[id:] IDENT CR
[device:]  d[isplay]                                        1b1a
```