RA3Y 22-JAN-75 11:21 31686

visit of Larry MacKechnie

I would like to meet with Mr. MacKechnie when he comes to ARC. As you probably know, Roger Hough and I have been doing an NSF study on computer, audio, and television conferencing. It might be nice to invite Roger to any meeting. 1

RA3Y 22-JAN-75 11:21 31686

visit of Larry MacKechnie

(J31686) 22-JAN-75 11:21;;;; Title: Author(s): Raymond R. Panko/RA3Y; Distribution: /JCN([ACTION]) DCE([ACTION]) RWW([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: RA3Y;

JMB 22-JAN-75 11:39 31687

A puzzle in Sendmail, for the bottom of your pile

This question is not urgent, so look into it whenever you get through with more urgent things

JMB 22-JAN-75 11:39 31687

A puzzle in Sendmail, for the bottom of your pile

A certain document was sendmailed to group UD for action and DIRT for info. The copy I received was marked FOR INFO ONLY. I am a member of both UD and DIRT; now, why didn7t my copy get marked FOR ACTIOn? I am in the habi of ignoring info-only items for several days, and this could have been unfortunate in the case of this particular document. I thought that if I sent an item to a group for info and also listed some of the members separately for action, the latter would get one copy only, and it would be marked FOR ACTION? 1

JMB 22-JAN-75 11:39 31687

A puzzle in Sendmail, for the bottom of your pile

(J31687) 22-JAN-75 11:39;;;; Title: Author(s): Jeanne M. Beck/JMB;
Distribution: /FEED([ACTION]) ; Sub-Collections: SRI-ARC; Clerk:
JMB;

FEED 22-JAN-75 15:26 31688

Sample message for Thursday's Class

This is just to show what it looks like when someone sends you a message using Sendmail. 1

FEED 22-JAN-75 15:26 31688

Sample message for Thursday's Class

(J31688) 22-JAN-75 15:26;;;; Title: Author(s): Special Jhb
Feedback/FEED; Distribution: /HEB([INFO-ONLY]) RA3Y([INFO-ONLY])
RAH([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: FEED;

FEED 22-JAN-75 15:19 31689

In the begining and JFR

I've moved your message about begining to bugs - so will be fixed one of these days and your suggestion about Jump to File Return has been moved to the design recommendations section, Thanks for the input, --Susan/FEED

1

In the begining and JFR

(J31689) 22-JAN-75 15:19;;;; Title: Author(s): Special Jhb
Feedback/FEED; Distribution: /RLL([INFO-ONLY]) FEED([INFO-ONLY])
; Sub-Collections: SRI-ARC; Clerk: FEED;

FEED 22-JAN-75 15:31 31690

Action & Info

Got your message about Action and Info mix ups. I'll put it in the design bugs branch of feedback for future consideration.
--Susan/FEED

1

FEED 22-JAN-75 15:31 31690

Action & Info

(J31690) 22-JAN-75 15:31;;;; Title: Author(s): Special Jhb Feedback/FEED; Distribution: /JMB([INFO-ONLY]) FEED([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: FEED;

MAP2 22-JAN-75 15:44 31691

re: journal mail from NDM

NDM 21-JAN-75 15:04 31680
Summer 74 Dialog on Marketing Information System for SRI
Location: (HJOURNAL, 31680, 1:w)
*****Note: [INFO-ONLY] *****

1

MAP2 22-JAN-75 15:44 31691

re: journal mail from NDM

(J31691) 22-JAN-75 15:44;;;; Title: Author(s): Michael A. Placko/MAP2; Distribution: /PWO([INFO-ONLY]) ; Sub-Collections: NIC; Clerk: MAP2;

MAP2 22-JAN-75 15:45 31692

re: name change for Susan ...

SRL 14-JAN-75 05:41 25080
Name Change - Roetter not Lee
Location: (HJOURNAL, 25080, 1:W)
*****Note: [INFO-ONLY] *****

1

MAP2 22-JAN-75 15:45 31692

re: name change for Susan ...

(J31692) 22-JAN-75 15:45;;;; Title: Author(s): Michael A. Placko/MAP2; Distribution: /PWO([INFO-ONLY]) ; Sub-Collections: NIC; Clerk: MAP2;

Table of Contents 1

Chapter 1: Introduction to the Datacomputer 2

Introduction 2a

The datacomputer is a shared large-scale data utility system designed to serve the computers on the ARPA network. It may be thought of as a "black box" that performs data storage and retrieval functions in response to commands phrased in a standard notation, called datalanguage. 2a1

This document describes the currently-running version of the datacomputer software, and includes information about how a user program can access the system, transmit datalanguage, process the datacomputer's responses, and transmit and receive data over the network. 2a2

The datacomputer in its full implementation will provide an on-line storage capacity of one trillion bits and an extensive set of services to user programs. The present version is a preliminary version, providing a limited amount of storage and a restricted set of user functions. Subsequent versions will progressively enlarge the range of services and the amount of storage available for users. 2a3

Chapter 2: Containers 3

Containers 3a

The container is a basic concept in datalanguage. A container is an imaginary box which, like a FORTRAN variable, may contain data; a container may also enclose other containers. 3a1

The description of a container has several parts. It includes the container's ident, type, and size, and perhaps some additional attributes. The container's ident, or simple name, is a string of 100 or fewer letters, digits or the special character %, by which datalanguage requests refer to the container. The first character of an ident must be a letter or the character %. Certain reserved words may not be used as container idents; these are listed in Appendix B of this document. 3a2

Some sample idents are:

```
AVERYLONGIDENTABCDEFGHIJKLMNOPQRSTUVWXYZ
PEOPLE
WEATHEROSTATIONS
%CCA
```

3a2a

Containers are of four types, depending on their contents. 3a3

A container that is a LIST contains some number of other containers. The LIST-members may be containers of any data type, but they must all have the same description. PEOPLE (above) is an example of a LIST. 3a3a

A container that is a STRUCT, or STRUCTURE, contains some number of other containers, which need not have identical descriptions.<NOTE: STRUCT and STRUCTURE are synonyms in datalanguage.> The descriptions of all the containers that are enclosed by the STRUCT form part of the description of the STRUCT itself; and on every occurrence of the STRUCT every one of its sub-containers must appear in the same order. ADDRESS is an example of a STRUCT. 3a3b

A container of type BYTE contains one byte of data. 3a3c

A container that is a STR or STRING contains a string of bytes.<NOTE: STR and STRING are synonyms in datalanguage.> FIRST is an example of a STR. The user can specify the byte size of BYTEs and STRs and can indicate an interpretation of 7- or 8-bit ASCII or uninterpreted (See below). 3a3d

A LIST or STR has a size associated with it. The size may be fixed or variable. The size of a STR is the number of characters in it, while the size of a LIST is the number of elements in the LIST. 3a4

Outermost Containers 3b

A container that is not contained by any other container is called an outermost container; outermost containers are different in several respects from other containers. 3b1

An outermost container in datalanguage has a function, which is either FILE, PORT, or TEMPORARY PORT [which may be abbreviated TEMP PORT]. A FILE contains data kept in the datacomputer. When a FILE is created [see below], datacomputer space is allocated for it. A PORT describes data that is transmitted to or from the datacomputer. A TEMP PORT is a PORT whose description is not permanently stored, unlike the descriptions of other containers. TEMP PORTs vanish at the end of the session in which they were created. 3b2

The Directory 3c

The ident of an outermost container, whether it is a FILE or a PORT, is unlike other idents, in that it is entered in the datacomputer's directory. The directory is conceptually a tree; the entries in it are called nodes. A node may have one or more subordinate nodes, unless it represents a container, in which case it cannot. 3c1

Only a bottom-most node of the directory may be a container ident; only an outermost container has its ident entered in the directory. 3c2

Normally, the first thing a user does after attaching to the datacomputer is log in to a directory node. For most purposes, he only sees his login node and the part of the directory that is subordinate to his login node. (The LOGIN request is discussed in detail in Chapter 4.) 3c3

Pathnames 3d

Pathnames are used to reference nodes in the directory tree by describing a path through it. They have the general hierarchical form

NODE1,NODE2,..,NODEn

where NODE2 is a node directly subordinate to NODE1. 3d1

There are several varieties of pathnames. The two classes of directory objects referenced by pathnames are closed nodes (including all nodes that are not outermost containers and all outermost containers that are not OPEN) and OPEN outermost containers. There are three areas in which names can be found: the TOP, LOGIN, and OPEN contexts. Thus there are six possible pathname types, only five of which are reasonable. (A closed node in the OPEN context isn't.) 3d2

Closed nodes can be referenced either by a complete pathname (started with the reserved word %TOP), which causes the name search to be anchored at the top of the directory tree, or a LOGIN pathname, which anchors the search at the current LOGIN node. Either pathname may contain passwords. (Passwords are discussed in Chapter 4.) 3d3

OPEN nodes may be referenced by a simple complete pathname or a simple LOGIN pathname, neither one of which can contain passwords, or by an OPEN node simple name. An OPEN node simple name is the name of the outermost container. 3d4

Creating Nodes 3e

A node in the directory is created with a CREATE request. Such a request has the form

```
CREATE <pathname> ;
```
3e1

Only one node may be created by a single CREATE request, and a higher-level node must always be created before one subordinate to it. The reserved words listed in Appendix B may not be used as directory node names. 3e2

As an example, let us create the outermost container F, a LIST of 4-character strings; the container's ident will be entered in the directory. We assume that nothing is presently in the directory, so we must start by creating the topmost node.

```
CREATE CCA;
CREATE CCA,DATA;
CREATE CCA,DATA,F FILE LIST
          FOO STR (4) ;
```

Now that CCA and CCA,DATA have been created, we could create CCA,DATA,G with only one CREATE request; i.e.

CREATE CCA,DATA,G PORT LIST etc, 3e2a

Creating Containers 3f

Outermost containers are created by a more complicated form of the CREATE request. The CREATE statement must tell the datacomputer all about the container, for example, its ident, function, size, and data type are included. An outermost container and all its subcontainers must be created at once, with one CREATE request. 3f1

The CREATE request causes the description to be stored. It also causes space to be allocated if the container is a FILE. 3f2

The full BNF in Appendix A indicates succinctly the precise syntax of the CREATE statement. It is worth looking at a few examples before looking at all the details of descriptions. A LIST of STRings:

```
CREATE ALPHA FILE LIST SUBCONTAINEDSTRING STR (44) ;
```

Here the size of the outermost LIST is omitted, so the datacomputer will calculate a default size. 3f3

A LIST of STRUCTs, each of which contains three strings:

```
CREATE BALLTEAM FILE LIST (25)
            PLAYER STRUCT
              NAME STR(20)
              POSITION STR(2)
              UNIFORM%NUMBER STR(2)
            END;
```
3f4

The datacomputer will allocate enough space for the file BALLTEAM to hold 25 copies of the STRUCT named PLAYER. Note that END is required to terminate the description of the STRUCT. 3f5

The example previously discussed:

```
CREATE PEOPLE FILE LIST
    PERSON STRUCT
      NAME STRUCT
         FIRST STR(15)
         LAST  STR(15)
      END
      ADDRESS    STRUCT
         STREET STR(15)
```

```
            CITY STR(15)
            STATE STR (15)
        END
        SOCSECNO STR(10)
    END;                                                    3f5a
```

The elementary data types are BYTE and STR. Containers of these types contain data, not other containers. 3f6

STRings and LISTs must have a size. For STRings, the size is the number of bytes in the STRing. For LISTs the size is the number of LIST members (e.g. the number of PERSONs in PEOPLE above.) The three forms for indicating the size are: 3f7

(n) -- a fixed size of n 3f7a

(m,n) -- a minimum size of m and a maximum of n 3f7b

(,n) -- a minimum dimension of 0 and a maximum of n where m and n are positive integers. 3f7c

For an outermost LIST or STRing, no size need be specified. The default minimum is 0, and the default maximum is based on what will fit in the default space allocation. 3f8

The datacomputer needs a way to find the end of the data in variable-sized LISTs and STRings. The three options are a preceding count, a trailing character, and punctuation (i.e. a device-dependent marker). A one-byte preceding count is indicated with the keyword parameter

```
,C=1
```

Version 0/10 cannot handle counts larger than one byte. Thus, if there is a count, then the maximum dimension must be small enough to fit into a one-byte count. (Byte size is discussed further below.) The value of the count does not include the count byte itself. 3f9

The syntax to indicate that there is a one-byte delimiter is,

```
,D=n
```

or

```
,D='a'
```

where n is a decimal number and a is any ASCII number, letter or special character. 3f10

The datacomputer considers punctuation to be different from delimiters. Punctuation over the network is a special character (specifically EOR, EOB, or EOF) inserted in the data but not considered part of the data. This is indicated by,

```
P=EOR
P=EOB
```

and

```
P=EOF
```

3f11

A fixed-size container (including a STRUCT) may have a P, D or C parameter, but no container (fixed or variable) may have more than one of these. 3f12

A datacomputer FILE can be punctuated, but none of its sucontainers can be. The FILE punctuation defaults to EOF. Variable-length subcontainers must have either a C (count) or D (delimited) parameter. 3f13

If a variable-sized PORT does not have a P, D, or C parameter, then it defaults to P=EOF. Variable-sized subcontainers of a PORT default to P=EOR. 3f14

Punctuation is hierarchical. A container that is punctuated with EOR cannot contain one that is punctuated with EOB or EOF. A container that is punctuated with EOB cannot contain one with EOF. If higher punctuation is found in a data stream where the datacomputer is looking for lowepunctuation (e.g., an EOB where an EOR is expected), the higher punctuation implies the lower. 3f15

The interpretation of a STR is one of ASCII (i.e. 7-bit ASCII), ASCII8, or BYTE, as in the following three examples: 3f16

```
A STR ASCII (5)
```

3f16a

```
P STR ASCII8 (1,10), C=1
```

3f16b

```
WALDO STR BYTE (73)
```

3f16c

The default byte size for BYTE is 36 bits. BYTE is optional if the byte size is given explicitly with the keyword parameter,

```
B=n
```

where n is a positive integer less than or equal to 36. The

,B=n option may not be used for ASCII STRings. If no byte size or interpretation is given, then the STR is 7-bit ASCII. 3f17

At times the datacomputer needs to fill in a value or a part of a value. The user can specify a fill character thus:,

```
F='a'
```

or ,

```
F=n
```

where a is an ASCII character and n is a decimal number. The default fill character is blank for ASCII data and zero for non-ASCII data. 3f18

Note that a byte size and a fill character can apply to a STRUCT or a LIST as well as a STR or a BYTE. Consider the following:

```
CREATE F FILE LIST
         R STRUCT, B=36
           A STR (5)
         END;
```

The byte size of A is 7. A takes up 35 bits. There is one "unused" bit after A before the next R. Thus, R must be filled. Even though the data (i.e. A) is ASCII, R is non-ASCII because it does not have a 7-bit byte size. Hence, the default filler of 0 is used for the bit. 3f19

The rules for punctuation, byte size and fillers are simple but not at all intuitive. In general, specifying punctuation rather than relying on defaults helps avoid errors. Also

```
LIST <pathname> %DESC;
```

will output a complete description, including all default lengths, dimensions, punctuation, byte sizes and fillers. (The LIST command is discussed more fully below.) It is often instructive to look closely at the %DESC to see where it is different from what the user expects. 3f20

BYTEs and STRings that will frequently be used for retrieval may be inverted. For members of outer LISTs, the option,

```
I=D
```

is used, for members of inner LISTs, the option,

I=I

is used. Inversions and the difference between outer list members and inner list members is discussed more fully in the section on WITH.

3f21

Chapter 3: Directory Commands 4

OPEN 4a

Before data can be input to or read from a FILE or PORT, the container must be open, and a mode must be specified for it. The mode of a FILE or PORT, which is set when the container is opened, determines the legality of various operations on that container. 4a1

Mode is one of READ, WRITE, or APPEND. Data can only be transmitted out of a FILE or PORT that is open in READ mode, but either out of or into a FILE or PORT that is open in WRITE or APPEND modes. The difference between WRITE and APPEND lies in their treatment of any data that is already in the container when it is opened. When an assignment is made to a container that was opened in WRITE mode, any data it contained previously is thrown away, but a container opened in APPEND mode has newly-arriving data written after the end of any already-present data, which is thus preserved. 4a2

A variation of WRITE and APPEND is WRITE DEFER and APPEND DEFER. When DEFER is indicated as part of the mode, a more efficient technique of updating the inversion is used. 4a3

When a FILE or PORT is created, it is opened in WRITE mode. A FILE/PORT that already exists may be opened with an OPEN request:

OPEN <pathname> <mode> ;

which specifies the name of the container that is to be opened and the mode of opening. The name can be either a complete pathname (started with the reserved word %TOP) or it can be a login pathname, started with a node immediately subordinate to the current login node. The mode must be one for which the user has privileges (see Chapter 4). The mode argument may be left out of an OPEN statement, in which case the container is opened in READ mode if it is a FILE and WRITE mode if it is a PORT. Two outermost containers with the same ident may not be open at the same time. 4a4

For example, to read data that was previously stored in CCA.DATA.F, a file, either

OPEN CCA.DATA.F;

or, if the current login node is CCA,

OPEN DATA,F;

WILL OPEN F PREPARATORY TO DATA TRANSFER REQUESTS. 4a4a

MODE 4b

The mode of a container that is already open may be changed with the MODE statement:

MODE <paragraph> <mode> ;

The pathname can be a simple complete pathname (i.e. a complete pathname with no passwords), a simple login pathname, or a node name. 4b1

CLOSE 4c

The complement of the OPEN request is the CLOSE request. When you have finished using an open container, close it with

CLOSE <pathname>

;where pathname must be the simple pathname of an open container. Closing a FILE/PORT with a function of TEMPORARY PORT has the effect of deleting its description from the datacomputer. 4c1

DELETE 4d

The ability to delete directory nodes is useful in maintaining a data base at the datacomputer. The DELETE request allows one to delete one or several outermost containers and all the data they contain.

DELETE <pathname> ;

causes the node named by <pathname> to be deleted from the directory. The pathname must be the login pathname. Thus, only nodes subordinate to the login node can be deleted. The node cannot have any subordinates.

DELETE <PATHNAME>,** ;

deletes the node and all subordinate nodes. If any of the deleted nodes are outermost containers, the container descriptions and any associated data are deleted as well. The DELETE request need not be used on TEMPORARY PORTs, as they are

automatically deleted either when they are closed,or at session end. 4d1

If the data stored in FILE is to be deleted, but the container description itself retained in storage, the DELETE request cannot be used. Instead, CREATE a port B with a description matching the container A that is to be emptied, and execute the assignment A = B with no data in B. the effect of this assignment is to delete all the data from A. 4d2

LIST 4e

The LIST request is the means by which the user interrogates the datacomputer about his environment. The request has two arguments: the node or nodes which are the object of the inquiry, and the type of information desired. 4e1

The first argument consists of a set of nodes in the directory. Possible node sets are: 1) a single node, 2) all nodes directly subordinate to a given node, 3) a node and all its subordinates, and 4) all open files and ports. A single node is specified with a full pathname, which can include passwords and can be anchored at the top node (%TOP). The set of a node's direct subordinates is indicated with either a "*" (the login pathame is implicit) or a full pathname followed by a "*". Either "**" or a full pathname followed by a "**" designates a node and all its subordinates. The set of all open nodes is referenced by %OPEN. %TOP alone defaults to %TOP.**. 4e2

There are five kinds of available information. These are: 4e3

node names and related data (node type, privileges, and possibly mode and connected argument), 4e3a

parsed data descriptions (of FILEs and PORTs), 4e3b

original source text of data descriptions, 4e3c

allocated space (for FILEs), and 4e3d

privilege blocks associated with nodes. 4e3e

These information options are specified by %NAME, %DESC or %DESCRIPTION, %SOURCE, %ALLOC or %ALLOCATION, and %PRIV or %PRIVILEGE, respectively. The default option is %NAME. 4e4

Not all of the kinds of information are available for all of the possible node sets. The options that are available are:

Node Set	Option
<pathname>	%DESC
<pathname>	%NAME
<pathname>	%SOURCE
<pathname>	%ALLOCATION
<pathname>	%PRIVILEGE
<pathname>.*	%NAME
<pathname>.**	%NAME
<pathname>.**	%SOURCE
%OPEN	%NAME
%OPEN	%DESCRIPTION
%OPEN	%SOURCE
%OPEN	%ALLOCATION

4e5

Chapter 4: Security and Passwords 5

Introductory Concepts 5a

The 0/10 version of the datacomputer provides file-level security (restricted access to nodes and attendant data) by means of a system of privilege blocks, described in the following sections. One or more (or no) blocks may be associated with a particular node. Each privilege block defines a class of users who may be given access to the node and the set of privileges to be granted to such users. Whenever a user attempts to access a node or file, the datacomputer will scan that node or file's privilege block(s), if any, to ensure that the user is 'legal' and to determine what privileges will be allowed. 5a1

Chapter Organization 5b

This chapter is divided into three principal parts. The first sections describe what privilege blocks are and how they provide file security functions for datacomputer users, and introduce the reader to the security features of datalanguage. The second part completely specifies the datalanguage needed for creating, deleting and manipulating privilege blocks, and completes the description of their components begun in the first part. The third section offers several examples of how to add, delete and look at privilege blocks. 5b1

Gaining Access to Nodes: LOGIN 5c

Every node in the directory has certain privileges associated with it. For example, the ability to create inferior nodes, or to read or write file data, are privileges which may be granted or denied to a particular node. When a user initially connects to the datacomputer he is automatically connected to the top node of the directory tree (%TOP), and he (i.e., the %TOP node) is granted minimal privileges. To acquire more, he must log in to some node, which is called, curiously enough, the login node. 5c1

Logging into this node establishes the user's identity for subsequent pathname references . It should be kept in mind that a user is identified to the datacomputer only by his login node. Thus, throughout this chapter, the terms 'user-id' or 'user name' are to be understood to mean nothing more than the full pathname, including the specified privilege block (if any) at each level (2), of the node to which the user has logged-in. 5c2

Whenever a logged-in user references a node, the login pathname

is compared against the user-id field of every block in the node's privilege block list. If a block is found whose user class description includes the pathname of the login node, the privilege-set described by the block will be added to (or taken away from) the privilege set already given to the login node. 5c3

Privileges 5d

Privilege set specifications come in two flavors: privileges to be granted (added) to the node and privileges to be denied (taken away). If a privilege is not specified (as either grant or deny), then that privilege (or denial of it) is passed, unchanged, from the superior node to its subordinate. At each node level, the deny bits specified in the given privilege block are NOT-AND'ed with the privileges of the superior node. Then the grant privileges are OR'ed with the result, to yield the privilege set for that node. 5d1

It is important to understand that privileges may be added and taken away at every level of the pathname. For example, suppose the login node has the privilege set <CLWA>, <NOTE: This is a shorthand way of saying 'this node has been granted control <C>, login <L>, write-to-file <W> and append-to-file <A> privileges. Specific privileges are described in detail below.>, and a subnode's privilege block specifies: grant read privilege(G=R), and deny write privilege (D=W). The result at thesubnode would be the final privilege set of <CRA> (4). 5d2

Pathnames may be qualified or unqualified. A qualified pathname is one containing password strings for the purpose of gaining particular privileges upon opening the node, e.g.,

NODE1('PASSWORD1'),NODE2,NODE3('PW3')

is a pathname qualified at the first and third levels by the passwords 'PASSWORD1' and 'PW3', respectively. The pathname NODE1,NODE2,NODE3, on the other hand, is unqualified. Prior to version 0/10, all pathnames were unqualified. 5d3

Note that a node can never look at, modify, or affect a superior node in any way not possible at the level of the superior. That is, if a user cannot look at the privilege blocks for a node, he cannot acquire that privilege for that node from an inferior one. However, an inferior node may well have privileges relative to its subnodes that its superior does not have relative to its subnodes. For example, scanning along the pathname A,B,C,D,E,...., A,B,C may have only read privileges, but does not have write privilege. Now, the node A,B,C,D may be granted write

privilege at level D (thus awarding A,B,C,D read/write privileges), this does not affect A,B,C, It still has only read privilege. 5d3a

Privilege Block 5e

Privilege blocks are data structures which define access to nodes. Each privilege block is associated with one particular node. Any node in the directory, including ports and file, may have privilege blocks defined for them. A node may have any number (including zero) of privilege blocks. When an attempt is made to access a node which has privilege block(s), those blocks are scanned for a user-id corresponding to the current login pathname and for a password string matching that supplied by the user in the request referencing the node (e.g., LOGIN, OPEN, DELETE, etc.). If a match is found, the matching block's privilege set bits are examined and the appropriate privileges are granted/denied the node. The matching algorithm is described below in more detail. 5e1

Each privilege block can contain:

user name
host name
socket number
password character string
grant privileges
deny privileges 5e2

Each of the above fields falls into one of two categories: 5e3

A description of the group of users which may access the associated node; 5e3a

The privileges to be granted to these users. 5e3b

The privilege block is completely specified at the time it is created. When a node is referenced, only the password string, if any, is required; the user-id (including host name and socket number), has been retained by the login process. 5e4

The login privilege is not propagated to subnodes. It applies only to the node for which it is explicitly granted. See below. 5e5

Privilege blocks are created by the datalanguage command CREATEP. They are deleted by the command DELETEP. Existing privilege blocks may be displayed via the LIST nodename

%PRIV(ILEGE) command. The full syntax of these commands is described below. 5e6

User Identification Fields (User-ID) 5f

The user identification fields include some or all of the following: a valid login pathname or a class of login pathnames, the number of a host computer, the datacomputer socket number, and a password character string. These fields are discussed in more detail in the following sections. 5f1

Host 5g

The host name is an optional field. If specified, it must be a decimal number from 1 to 255 designating the number of the host computer. The host name cannot be a number greater than 255, or less than 1. It cannot be a character string, except for the special cases LOCAL and ANY. 5g1

LOCAL host indicates that the user should not have connected to the datacomputer via the ARPANET. Effectively, this means (at this time) that the user is located at CCA and is connected to the datacomputer via a local terminal. 5g2

The host name may also be ANY, which means that any host, foreign or local, is acceptable. 5g3

If a host name is not specified, the default value is ANY. 5g4

User Name 5h

The user name is the pathname or classname <note: user classnames are defined below> of the login node(s) which may gain access to the node associated with the privilege block. Note that a different privilege block must be created for each specific user permitted to use a given password. For example, if two different users, say CCA,WALDO and CCA,DINGLE, wanted to use the same password string ('FOO') to gain access to a node, two separate blocks would have to be created, one per specific user name. Thus, in this example, one privilege block would contain the information CCA,WALDO ('FOO'); the other , CCA,DINGLE ('FOO'). 5h1

If no user name is specified, the default is **, which grants any user access to the node. 5h2

Socket 5i

The socket number is a 32-bit number, e.g., 600403, or ANY.

This is an identification number assigned by the foreign host to the user logged in on that foreign host. Usage of the socket number in the CREATEP statement ensures that only specified users at the foreign host site may gain access to a particular node. 5i1

Socket number defaults to ANY. 5i2

Password 5j

A password consists of an alphameric string enclosed by single quote (') characters, e.g., P='FOO', Non-printing characters, except blanks, are not valid in a password string. Blanks may appear at any point in the quoted string. Tab characters are not permitted. 5j1

A privilege block need not contain a password. In this case, none should be given when referencing the node. Note that no password is not the same as, and is treated differently from, a null password (''). Null password is treated as a password of zero length, and must be supplied as such whenever the node is referenced. 5j2

Privilege Set Specification 5k

The following privilege bits are defined for 0/10: 5k1

LOGIN (L) 5k1a

In order to control login identities more closely, the ability to log in to a node is not passed to subordinates. As a result, -L (deny login) is meaningless. 5k1a1

CONTROL (C) 5k1b

Control includes complete subordinate control and privilege control. Control is required for creating and deleting nodes, files and privilege blocks. It is also required for listing privilege blocks. It is very powerful, and cannot be removed by an inferior: -C is not permitted. After 0/10, C may be split into meaningful components. 5k1b1

Data Control Privileges 5k1c

READ (R)
WRITE (W)
W implies R and A.
APPEND (A)
A does not imply R.
Conflicts are not allowed in one tuple, e.g.,
+R and -R. 5kic1

Ordering of Privilege Blocks 51

The ordering of privilege blocks is important. When a node is referenced, the privilege blocks (if any) for that node are scanned linearly for a password string matching the password entered by the user. If a match is found, the user-id of the privilege block is compared to the login identity. If they match, the associated privileges are granted/denied, and access appropriate to the granted privilege set are awarded to the node. If the end of the privilege blocks is reached without finding a password/user-id match, the node is opened with no privileges. 511

Since the privilege blocks are scanned linearly, their ordering defines their selectivity. For example, suppose a node to have two privilege blocks which specify the same password ('foo') but different login nodes, say, A and **, and suppose that the block with user name A grants greater privileges (read/write/append) than that with ** (which permits read). The proper ordering, as displayed by a

```
LIST WALDO,NODENAME %PRIV(ILEGE);
```

statement, is as follows:

```
(1),U=A,H=ANY,S=ANY,G=RWA
(2),U=**,H=ANY,S=ANY,G=R
```

<NOTE: U=** means that any user name will be accepted as valid> 512

If the order of these blocks were reversed, so that the block with the user name '**' were first, then whenever the password FOO was encountered the first block would be selected; i.e., every login pathname would match the '**', and the matching process would be complete. Thus, the block with the user name A would never be found, and the user A would be unable to open the node with the greater privileges which should be granted him. 513

In 0/10 the user is responsible for maintaining the desired search order, by adding and deleting privilege blocks via their

block index numbers. The datalanguage for this process is described below. Future versions of the datacomputer may provide an automatic ordering algorithm, which could be manually overridden, if desired. 514

User Classes ('Star' Feature) 5m

Classes of users may be given access to a node by specifying a user class as the user name instead of a single user. This is done by means of the '*' and '**' ('star' and 'star-star') features. If a star appears in a pathname, it is interpreted to mean: 'any single (non-null) partial pathname is acceptable here'. That is, if the nodes A,B,N1, A,B,N2, and A,B,N3 exist in the directory tree, usage of the user classname A,B,* would specify any of these three pathnames. Stars may appear at any number of levels; for example, if the nodes A,X,N1 and A,Y,N4 exist, then the user-name A,*,* would specify both of these nodes, as well as any of the previous three. The use of a star at any level implies that there must be a partial pathname at that level; e.g., the classname A,*,* could not specify node A or A,J. 5m1

User Classes, cont. ('Star-star' Feature) 5n

The use of a single star in a pathname indicates that a node must exist at the level corresponding to that of the star, and a star must be explicitly specified for each desired level. The star-star feature is designed to permit access to several levels of nodes. A star-star ('**') in a user name is interpreted to mean: 'any number (including zero) of partial pathnames are acceptable here'. Thus, referring to the example of the preceeding paragraph, A,B,N1 could be specified by any of the following:

```
A,B,N1,**
A,B,*,**
A,B,**
A,*,**
A,**
*,**
**
```

5n1

For 0/10, only trailing *'s and/or a final ** are allowed. The following, for example, are illegal:

```
A,*,C
A,**,C
A,*,**,D
A,**,*
```

```
*,B,**
**,*                                                    5n2
```

Datalanguage for File Security 5o

Two new datalanguage statements, CREATEP and DELETEP, create and delete privilege blocks. They are discussed in the following sections. The list command has a new option, %PRIV (or %PRIVILEGE), which allows the user to list the privilege blocks for a node. 5o1

CREATEP and DELETEP are privileged requests. They are only accepted when the associated node can be referenced with control privilege <C>. (This means that it may be necessary to login to some particular node before any privilege blocks can be added to another, and that passwords may be required for the login process or for referencing nodes superior to the node for which the privilege block is to be added.) 5o2

Creating Privilege Blocks: CREATEP 5p

Privilege blocks are created, and fully specified by, the CREATEP command. A fully specified CREATEP statement might appear as follows:

```
CREATEP NODE1('PW1'),NODE2, U=CCA.WALDO.*.**, H=34,
S=604320,
        P='SECRET PASSWORD', G=R, D=WA, N=2;
```

In this example, the node for which we are creating a privilege block is NODE1.NODE2. We must specify ('PW1') for NODE1 in order, perhaps, to gain control privileges at the first level. The parameters which follow the nodename is the privilege keyword list. These are discussed individually in the following sections, and are summarized in Appendices A and B. 5p1

CREATEP: User Name 5p2

The user name is specified by 'U=' followed by an unqualified pathname or classname string. The pathname may have any number of levels. It must not contain password strings for any level. 5p2a

The following are valid pathnames/classnames.

```
CCA
```

```
CCA,WALDO,DINGLE
CCA,*,*
CCA,**
*,*,*
*,**
**
```
5p2b

CREATEP: Host Number

5p3

The host number is specified by 'H=' followed by a decimal number from 1 to 255, or either of the strings LOCAL or ANY.

```
H=28
H=ANY
H=LOCAL
```
5p3a

CREATEP: Socket Number

5p4

The socket number is specified by 'S=' followed by the 32-bit foreign-host assigned decimal number corresponding to the directory the user is logged into at that foreign host, or the string ANY.

```
S=309483
S=ANY
```
5p4a

CREATEP: Password String

5p5

The password string is specified by 'P=' followed by any datacomputer string constant (tabs may not be included, although blanks are permitted), e.g., 'PASSWORD 1', '? * ++!!', or '' (null password). 5p5a

Note that if no password string is specified at CREATEP time, then that privilege block will have no password associated with it. No password is different from null password (P=''), which is a valid password zero characters in length. 5p5b

CREATEP: Grant Privileges

5p6

Privileges are granted by 'G=' followed by

```
C       (control)
```

L (login)
R (read file data)
W (write file data)
A (append data to file)

in any combination and in any order, e.g., G=CRAWL (all privileges), G=WAR (read/write/append), etc. 5p6a

CREATEP: Deny Privileges 5p7

Deny privileges are specified by 'D=' followed by R, W or A. Login (L) applies only to the node for which it is specified. It is not passed to subordinates. Control (C) cannot be removed by any inferior node, i.e., it is passed to all subnodes. 5p7a

CREATEP: Privilege Block Index 5p8

As CREATEP privilege blocks are created, they are assigned index numbers by the datacomputer. Block numbers are assigned to privilege blocks sequentially according to their search order. Block numbers can range from one to n, where n is the total number of password blocks in the search sequence. Blocks can be explicitly ordered by the user at CREATEP time by entering 'N=' followed by the number that the newly added block is to have in the search sequence. N must be greater than zero, and not greater than the total number of privilege blocks currently existing for the node. Note that this index is not in any sense a part of the data contained in the privilege block; it is merely the position of the block in the password block list. 5p8a

An example. If there were three blocks in the privilege block list for a node (NODE1),

1 U=AAA
2 U=CCC
3 U=DDD

and a new block were to be added between the first and second existing blocks, i.e., so that the new block would then occupy second position, we add a keyword, N=2, to a CREATEP command:

CREATEP NODE1,U=BBB,P='ZOO',N=2;

which results in the following privilege block list:

```
1  U=AAA
2  U=BBB
3  U=CCC
4  U=DDD
```

If N had been omitted, the new block would have been added at the end of the list. Note that the indices of the two blocks following the new one have been bumped by one. Similarly, if any block is deleted, the indices of all the following blocks are reduced by one. 5p8a1

Looking at Privilege Blocks: LIST 5q

In order to permit the user to list privilege block information, the %PRIV (or %PRIVILEGE) option has been added to the datalanguage LIST request. It looks like this:

```
LIST CCA,WALDO %PRIV   (or)

  LIST CCA,WALDO %PRIVILEGE
```

5q1

Passwords cannot be listed with the %PRIV option (or in any other way - so don't forget 'em!). Privilege block information is preceeded by the index number of that block. All other information in the privilege block is listed in a format similar to that which might be found in a CREATEP command, e.g. either of the LIST requests above might generate the following output from the datacomputer:

```
(1),U=CCA,WALDO,H=LOCAL,S=ANY,G=CRAWL
(2),U=CCA,*,**,H=ANY,S=ANY,G=RWAL
(3),U=*,**,H=32,S=654364,G=RL,D=WA
```

%PRIV may be used only when the controlling node has control privileges. 5q2

Deleting Privilege Blocks: DELETEP 5r

Privilege blocks may be deleted with DELETEP followed by the index number of the privilege block to be deleted.

```
DELETEP 3
```

The controlling node must have control privilege. 5r1

Example 5s

This example will create a node which will be the controlling

node for all other nodes at site CCA. Presumably, access to this controlling node would be restricted to very few persons at that site; 'super-users', as it were. This could be done by means of a password. Inaddition, anyone seeking control privileges for CCA might be required to be logged-in to some other (access restricted) node. The person with access to CCA would be responsible for creating subnodes, perhaps one for each programmer permitted to use the datacomputer. These individual programmers could then create their own directory structures (nodes, ports and files) in any manner they wish. 5s1

The site-node CCA is created by the following series of requests:

```
CREATE CCA;
CREATEP CCA,P='HONCHO',G=CL;
CREATEP CCA,P='FLUNKY',G=L;
LOGIN CCA('HONCHO');
```
5s2

The user is now logged in to CCA. He has control privileges. Next he creates a series of programmer-nodes, each with control privileges. Initially, two privilege blocks are created for each programmer node. One requires a password (known to, and probably specified by, the individual programmer), and the other requires no password and is accessible to anyone logged in to CCA or any of its subnodes. However, persons who log in to a programmer node without specifying a password are not given control privileges and thus cannot modify or delete anything that the programmer wishes to keep secure.

```
CREATE WALDO;   CREATE PWALDO,U=CCA,P='TURKEY',G=CL;
                CREATEP WALDO,U=CCA,**,G=L;
CREATE CLYDE;   CREATEP CLYDE,U=CCA,P='FETCH',G=CL;
CREATEP CLYDE,U=CCA,**,G=L;.
        .
        .
CREATE DINK;    CREATEP DINK,U=CCA,P='PODUNK',G=CL;
                CREATEP DINK,U=CCA,**,G=L;
```
5s3

After this is done, super-user checks the privilege blocks he has created, first at his own node level:

```
LIST %TOP.CCA('HONCHO') %PRIVILEGE;
```

and he receives a datacomputer printout in the following format:

```
(1),U=**,H=ANY,S=ANY,G=CL
```

```
(2),U=**,H=ANY,S=ANY,G=L
```

He next verifies that each of the programmer-node privilege blocks has been correctly entered, e.g.,

```
LIST WALDO %PRIV;
```

and the datacomputer replies:

```
(1),U=CCA,H=ANY,S=ANY,G=CL
(2),U=CCA,**,H=ANY,S=ANY,G=L                                    5s4
```

At this point, programmer Waldo tells super-user that he would rather have 'donkey' as his control password rather than 'turkey'. Since the user name (U=CCA) in Waldo's control privilege block is more restrictive than the user name (U=CCA,**) in the non-control privilege block, the first privilege block must be deleted and the new one added in the same position (N=1):

```
DELETE WALDO 1;
CEATEP WALDO,U=CCA,P='DONKEY',G=CL,N=1;
```

We now have the following directory:

```
CCA
CCA,WALDO
CCA,CLYDE
    .
    .
    .
CCA,DINK                                                        5s5
```

Each of the programmer-nodes listed above has its own password which is known to the person having access to that node. In addition, each is required to login to CCA before being able to acquire login and control privileges at its own level. (Most or all of the programmers at CCA are given only the password FLUNKY, which does not give control privileges. Thus, they cannot create or delete any nodes at the programmer-node level or look at the restricted data of any other programmers.)   5s6

As soon as he is informed that he may join the select international hoard of datacomputer users, Waldo rushes to his terminal to login:

```
LOGIN CCA('FLUNKY');
LOGIN WALDO('DONKEY');
```

Since he has logged in to his node using the password which grants control privileges, Waldo now creates BOOKFILE and BOOKPORT and reads some data into BOOKFILE from a TENEX file named TENEX-BOOK,FILE <NOTE: A TENEX filename is used in this example for the purpose of didactic clarity. In practice, this would usually be done only by local datacomputer users (users located at the site of the datacomputer). Remote users would have to arrange for operator intervention, if connecting to a file at the datacomputer site; or would specify the host name and socket number from which the data would be sent to the datacomputer.>:

```
CREATE BOOKFILE FILE LIST(,1000),P=EOF
  BOOK STRUCT
        TITLE STR (,100),C=1
        AUTHORS LIST(,5),C=1
          AUTHOR STR (,50),C=1
        PUBLISHER STR (,50),C=1

END;


CREATE BOOKPORT PORT LIST(,1000),P=EOF
  BOOK STRUCT
        TITLE STR (,100),P=EOR
        AUTHORS LIST(,5),P=EOB
          AUTHOR STR (,50 ),P=EOR
        PUBLISHER STR (,50),P=EOR
  END;

CLOSE %OPEN;

OPEN BOOKFILE WRITE;
OPEN BOOKPORT; CONNECT BOOKPORT 'TENEX-BOOK,FILE';
BOOKFILE=BOOKPORT;

CLOSE %OPEN;                                                    5s7
```

In order to permit others to look at his library file, Waldo creates a couple of privilege blocks. The first permits anyone at CCA to look at his book list, while denying him the right to change anything. The second is for Waldo's private use in changing the file:

```
CREATEP BOOKFILE,U=CCA,*,G=R,D=AW;
CREATEP
 BOOKFILE,U=CCA,WALDO,P='READ*MORE*EVERY*DAY',G=RWA;          5s8
```

Chapter 5: Assignment and For-loops 6

Assignment Involving Outermost Containers 6a

Transmission of data is achieved with an assignment. The syntax of an assignment request that involves two outermost containers is

<ident> = <ident>;

where the <ident>s are the node names of open outermost containers. The first ident in the statement is that of the receiving container; it must be open in either WRITE or APPEND mode. The second ident is that of the transmitting container; it can be open in any mode, but it must have READ privilege (see Chapter 4). If the second ident is a FILE, it must contain some data. 6a1

The containers in the assignment may be either files or ports. The various combinations are listed here, with a description of the action of the assignment request in each case. 6a2

Receiving container	Transmitting container	Comment
FILE	FILE	copies data from one FILE to another within the datacomputer.
FILE	PORT	transmits data from some source external to the datacomputer through a PORT, into a FILE.
PORT	FILE	transmits data from a FILE, where it is being kept in the datacomputer, through a PORT, to the outside world
PORT	PORT	transmits data from one place to another in the outside world, using the datacomputer only as a channel for transmission. 6a3

The Matching Rules 6b

In any assignment statement such as

X = Y;

(not only one involving two outermost containers) the two operands, X and Y, each has its own description. The datacomputer will transform the data in Y to match the description of X. In order for the datacomputer to be able to do this, the descriptions must match. This amounts to a restriction that only similar objects can be assigned to each other. Specifically, for two assignment-operands X and Y to match: 6b1

Rule 1a: X and Y must have the same type: LIST, STRUCT, or STR, or BYTE,

AND

Rule 1b: If X and Y are both LISTs, then they must have compatible sizes, or else X must be a PORT. The sizes are compatible if the minimum size of X is less than or equal to the minimum of Y and the maximum size of X is greater than or equal to the maximum size of Y. This restriction leads to cases where it is legal to assign Y to X but not to assign X to Y. Note that if X and Y are outermost lists with no list size specified, the datacomputer supplies a default size based on the space allocation. (use the LIST request with the %DESC option to find out what the default size is.)

AND

Rule 1c: If X and Y are STRUCTs or LISTs, then at least one container immediately enclosed in X must match, and have the same ident as, one container immediately contained in Y

OR

,X must be a STRing and Y a constant. A constant is an arbitrary string of characters. If they are enclosed by single quote marks, then is is an ASCII constant; a single or double quote mark may be included in such a string only by prefixing it with another double quote. The constant 'DON""T' represents the string DON'T. (This rule is included here for completeness and will be discussed later.) 6b1a

Padding and Truncation 6c

If two containers of type STR are used in an assignment, the matching rules do not require that their

sizes match. There are three cases: 6c1

The two sizes are equal. The string is assigned without change. 6c1a

In the assignment X=Y, the size of X is greater than that of Y. In this case, it is as if the string in Y is padded at the right-hand side to make it as long as X, before assignment is performed. If a fill character is specified in the description of X (i.e. if the parameter,F='a' or ,F=n is used in the CREATE request), then that character is used. Otherwise, a blank is used for ASCII strings and zero is used for non-Ascii data. 6c1b

The size of X is less than that of Y. The string contained in Y is truncated at the right-hand side to be as short as X, and the shortened string is then assigned. 6c1c

Examples 6d

Let us consider a few examples of the operation of the rules. Suppose we have ;

```
CREATE M FILE LIST (25), P=EOF RECOR D STR(10);
CREATE N TEMP PORT LIST (25), P=EOF RECORD STR(10) ;
M = N;
```
6d1

Where M is a FILE in which data read from the PORT N is to be stored in the datacomputer. The assignment M = N is legal because M is in WRITE mode and both M and N are open (opened by the CREATE statements). In addition, M and N match: their subcontainers have the same ident (RECORD), and matching descriptions. They satisfy rule 1.A, since the type is STR in both cases, and rules 1.B and 1.C do not apply to containers of type STR. 6d1a

The effect of this assignment is to read strings of length 10 from the PORT N, and to store them in the FILE M. If an attempt is made to store more than 25 strings in M, the dataComputer will complain, as space was allocated for only 25 strings. However, the 25 in the PORT description is ignored. 6d1b

A similar example, using the above description for M: 6d2

```
OPEN M APPEND;
CREATE O TEMP PORT LIST, P=EOF
```

```
        RECORD STR (,15), P=EOR ;
M = 0;                                                          6d2a
```

Each STRing in O is no more than 15 ASCII characters and ends with an EOR. Each one will be padded or truncated to 10 characters since M has fixed-length rather than punctuated STRings. 6d2a1

Now a more complex example. 6d3

```
CREATE FF FILE LIST (25), P=EOF
        PERSON STRUCT
           NAME STR (15)
           ADDRESS STR (20)
           CITY STR (10)
           STATE STR (2)
           ZIP STR (5)
           SOCSECNO STR (10)
           DEPENDENTS LIST (10) NAME STR (15)
        END;

... requests that store data in the FILE FF ...

CREATE PP PORT LIST, P=EOF
       PERSON STRUCT, P=EOR
           NAME STR (15)
           SOCSECNO STR (10)
       END;
      PP = FF ;                                                 6d3a
```

Here, the assignment PP = FF is legal because:PP is in WRITE mode, both FF and PP are open, and their descriptions match, Rule 1.A: the type of both FF and PP is LIST. Rule 1.B:PP is a PORT. Rule 1.C: the subcontainer PERSON immediately contained in FF has the same ident as PP.PERSON, and the two STRUCTs PERSON match. We determine this last fact by going round once again with the matching rules. 6d3a1

Rule 1.A: FF.PERSON and PP.PERSON have the same type, STRUCT. Rule 1.B does not apply to STRUCTs. Rule 1.C: a container immediately contained in FF.PERSON, FF.PERSON.NAME, has the same ident (NAME) and a matching description (STR (15)) as a container immediately enclosed by PP.PERSON, that is, PP.PERSON.NAME. 6d3a2

The effect of this assignment is to create a new instance of the struct PP.PERSON for each instance of PERSON in FF, and add it to the LIST PP (that is, output it through

the PORT PP). Each PERSON that is output contains only a selection of the data stored in FF: only the NAME and SOCSECNO. 6d3a3

If the situation here were reversed, that is, if FF were open in WRITE mode, and PP were in READ mode, the effect of the assignment

FF = PP;

would be to read data from the PORT PP and store it in the FILE FF. However, only the NAME and SOCSECNO would be available as data. The datacomputer handles this situation by assigning strings consisting only of blanks (the default since no fill character is specified in the description) to the unmatched STRs in the output LIST-member. Thus, ADDRESS, CITY, STATE, ZIP, and all 10 instances of DEPENDENTS.NAME would be blank in the FILE FF. 6d3a4

Very often, assignment at the level of outermost containers is all that a user's program will require of the datacomputer. An example would be a time-sharing monitor system, which might want to store backup files, large files, or infrequently-used files at the datacomputer ratherthan locally on (expensive) disc storage devices. Typically, such a monitor system would itself keep track of where various files resided, and move them from place to place over the ARPA network without burdening its users with the details of exactly where their files were stored. 6d4

For such an application, a directory might be set up with one node identifying the operating system that is doing the file storage. Subordinate to this node might be the user idents of its various time-sharing users whose files might be stored on the datacomputer. These user nodes, in turn, would have the file-names themselves as subordinate nodes; as bottom-most nodes, these would also be outermost containers and thus could store the data itself. 6d5

A directory of this sort would initially be set up by several CREATE requests; i.e.

```
CREATE SYS87;
  CREATE SYS87.SAM; CREATE SYS87.SMITH;
  CREATE SYS87.JONES; etc.
```

Then, whenever a particular file was to be moved to the

datacomputer, a directory node for that file would be set up by, for example,

```
CREATE SYS87,SMITH,FILE1 FILE LIST (999)
                A STR(80);
```

(describing a file with less than 1000 80-character records) and the file would be moved with an assignment statement specifying a PORT with a matching description, and the FILE FILE1, open in WRITE mode. Thus:

```
CREATE T TEMP PORT LIST A STR(80);

  FILE1 = T;                                                    6d6
```

Note that the two outermost containers FILE1 and T in the assignment statement FILE1 = T match each other. 6d7

In order to recover the file from the datacomputer when it is again needed, a PORT would be opened in WRITE mode with

```
CREATE T TEMP PORT LIST A STR(80);
 OPEN SYS87,SMITH,FILE1 READ;
T = FILE1 ;
```

and the reverse assignment would take place. 6d8

Selection of LIST Members 6e

In the examples given above there is one output LIST member for every input LIST member. Subsets of the input LIST member (i.e. the LIST on the right side of the =) may be specified by the use of WITH clause as an input-spec. For example consider the description

```
CREATE F FILE LIST P=EOF

    P STRUCT A STR(3) B STR(5) END;
```

and a matching PORT R. If only some of the P's on the LIST F were to be output -- those with the string A equal to the string '500' say -- one could specify

```
R = F WITH A EQ '500';
```

referring to the set of all members P of the LIST F that have the given property. Note that A is understood to refer to F.P.A; see the section on the context rules below for an

explanation. Quotes are used in the expression '500' to indicate that an ASCII string constant is intended. 6e1

In a WITH clause, the expressions one can use to choose certain LIST members, which are called Boolean expressions, must involve comparison of a container that is a STR or BYTE with a constant (like '500' in the example), using the comparison operators

EQ (equals)
NE (not equal to)
GT (greater than)
LT (less than)
GE (greater than or equal to)

and

LE (less than or equal to). 6e2

Combinations of comparisons with

OR, AND, NOT, and ANY

are also possible. In precedence of operators, ANY (see below) is highest; NOT is next in precedence, then AND, which is in turn higher than OR; parentheses may be used to affect the order of evaluation of these operators. 6e3

When using an input-spec, the name of the input LIST-member may be used instead of the name of the input LIST. (This is for consistency with the syntax of the FOR-loop, discussed below.) Thus,

```
R = F.P WITH A "LT" EQ '500';
```

is equivalent to the example above. Some sample input-specs are thus:

```
F.P
F.P WITH A EQ '500'
F WITH A EQ '500' AND B GT 'AZZZZ'
F.P WITH (A EQ '500' AND NOT B GT 'MONDA') OR
     (A EQ '600' AND B NE 'ZYYYY')
```
6e4

For ASCII containers, the operators GT, LT, etc. compare the ASCII codes for the given strings and the given strings must be of the specified length. This means that the character blank is less than the digits, which in turn are less than the letters. Consult a reference document for the

complete list of ASCII codes for all characters.

Also, while an input-spec like

F,P WITH A EQ '5'

is legal, it will not find any P's, since there are no A's with only one character. 6e5

Retrievals Using Inner List Members 6f

Consider a description like

```
G FILE LIST, P=EOF
  R STRUCT
     A STR (4)
     B STR (4)
     W LIST (20)
     WA STR (5)
  END
```

Each R has 20 Wa's, since R contains an inner list (W). An input-spec like

```
G,R WITH WA EQ 'ABCDE'
```

specifies all R's with at least one Wa with value 'ABCDE'. This may also be expressed as

```
G,R WITH ANY WA EQ 'ABCDE'
```

The former is called an implicit ANY and the latter, an explicit ANY.

6f1

The container WA can be used in boolean expressions such as

```
G,R WITH
      ANY (WA EQ 'MARCH' AND WA EQ '33103')
G,R WITH ANY
      (WA EQ 'MARCH' OR WA EQ 'WORD ')
G,R WITH ANY WA EQ '12345' AND B EQ 'CALI'
```

6f2

An ANY expression cannot be used within the object of another ANY expression (nested ANY's). 6f2a

In most cases, the explicit ANY is not required. However, consider the description:

```
FAMILIES  FILE LIST (100), P=EOF
  FAMILY STRUCT
        MOTHER STR (10)
        FATHER STR (10)
        CHILDREN LIST (10)
                CHILD STRUCT
                  NAME STR (,10), C=1
                  AGE STR (2)
                END
END;
```

6f3

The following expressions are not equivalent:

```
FAMILY WITH ANY (CHILD.NAME EQ 'ELLEN' AND
    CHILD.AGE EQ '21')
FAMILY WITH CHILD.NAME EQ 'ELLEN' AND
    CHILD.AGE EQ '21'
```

The latter case is interpreted as:

```
FAMILY WITH ANY CHILD.NAME EQ 'ELLEN'
    AND ANY CHILD.AGE EQ '21'
```

and refers to any FAMILY with an ELLEN who either is 21 or has a sibling who is 21. The former only refers to FAMILYs with a 21-year-old ELLEN.

6f3a

In all of these examples, the inner list is the second-level list. If there is a third level list, its members may not be used in a boolean expression. For example, given the description:

```
F FILE LIST R STRUCT
        A STR(1)
        L LIST (5)
                L1 LIST (5)
                        B STR (1)
END;
```

L1 is a third-level list, and so B cannot be used in a WITH expression. However, A may still be used in a WITH expression.

6f4

Retrievals Using Inverted Containers

6g

A STR may be inverted if it is contained in a FILE which is a LIST and if the LIST members are fixed-size. This is useful if the STR will be used often in a boolean

expression. Inversion is specified by "I=D" or "I=I" as follows:

```
CREATE F FILE LIST (0,100), P=EOF
    P STRUCT
     A STR (3), I=D
     Q LIST (10)
       B STR (5), I=I
 END;
```

The "I" of the above stands for inversion, the "=D" is used with members of outer lists, the "=I" with inner lists. 6g1

An inversion on the string A greatly increases the efficiency of retrieving sets of outermost-LIST members by the contents of the string A -- that is, retrieving subsets of the P's that are defined by their values of A. Retrieval by content based on a particular string is possible whether or not that string is inverted; only the efficiency is improved by the existence of an inversion on the string. 6g2

There is a certain cost associated with inversion, however, storage space must be allocated for a secondary data structure that the datacomputer uses for retrievals based on inverted strings. Updating a FILE takes longer when it is inverted, since the secondary data structure must be updated as well. Thus, the decision to invert a particular string will depend on the relative cost of increased retrieval time versus increased storage space, the frequency of retrieval based on the particular string, and other considerations. Appendix C contains further technical details concerning inversion. 6g3

Assignment with FOR 6h

Containers that are not outermost can also be used in assignment statements. With FOR, assignments that retrieve subsets of LIST-members may be performed, in contrast with assignment of outermost containers. FOR causes some set of datalanguage statements (usually assignment statements) to be executed several times, once for each member of a given set of LIST-members. 6h1

The syntax of the FOR-request is:

FOR <output-spec>, <input-spec> <body> END ;

The <input-spec> specifies a set of LIST-members to which the operations specified in the <body> are to be applied. A

new member of the LIST specified by the <output-spec> is created for each member of the input set processed. If the output-spec is omitted, the FOR-request generates no output. 6h1a

The input-spec must specify a set of LIST-members. The simplest kind of input-spec is just an entire LIST -- i.e. the set of all the LIST-members. However, the name of the LIST-member and not the LIST itself must be given. For example, if

```
CREATE F FILE LIST, P=EOF
          P STRUCT  A STR (3)  B STR (5)  END;
```

then F.P would be a legal input-spec, and would refer to the set of all P's in the LIST F. 6h2

A subset of the LIST-members may be specified by the use of a WITH clause in the input-spec. The input-spec on a FOR-loop looks like the input spec on the assignment of outermost containers (discussed above), except that the LIST-member must be named rather than the LIST. Thus

```
F.P WITH A EQ '500'
```

can be used in a FOR-loop, but not

```
F WITH A EQ '500'
```
6h3

The output-spec is an optional argument. Like the input-spec, it must be the name of a LIST-member. The LIST that contains the LIST-member specified by the output-spec is often called the output LIST. A new member is created and added to the output LIST for each execution of the FOR-body. 6h4

A FOR-loop may be loosely thought of as assignment between two LISTs. However, the descriptions of the members of the input and output LISTs need not match. Otherwise, the restrictions governing the input and output LISTs of a FOR are largely the same as those governing outermost LISTs used in assignment: 6h5

Both LISTs must be open or contained in open outermost containers. 6h5a

If the output LIST is an open outermost container, it must be in WRITE or APPEND mode. 6h5b

If the input LIST is not an outermost container,

the LIST that most immediately encloses it must be the input LIST of an enclosing FOR loop. 6h5c

Similarly, if the output LIST is not outermost, the LIST that most immediately encloses it must be the output LIST of an enclosing FOR. 6h5d

The operations that are legal in a FOR-body are assignment and another (nested) FOR. The assignment may be of the form

<name> = <constant> ;

where <name> refers to a container that is a STR (see matching rule number 3), or assignment may be of the form

<name> = <name>;

to transfer data from one container to another. If the latter is the case, then assignment is subject to the following restrictions 6h6

the restrictions specified in the matching rules above, 6h6a

the usual restriction that data can be transmitted into a container only if it is open in WRITE or APPEND mode, and 6h6b

the restriction that assignment must occur between objects, not sets of objects. 6h6c

In Version 0/10 of datalanguage, there are other restrictions governing the containers that can be referenced in the body of a FOR-loop. See Appendix E. 6h6d

Let us look at a few examples, and describe their operation in words. 6h7

With F a FILE as above, and

```
CREATE Q FILE LIST
    P STRUCT
        A STR (3)
        B STR (5)
    END;
...
OPEN F WRITE;
```

then

```
F = Q;
```

and

```
FOR F.P, Q.P
    F.P = Q.P ;
END;
```

have the same effect: a new member P is created and added to the LIST F. 6h7a

Likewise,

```
FOR F.P, Q.P WITH A EQ '500'
    F.P = Q.P;
  END;
```

has the same effect as

```
F = Q.P WITH A EQ '500'
```

6h7b

A final example: with FF.PERSON and PP.PERSON as given in the example for the matching rules,

```
F OR PP.PERSON, FF.PERSON WITH STATE EQ 'RI'
                OR STATE EQ 'CT'  OR STATE EQ 'MA'
                OR STATE EQ 'VT' OR STATE EQ 'NH'
                OR STATE EQ 'ME'
        PP.PERSON.NAME = FF.PERSON.NAME;
END;
```

will have the effect of outputting through the PORT PP, the NAMEs of all PERSONs in the FILE FF who live in New England; i.e. with STATE equal to one of the New England states.

6h7c

Chapter 6: Using the Datacomputer 7

Interacting with the Datacomputer 7a

We proceed now from the basics of the language itself, such as containers and assignment, to a broader view of how datalanguage might be employed by a user's program. We will discuss such matters as accessing the datacomputer, transmitting data to and from datalanguage PORTs, and various aids to the maintenance of data and FILE and PORT descriptions on the datacomputer. 7a1

Typically, datalanguage requests will be sent to the datacomputer by a user program residing on some computer on the ARPA network. All interaction between the user program and the datacomputer takes place over the network. 7a2

Information transmission over the network takes place along uni-directional paths. For a two-way conversation, two such paths are needed, one for transmission in each direction. The end of a transmission path is called a socket; a socket can be either a send (output) or receive (input) socket. Obviously, a transmission path requires a send socket at one end and a receive socket at the other. 7a3

A host computer is identified on the network either by a number or by an alphabetic name, like BBN-TENEX. A socket within a given host is identified by a number; send sockets have odd numbers and receive sockets even ones. For a connection to be opened, both hosts involved must request that it be opened. Likewise, after data transmission is complete, both hosts must close their ends of the connection. The period of time during which network connections are open between a user host and the datacomputer is called a session. 7a4

In the user program's dialogue with the datacomputer, the transmission in one direction consists largely of datalanguage requests, while messages from the datacomputer are sent in the other direction, to the user program. The sockets at the datacomputer that are used for these purposes are called the datalanguage input socket and the datalanguage output socket. The terms datalanguage input/output port are also used. These ports, like the PORTs that a user can create with datalanguage CREATE requests, are channels for the input and output of information. However, the purpose of the datalanguage ports is to receive datalanguage and transmit datacomputer messages; the purpose of a user PORT is to transmit or receive data. 7a5

The protocol by which a user program can set up datalanguage input and output sockets connected to its own output and input sockets is described in Appendix D of this document. 7a6

Synchronization 7b

Since use of the datacomputer typically involves the interaction of two programs at opposite ends of a communication network with a finite time delay, steps must be taken to ensure that the programs remain in synchrony with each other. If they do not, the user program might blithely go on sending datalanguage when the datacomputer expects data or might receive diagnostic messages when it expects a list of directory node names. 7b1

To avoid such problems, the datacomputer generates a variety of messages that keep the user program informed of what is going on. The messages fall into several categories: there are error messages, which will be discussed in a later section; informational messages, which can safely be ignored or merely logged by a user program; and synchronization messages, some of which at least must be processed by the user program to ensure proper communication. The first character of the message differs from category to category, allowing the user program easily to differentiate the various classes of message. 7b2

Prefix	Type of Message
?, -, or +	error message
;	informational message
.	synchronization message

7b2a

Other special characters may be added as datacomputer message-prefix characters in future versions. The letters, digits, tab, and space will never be used as message prefixes, however. 7b3

The datacomputer's messages all follow a common format, which includes the special header character just described, a letter and three digits that a program can use to identify the message, the date and time of the message's transmission, and a variable-length string of text that can be read by a human user. Specifically, the format is:

.X999 dd-mm-yy hhmm:ss (TAB) TEXT STRING (CR, LF)

where . represents the header character, X999 represents the message identifier (for example, I210), dd-mm-yy represents the day, month, and year (for example, 25-09-73),

hhmm:ss represents the time on a 24-hour clock in hours, minutes, and seconds, (TAB) represents a tab character, and (CR, LF) represents the carriage return, line feed characters that terminate the message. All alphabetic characters in the message are capitalized. Note that the message may be very long (too long to print on a 72-column printer, for instance), so a user program that processes datacomputer messages may have to format them to be readable.
In this manual, only the invariant parts of messages will be displayed; that is, the header character, the identifying letter and digits, and the message text. 7b4

To illustrate the use of synchronization messages in pacing interaction with the datacomputer, consider these two:

```
,I210 LAGC: READING NEW DL BUFFER
,J900 FCFINI: END OF SESSION
```
7b5

The first message, ,I210, is sent by the datacomputer over the datalanguage output socket, and hopefully received by the user program over an input socket, whenever the datacomputer is ready to accept datalanguage requests. The user program will in general respond to this message by transmitting a line of datalanguage. A line is some number of characters (currently there is an upper limit of about 2500) terminated by either the character sequence carriage return, line feed (ASCII codes 15, 12 octal) or the single character eol (37 octal). On a line may be one datalanguage request (terminated by a semicolon), several requests (each terminated by a semicolon), or a portion of a request. 7b6

In the first two cases, when the datacomputer receives the requests (and if they contain no errors) it will proceed to execute them, (typically generating messages and/or initiating data transfers as it does). Following execution, it will again send the ,I210 message signifying that it is again ready to receive datalanguage. In the third case, the datacomputer will continue to send ,I210 messages, prompting the user program for lines of datalanguage, until a complete request has been assembled; the request will then be executed as described above. 7b7

The second message, ,J900, is sent by the datacomputer at the end of a session. The user program may request that the session end by sending the datacomputer a control-Z (ASCII code 32 octal) in response to a ,I210 message. The datacomputer responds to control-Z by executing an end of session procedure,

which involves closing any open containers, deleting TEMP PORTs, and sending the .J900 message. The user program may then close its network connections with the datacomputer. 7b8

Synchronization after an error is discussed in the section entitled Error Messages below. 7b9

Transmitting Data through the Datalanguage Ports 7c

Often, a user program will need to send data over the network to be stored at the datacomputer, or to process data that it receives from the datacomputer. If all of the data is described as ASCII, then this may be done by using the datalanguage input or output port. 7c1

To reference data that he or she will transmit through the datalanguage input socket, the user need only open a PORT and use it on the right-hand side of an assignment in datalanguage. When the assignment is executed, data will be accepted through the datalanguage input port and assigned to whatever container appears on the left side of the request. 7c2

Similarly, to output data through the datalanguage output socket so that it can be picked up by the user program, all that is needed in datalanguage is a PORT used on the left-hand side of an assignment. Any data assigned to that container will be transmitted out through the datalanguage output port over the network. 7c3

Of course, performing this feat requires the use of more synchronizatio messages. To treat the data-input case first:

.I231 OCPBO: (DEFAULT) INPUT PORT OPENED

.I251 OCPBC: (DEFAULT) INPUT PORT CLOSED 7c4

After the user program has sent the datalanguage assignment request that references the open input PORT, the datacomputer will transmit the .I231 message over the datalanguage output port. The message signals that input data is now expected through the datalanguage input port, and the user program should send the data. Data transmission is terminated by a control-Z character, which causes the datacomputer to send the .I251 message confirming that data transmission is finished. The next synchronization message will be .I210, a request for more datalanguage. 7c5

The synchronization procedure governing data output through the

datalanguage output port is similar. The messages are

,I241 OCSOP: (DEFAULT) OUTPUT PORT OPENED

,I261 OCSCL: (DEFAULT) OUTPUT PORT CLOSED 7c6

When the assignment statement is executed which requests that data be output through the datalanguage port, the datacomputer first sends ,I241, followed by the requested data, followed in turn by ,I261. The datacomputer does not output a control-Z at the end of the data. The user program can use these messages to separate out the data from all other information. 7c7

Opening a Secondary Port 7d

Instead of a datalanguage port, an additional network connection or secondary port can be used for transmitting data. Non-ASCII data, including an ASCII STR with a preceding count or a non-ASCII delimiter, must be transmitted over a secondary port. The CONNECT request sets up the secondary port. 7d1

The CONNECT request names an open PORT, and gives a host (that is, a computer on the network) and socket number to which that PORT is to refer. As mentioned above, if a CONNECT request is never executed for a PORT, it will refer to the socket from which the user program transmits datalanguage (if it is a READ PORT) or the socket at which the user program receives the datacomputer's messages (a WRITE or APPEND PORT). The form of the CONNECT request is

CONNECT <pathname> TO <address> ;

where <pathname> is the node name, complete name (i.e. starting with %top) or simple login name (i.e. starting immediately subordinate to the login node) of an open PORT, and <address> can have several forms. 7d2

It can be one of

<socket-no> the decimal number of a socket at the user's host computer.

<host-no> <socket-no> where <host-no> is the decimal number of a computer on the ARPA network

'<host-name>' <socket-no> where <host-name> is the host computer's TENEX alphabetic name

<host-name> <socket-no> where <host-name> is the host computer's TENEX alphabetic name (such as 'CCA')

OR

'<local-file-designator>' This last form of <address> does not refer to the network, but is included here for completeness, <local-file-designator> is a TENEX file designator that refers to a file at the datacomputer site. 7d3

A CONNECT may be executed any time the PORT is open, but it does not actually establish the network connection. Those connections are established, used, and then closed again during the execution of an assignment statement in datalanguage, and CONNECT merely sets up the socket address to be used when the PORT is later referenced in an assignment. 7d4

A DISCONNECT request may be used to cause a CONNECTed PORT to refer once again to the datalanguage input or output port.

```
DISCONNECT <pathname> ;
```

Two CONNECT requests may be issued for the same PORT without an intervening DISCONNECT. 7d5

Additional synchronization messages are generated at the time a CONNECTed PORT is used in an assignment statement. These messages are

```
,I230 OCPBO: OPENING INPUT PORT
;I239 OCPBO: INPUT PORT OPENED
,I250 OCPBC: CLOSING INPUT SOCKET
,I240 OCPOO: OPENING OUTPUT PORT
;I249 OCPOO: OUTPUT PORT OPENED
,I260 OCPOC: CLOSING OUTPUT SOCKET
```

7d6

When a CONNECTed PORT is used on the right-hand side of an assignment (that is, in READ mode), the ,I230 message is sent over the datalanguage output port. This signals the user program that the datacomputer is attempting to open a network connection to the host and socket specified by the CONNECT request for the PORT. The user program should thus open its end of the connection itself (if it is a connection to a different socket on the user program's own host) or ensure that the third

host opens its end of the connection at this time (if it is a connection to another host on the network). 7d7

The ;I239 message indicates that indeed the network connection was opened correctly. After this message is received, data can be transmitted, terminated by closing the network connection. Once the connection is closed, the datacomputer sends .I250 over the datalanguage output port, signaling the user program that use of the secondary network connection is complete. The .I250 may precede or follow the closing of the connection on the user's side. The messages for output PORTs work similarly, with .I240 signaling that the output network connection is being opened, ;I249 that the connection is opened, and .I260 that output is complete and the connection is being closed. 7d8

If there are errors in the data, other messages will be sent before the .I250 or .I260 message. This would be the case, for example, if the data does not match the description. 7d9

A user program can interrupt the datacomputer's transmission of data; see Appendix D for details. 7d10

The form CONNECT <pathname> TO <local-file-designator>; may be useful to those with large amounts of data to send to the datacomputer. In some cases, the shipment of magnetic tapes by air-freight produces higher bit rates than sending the data over the network; the magnetic tape may then be addressed from datalanguage as a local file. Contact CCA for information on this procedure. 7d11

Error Messages 7e

Datacomputer error messages will in general be seen by a human user, although they have header characters which make them potentially processable by a smart user program. Error messages fall into several categories, distinguished by their first character 7e1

First Character	Meaning
?	indicates a datacomputer or system bug. A user program should rarely see one of these.

Examples:

```
?U000 TRDN; NODE CHAIN SNAFU
?U000 DKWR; DISK I/O WRITE ERROR
```

- indicates a user error -- typically bad datalanguage, data, or i/o handling. A debugged user program should rarely see one of these.

Examples:

-U000 LPNM: FORARG NOT DIRECT LIST MEMBER

-I246 OCSOP: CAN'T OPEN OUTPUT PORT (BAD CONNECT ARGS?)

\+ indicates a circumstantial error, such as a file's being busy, or an error which is due to current datacomputer limitations.

Examples:

+U000 OCDOP: CAN'T OPEN FILE (SOMEBODY ELSE UPDATING?)

+L000 DHIN: DESCRIPTOR TOO LARGE 7e2

After the datacomputer generates one or more error message, it follows a special procedure to resynchronize itself with the user. This procedure involves waiting for a special character, control-L or form feed (ASCII 14 octal), to be transmitted by the user. That is, after the error message the datacomputer sends

,I220 LAEB: LOOKING FOR CONTROL-L 7e3

This is repeated for each line of input it receives on the datalanguage input port until the user sends a control-L character. Following receipt of a control-L, ,I210 will again be sent and datalanguage requests again processed. 7e4

More severe action must be taken following certain system or ?-type errors. One of the following synchronization messages may be generated:

,J151 FCERRH: RESTARTING THE REQUEST HANDLER
,J140 FCREIN: REINITIALIZING USER JOB
,J910 FCERRH: CRASHING JOB 7e5

The ,J151 message indicates that TEMP PORTs have been deleted; otherwise, the status of the session remains the same (PORTs and FILEs will still be open, etc.). This message will usually be followed by ,I220, a request for control-L. The ,J140 message is more serious. The user's job is completely

reinitialized, leaving his status the same as when the session was begun. This message will also be followed by .I220. The .J910 message indicates a condition so severe that the datacomputer does not know how to recover. The user's job is crashed and the datalanguage network connections closed. That is, the session is forcibly ended. If this happens, and also if the user's network connections to the datacomputer are accidentally broken, the datacomputer will do its best to close his open PORTs and FILEs in an orderly manner. However, if the user was in the process of transmitting data into a FILE, the last few thousand characters of data his program sent may have been lost in transit and not incorporated into the FILE. 7e6

Not much in general can be said about handling ? or - errors, except that a human user will have to read and interpret the text of the error message in each case, and (in the case of - errors) correct the datalanguage he is having his program send. 7e7

+ errors, on the other hand, could be processed by a user program. The most reasonable thing to do in many cases is to wait five minutes and retry the datalanguage request that caused the error. For example, a FILE which was busy (i.e. in use by someone else) may be free by that time, so the second attempt to use it may be successful. 7e8

Messages beginning with +L are an exception to this, in that the appropriate time to wait may be several weeks instead of minutes. Such messages indicate limitations of the current datacomputer system, such as limitations imposed by internal table sizes. A new version of the datacomputer may remove many of these limitations. Realistically, this means that +L messages are like - messages in that a program probably could not handle them. 7e9

Appendix A: Summary of Datalanguage Syntax 8

The following is the complete BNF (Backus Normal Form) specification of datalanguage syntax for version 0/10 of the datacomputer. 8a

Requests 8a1

Directory Requests 8a1a

```
<request>  ::=  LOGIN <login body> ;
<request>  ::=  CREATE <create body> ;
<request>  ::=  DELETE <delete body> ;
<request>  ::=  OPEN <open body> ;
<request>  ::=  CLOSE <close body> ;
<request>  ::=  CONNECT <connect body> ;
<request>  ::=  DISCONNECT <disconnect body> ;
<request>  ::=  MODE <mode body> ;
<request>  ::=  CREATEP <createp body> ;<request>  ::=
DELETEP <deletep body> ;
<request>  ::=  LIST <list body> ;
```

8a1a1

Data Transfer Requests 8a1b

```
<request>  ::=  <direct assignment> ;
<request>  ::=  <for loop> ;
```

8a1b1

Pathnames 8a1c

```
<pathname>  ::=  <complete pathname>
<pathname>  ::=  <simple complete pathname>
<pathname>  ::=  <login pathname>
<pathname>  ::=  <simple login pathname>
<pathname>  ::=  <open node name>
```

8a1c1

```
<node name>  ::=  <identifier>
<node name>  ::=  <identifier> ( <password string> )
<password string>  ::=  <string constant>
<simple node name>  ::=  <identifier>
```

8a1c2

```
<complete pathname>  ::=  %TOP , <node name>
<complete pathname>  ::=
   <complete pathname> , <node name>
```

8a1c3

```
<simple complete pathname>  ::=
   %TOP , <simple node name>
<simple complete pathname>  ::=
   <simple complete pathname> , <simple node name>
```

8a1c4

```
<login pathname>  ::=  <node name>
<login pathname>  ::=  <login pathname> , <node name>          8a1c5

<simple login pathname>  ::=  <simple node name>
<simple login pathname>  ::=
   <simple login pathname> , <simple node name>                8a1c6

<open node name>  ::=  <simple node name>                      8a1c7

<node pathname>  ::=  <complete pathname>
<node pathname>  ::=  <login pathname>                         8a1c8

<open pathname>  ::=  <simple complete pathname>
<open pathname>  ::=  <simple login pathname>
<open pathname>  ::=  <open node name>                         8a1c9
```

Requests 8a1d

```
<login body>  ::=  %TOP
<login body>  ::=  <node pathname>                             8a1d1

<create body>  ::=  <simple node name>
<create body>  ::=
   <node pathname> , <simple node name>
<create body>  ::=  <data description>
<create body>  ::=
   <node pathname> , <data description>                        8a1d2

<delete body>  ::=  **
<delete body>  ::=  <login pathname>
<delete body>  ::=  <login pathname> , **                      8a1d3

<open body>  ::=  <node pathname>
<open body>  ::=  <node pathname> <mode>                       8a1d4

<close body>  ::=  %OPEN
<close body>  ::=  <open pathname>                             8a1d5

<connect body>  ::=
   <open pathname> <tenex file specification>
<connect body>  ::=
   <open pathname> <network specification>
<tenex file specification>  ::=  <string constant>
<network specification>  ::=  <socket number>
<network specification>  ::=
   <host specification> <socket number>
<socket number>  ::=  <integer constant>
<host specification>  ::=  <integer constant>
```

```
<host specification>  ::=  <identifier>
<host specification>  ::=  <string constant>                    8a1d6

<disconnect body>  ::=  <open pathname>                         8a1d7

<mode body>  ::=  <open pathname> <mode>
<mode>  ::=  READ
<mode>  ::=  WRITE
<mode>  ::=  APPEND
<mode>  ::=  WRITE DEFER
<mode>  ::=  APPEND DEFER                                       8a1d8

<createp body>  ::=  <node pathname>
<createp body>  ::=
   <node pathname> <privilege tuple specification>
<privilege tuple specification>  ::=
   <privilege tuple option>
<privilege tuple specification>  ::=
   <privilege tuple specification>
         <privilege tuple option>
<privilege tuple option>  ::=  , U = <user identity>
<privilege tuple option>  ::=  , H = <host identity>
<privilege tuple option>  ::=  , S = <socket identity>
<privilege tuple option>  ::=  , P = <password string>
<privilege tuple option>  ::=
   , G = <grant privilege list>
<privilege tuple option>  ::=
   , D = <deny privilege list>
<privilege tuple option>  ::=
   , N = <privilege tuple index>
<user identity>  ::=  **
<user identity>  ::=  <user node>
<user identity>  ::=  <user node set>
<user identity>  ::=  <user node> , **
<user identity>  ::=  <user node set> , **
<user identity>  ::=
   <user node> , <user node set> , **
<user node>  ::=  <identifier>
<user node>  ::=  <user node> , <identifier>
<user node set>  ::=  *
<user node set>  ::=  <user node set> , *
<host identity>  ::=  ANY
<host identity>  ::=  LOCAL
<host identity>  ::=  <integer constant>
<socket identity>  ::=  ANY
<socket identity>  ::=  <integer constant>
<grant privilege list>  ::=  <grant privilege>
<grant privilege list>  ::=
   <grant privilege list><grant privilege>
```

```
<grant privilege>  ::=  C
<grant privilege>  ::=  L
<grant privilege>  ::=  R
<grant privilege>  ::=  W
<grant privilege>  ::=  A
<deny privilege list>  ::=  <deny privilege>
<deny privilege list>  ::=
   <deny privilege list><deny privilege>
<deny privilege>  ::=  R
<deny privilege>  ::=  W
<deny privilege>  ::=  A
<privilege tuple index>  ::=  <integer constant>                8a1d9

<deletep body>  ::=
   <node pathname> <privilege tuple index>                      8a1d10

<list body>  ::=  <list node set>
<list body>  ::=  <list node set> <list option>
<list node set>  ::=  %TOP
<list node set>  ::=  %OPEN
<list node set>  ::=  *
<list node set>  ::=  **
<list node set>  ::=  <open node name>
<list node set>  ::=  <node pathname>
<list node set>  ::=  <node pathname> . *
<list node set>  ::=  <node pathname> . **
<list option>  ::=  %NAME
<list option>  ::=  %DESCRIPTION
<list option>  ::=  %DESC
<list option>  ::=  %SOURCE
<list option>  ::=  %ALLOCATION
<list option>  ::=  %ALLOC
<list option>  ::=  %PRIVILEGE
<list option>  ::=  %PRIV                                       8a1d11
```

Data Description 8a1e

```
<datatype>  ::=  <compound datatype>
<datatype>  ::=  <simple datatype>
<datatype>  ::=  <string>                                       8a1e1

<compound datatype>  ::=  LIST
<compound datatype>  ::=  <structure>
<structure>  ::=  STRUCTURE
<structure>  ::=  STRUCT                                        8a1e2

<simple datatype>  ::=  BYTE
<simple datatype>  ::=  <integer>
```

```
<integer>  ::=  INTEGER
<integer>  ::=  INT                                                    8a1e3

<string>  ::=  <string type>
<string>  ::=  <string type> <string interpretation>
<string type>  ::=  STRING
<string type>  ::=  STR
<string interpretation>  ::=  ASCII
<string interpretation>  ::=  ASCII8
<string interpretation>  ::=  BYTE
<string interpretation>  ::=  INT
<string interpretation>  ::=  INTEGER                                  8a1e4

<data description>  ::=
<simple node name> <function>
<outermost description>
<function>  ::=  FILE
<function>  ::=  PORT
<function>  ::=  TEMPORARY PORT
<function>  ::=  TEMP PORT
<outermost description>  ::=  LIST <description>
<outermost description>  ::=
LIST <compound datatype options> <description>
<outermost description>  ::=  <string>
<outermost description>  ::=  <string> <string options>
<outermost description>  ::=  <description>                            8a1e5

<description>  ::=
LIST <dimension> <description>
<description>  ::=
LIST <dimension> <compound datatype options>
<description>
<description>  ::=
<structure> <descriptions> END
<description>  ::=
<structure> <compound datatype options>
<descriptions> END
<description>  ::=  BYTE
<description>  ::=  BYTE <simple datatype options>
<description>  ::=  <integer>
<description>  ::=  <integer> <simple datatype options>
<description>  ::=  <string> <dimension>
<description>  ::=
<string> <dimension> <string options>
<descriptions>  ::=  <description>
<descriptions>  ::=  <descriptions> <description>                       8a1e6

<description option>  ::=  <inversion option>
<description option>  ::=  <byte size option>
```

```
<description option>  ::=  <filler option>
<description option>  ::=  <variable length option>
<inversion option>  ::=  , I = D
<inversion option>  ::=  , I = I
<byte size option>  ::=  , B = <integer constant>
<filler option>  ::=  , F = <integer constant>
<filler option>  ::=  , F = '<nonquote character>'
<variable length option>  ::=  , C = 1
<variable length option>  ::=  , P = EOF
<variable length option>  ::=  , P = EOB
<variable length option>  ::=  , P = EOR
<variable length option>  ::=  , D = <integer constant>
<variable length option>  ::=
, D = '<nonquote character>'                                    8a1e7

<compound datatype options>  ::=
<compound datatype option>
<compound datatype options>  ::=
<compound datatype options>
<compound datatype option>
<compound datatype option>  ::=  <byte size option>
<compound datatype option>  ::=  <filler option>
<compound datatype option>  ::=
<variable length option>                                        8a1e8

<simple datatype options>  ::=
<simple datatype option>
<simple datatype options>  ::=
<simple datatype options>
<simple datatype option>
<simple datatype option>  ::=  <inversion option>
<simple datatype option>  ::=  <byte size option>
<simple datatype option>  ::=  <filler option>                  8a1e9

<string options>  ::=  <string option>
<string options>  ::=  <string options> <string option>
<string option>  ::=  <inversion option>
<string option>  ::=  <byte size option>
<string option>  ::=  <filler option>
<string option>  ::=  <variable length option>                  8a1e10

<dimension>  ::=  ( <integer constant> )
<dimension>  ::=  ( , <integer constant> )
<dimension>  ::=
( <integer constant> , <integer constant> )                     8a1e11
```

Data Transfer 8a1f

```
<data reference>  ::=  <identifier>
```

```
<data reference>  ::=  <data reference> , <identifier>
<constant>  ::=  <string constant>
<constant>  ::=  <integer constant>
<assignment>  ::=  <data reference> = <data reference>
<assignment>  ::=  <data reference> = <constant>                    8a1f1

<direct assignment>  ::=  <assignment>
<direct assignment>  ::=  <implicit for loop>
<implicit for loop>  ::=  <assignment> <qualifier>                  8a1f2

<for loop>  ::=  FOR <input> <for body> END
<for loop>  ::=  FOR <input> <qualifier> <for body> END
<for loop>  ::=  FOR <output> , <input> <for body> END
<for loop>  ::=
FOR <output> , <input> <qualifier> <for body> END
<input>  ::=  <data reference>
<output>  ::=  <data reference>
<for body>  ::=  <for loop>
<for body>  ::=  <for loop> ;
<for body>  ::=  <assignment list>
<for body>  ::=  <assignment list> ;
<assignment list>  ::=  <assignment>
<assignment list>  ::=
<assignment list> ; <assignment>                                    8a1f3

<qualifier>  ::=  WITH <boolean expression>                         8a1f4

<boolean expression>  ::=  <relational expression>
<boolean expression>  ::=  ( <boolean expression> )
<boolean expression>  ::=  NOT <boolean expression>
<boolean expression>  ::=  ANY <boolean expression>
<boolean expression>  ::=
<boolean expression> AND <boolean expression>
<boolean expression>  ::=
<boolean expression> OR <boolean expression>                        8a1f5

<relational expression>  ::=
<data reference> <comparison operator>
<data reference>
<relational expression>  ::=
<data reference> <comparison operator> <constant>
<comparison operator>  ::=  EQ
<comparison operator>  ::=  NE
<comparison operator>  ::=  GT
<comparison operator>  ::=  GE
<comparison operator>  ::=  LT
<comparison operator>  ::=  LE                                      8a1f6
```

Lexical Items 8a1g

```
<lexical item>  ::=  <identifier>
<lexical item>  ::=  <integer constant>
<lexical item>  ::=  <string constant>
<lexical item>  ::=  <autonomous character>                    8a1g1

<identifier>  ::=  <letter>
<identifier>  ::=  %
<identifier>  ::=  <identifier> <letter>
<identifier>  ::=  <identifier> %
<identifier>  ::=  <identifier> <digit>                        8a1g2

<integer constant>  ::=  <digit>
<integer constant>  ::=  <integer constant> <digit>            8a1g3

<string constant>  ::=  '<string constant body>'
<string constant body>  ::=  <nonquote character>
<string constant body>  ::=
<string constant body> <nonquote character>                    8a1g4
```

Character Set 8a1h

```
<letter>  ::=  A
<letter>  ::=  B
.................
<letter>  ::=  Z
<letter>  ::=  a
<letter>  ::=  b
.................
<letter>  ::=  z                                               8a1h1

<digit>  ::=  0
<digit>  ::=  1
.............
<digit>  ::=  9                                                8a1h2

<nonquote character>  ::=  <letter>
<nonquote character>  ::=  %
<nonquote character>  ::=  <digit>
<nonquote character>  ::=  <autonomous character>
<nonquote character>  ::=      (space)
<nonquote character>  ::=      (horizontal tab == HT)
<nonquote character>  ::=  "'
<nonquote character>  ::=  ""                                  8a1h3

<separator>  ::=      (space)
<separator>  ::=      (horizontal tab == HT)
<separator>  ::=  <eol>
<eol>  ::=      (end of line == octal 37)
<eol>  ::=  <carriage return> <line feed>
```

```
<carriage return>  ::=      (carriage return == CR)
<line feed>  ::=      (line feed == LF)                          8a1h4

<autonomous character>  ::=  !
<autonomous character>  ::=  #
<autonomous character>  ::=  $
<autonomous character>  ::=  &
<autonomous character>  ::=  (
<autonomous character>  ::=  )
<autonomous character>  ::=  *
<autonomous character>  ::=  +
<autonomous character>  ::=  ,
<autonomous character>  ::=  -
<autonomous character>  ::=  .
<autonomous character>  ::=  /
<autonomous character>  ::=  :
<autonomous character>  ::=  ;
<autonomous character>  ::=  <
<autonomous character>  ::=  =
<autonomous character>  ::=  >
<autonomous character>  ::=  ?
<autonomous character>  ::=  @
<autonomous character>  ::=  [
<autonomous character>  ::=  \
<autonomous character>  ::=  ]
<autonomous character>  ::=  ^
<autonomous character>  ::=  _
<autonomous character>  ::=  `
<autonomous character>  ::=  {
<autonomous character>  ::=  |
<autonomous character>  ::=  }
<autonomous character>  ::=                                     8a1h5
```

Directory 8a2

Notes 8b

Character codes are 7 bit ASCII. 8b1

Separators are always permitted between lexical items, except between grant privileges, between deny privileges, and inside string constants. 8b2

Comments may be inserted wherever separators are allowed. Comments begin with '/*' and end with '*/' (e.g., /* THIS IS A COMMENT */). 8b3

<carriage return> and <line feed> may only appear together in that order (as an <eol>). Otherwise they

are treated as control characters, which are rejected,

8b4

Appendix B: Reserved Words 9

```
AND
ANY
ASCII
ASCII8
BYTE
CLOSE
CONNECT
CREATE
CREATEP
DELETE
DELETEP
DISCONNECT
END
EQ
FILE
FOR
GE
GT
INT
INTEGER
LE
LIST
LOGIN
LT
MODE
NE
NOT
OPEN
OR
PORT
STR
STRING
STRUCT
STRUCTURE
WITH
%OPEN
%TOP
```

9a

Appendix C: Inversion: Technical Considerations 10

An inversion is a secondary data structure that the datacomputer can use to improve its efficiency in retrieving data by content from a datalanguage FILE. Specifically, an entry in the inversion is constructed for every STR with the inversion attribute. For each data value which occurs for the STR, the inversion contains pointers to all the records in the FILE for which that STR contains that value. 10a

For example, if

```
CREATE PEOPLE FILE LIST
      PERSON STRUCT
          NAME STR (15)
          SOCSECNO STR(9),I=D
          SEX STR (1) /* 'M' OR 'F'*/,I=D
          ZIP STR(5),I=D
      END;
```

10a1

then the data structure for the inversion on SEX contains pointers to all instances of PERSONs with SEX equal to 'F', and similarly for 'M'. Thus, evaluation of a simple FOR input-spec like

```
FOR ... , PEOPLE,PERSON WITH SEX EQ 'M'
```

would be quick and simple, and would require only a read of the inversion, not any reading of the FILE PEOPLE itself. 10b

An inversion is not only constructed automatically by the datacomputer when the FILE is loaded with data, but is automatically maintained (updated) whenever information in the FILE is updated. 10c

Unfortunately, even if an inversion for the appropriate STR exists, the datacomputer cannot always use it for the evaluation of input-specs, and must sometimes resort to time-consuming searches of the FILE. In particular, the inversion can be used only when the STR is compared with a constant using the operators EQ and NE. That is,

```
      PEOPLE,PERSON WITH ZIP EQ '02138' OR ZIP EQ
'02139'
                    OR ZIP EQ '02140' OR ZIP EQ
```

'02141'

can be evaluated directly from the inversion. However,

```
PEOPLE.PERSON WITH ZIP GE '02138'
        AND ZIP LE '02141'
```

while it still can be evaluated, cannot take advantage of the inversion and so would be much less efficient datalanguage. 10d

Furthermore, when the STR is a member of an inner LIST, only the oehrator EQ can be evaluated using the inversion. A sequential search is used for evaluating NE. 10e

Complex Boolean expressions, those involving several comparisons, fall into three classes: those with all comparisons evaluable from the inversion, those containing no comparisons evaluable from the inversion, and those which mix the two kinds of comparisons. The first two classes pose no problem; the datacomputer will use the inversion to evaluate expressions in the first category, and not for expressions in the second category. 10f

For mixed expressions, the datacomputer will use the inversion as much as it can. For the present, this can be stated as follows: if the Boolean expression is of the form

```
<expr> AND <expr> AND ...
```

(where <expr> is an arbitrary Boolean expression, in parentheses if it contains OR) then the datacomputer will separate the <expr>s into those that can be completely evaluated from the inversion and those that cannot, and will process those that can use the inversion first. The <expr>s that cannot use the inversion are evaluated by an exhaustive search of the set of records selected by the earlier <expr>s. 10g

For an example, take the above FILE, PEOPLE. Suppose a list of all males with ZIP GT '02000' were desired. ZIP is indeed inverted, but since the operator GT is involved, the evaluation of that part of the Boolean expression cannot use the inversion. As a result, in

```
FOR ... , PEOPLE.PERSON WITH ZIP GT '02000'
 AND SEX EQ 'M'
```

the datacomputer will first use the inversion to find the set of all PERSONs with SEX EQ 'M', and only this smaller set of PERSONs would be searched for the desired ZIPs. 10g1

A more difficult example: consider the problem of retrieving all the records for events that occurred between 10:05 on the 25th and 15:07 of the 30th from a FILE that is inverted on DAY but not on TIME. A straightforward way to do this is

```
... WITH (DAY EQ '25' AND TIME GT '10:05')
        OR (DAY EQ '26') OR (DAY EQ '27') OR
...
        OR (DAY EQ '30' AND TIME LT '15:07')
```

but this is quite inefficient: the inversion cannot be used at all, for this Boolean expression is mixed and is not set up as a series of terms connected by AND. The best way to express this condition is

```
... WITH (DAY EQ '25'  OR DAY EQ '26' OR ... OR
DAY EQ '30')
      AND (DAY NE '25' OR TIME GT '10:05')
       AND (DAY NE '30' OR TIME LT '15:07')
```
10g2

In this case, only records for the correct six days are retrieved by the first term, so only they need to be searched through for the evaluation of the second and third terms. 10g3

Future versions of the datacomputer will automatically optimize mixed Boolean expressions, freeing the user from this task. 10h

The computation of the space requirements for an inversion is best left to the datacomputer's operational staff at CCA, who should be contacted by any user interested in setting up a data file with an inversion. 10i

Appendix D: Network Interaction with the Datacomputer 11

The procedure for establishing network connections with the datacomputer is that documented in J. Postel, Official Telnet - Logger Initial Connection Protocol, NIC 7103, 15 June 1971. The following is a simplified, informal description of that procedure. 11a

The datacomputer listens for connections on a well-advertised socket, currently number 103 (octal) at CCA, host number 37 (octal). This is an odd-numbered or send socket. The user program wishing to use the datacomputer will address this socket from a socket on his own host computer -- say from socket number U. U must, of course, be an even number or a receive socket. The user program should read one 32-bit byte of information over this connection and then immediately close it (leaving socket CCA-103 free for other users). This byte of information is a socket number at the datacomputer -- say socket D. D will be an even number. 11b

The last step is the opening of two network connections, the permanent datalanguage connections. They are

from D+1 at CCA to U+2 at the user host
and from U+3 at the user host to D at CCA. 11c

Note that U+2 is even (since U is) and D+1 is odd -- this is the datalanguage output socket. Also, U+3 is odd, and D is even: the datalanguage input socket. These connections will remain in effect until the end of the datalanguage session. 11d

The byte size of the permanent datalanguage connections is 8 bits. The datacomputer sends, and expects to receive, 7-bit ASCII characters right-justified in 8-bit bytes. 11e

Two special network control signals, INS and INR, may be used to interrupt the datacomputer. INS, for interrupt the sender, may be sent at any time during the processing of a request and stops data output from the current request. No error message or other acknowledgement will be generated; the output simply stops. INS might be useful to a program which receives output from the datacomputer and displays it to a human operator sitting at a teletype; at the request of the user, the program could send INS to stop an overly-long

printout,
INR, 11f

INR, for interrupt the receiver, performs all the functions of INS. In addition, compilation or any other processing that is under way when INR is received will be aborted, possibly generating an error message and a request, for control-L. INR thus requests a more immediate halt than does INS. 11g

Appendix E: Implementation Restrictions

12

A number of datalanguage restrictions specific to Version 0/10 are collected here for ready reference. Note that some of these restrictions have been mentioned in the body of this manual, while others have not.

12a

There is a restriction on the containers that can be referenced in the body of a FOR-loop. Consider the following example:

```
CREATE FF FILE LIST
    PERSON STRUCT
        NAME STR (15)
        ADDRESS STR (20)
        CITY STR (10)
        STATE STR (2)
        ZIP STR (5)
        SOCSECNO STR (10)
        DEPENDENTS LIST (10)
            NAME STR (15)
    END;

CREATE PP PORT LIST
    PERSON STRUCT
        NAME STR (15)
        SOCSECNO STR (15)
    END;
```

12b

To output all the DEPENDENTS.NAMEs from the file FF, together with the SOCSECNO of the PERSON whose DEPENDENTS they were,

```
FOR PP.PERSON,FF.PERSON
     NAME=NAME;
     SOCSECNO=SOCSECNO;
       END;
```

This example as written will work in datalanguage 0/10. However, SOCSECNO occurred after DEPENDENTS in the description of FF.PERSON, the request would fail due to a compiler restriction.

12c

When an inner FOR-loop is processing a LIST which occurs within a STRUCT, references may be made in the

body of that FOR to objects which occur before that LIST in the STRUCT, but not after the LIST. 12d

There are certain cases of assignment involving inner LISTs which the compiler in Version 0/10 cannot handle. For example, given two structures of the following format:

```
L1 FILE LIST
    S1 STRUCT
        A1 STR (8)
        A2 LIST (4)
           B2 STR (6)
    END;
```

and

```
L2 PORT LIST
    S1 STRUCT
        A1 STR (8)
        A2 LIST (4)
           B2 STR (6)
    END;
```

the following FOR-loop will not work:

```
FOR L1,S1,L2,S2
    FOR A2,B2,A2,B2
        S1=S1
    END
END;
```

The A2 lists are in use by the inner FOR-loop (FOR A2,B2,A2,B2) when the assignment S1=S1 is encountered. The datacomputer expands S1=S1 internally into:

```
A1=A1
FOR L1,S1,A2,B2,L2,S1,A2,B2
    B2=B2
END;
```

This constitutes a second use of the A2 lists, which cannot be handled. 12e

In Version 0/10 of datalanguage, there is one general restriction on sequences of nested FOR-loops, which can be stated as follows: 12f

Sequences of nested FOR-loops are restricted to be

a number (possibly 0) of FOR-loops without output LISTs, followed by an arbitrary number, at least 1, of FOR-loops with output LISTs. 12g

For example,

```
   FOR A                  FOR A                FOR A
     FOR B,C                FOR B
(ASSIGNMENT)
       (ASSIGNMENT)           FOR C,D          END;
     END;                       (ASSIGNMENT)
END;                         END;
                              END;
```

The first two examples are legal, whereas the third is not. 12g1

A FOR-loop with no output LIST can contain only one datalanguage statement as the FOR-body, not a series of statements. Because of restriction 2, that one statement must be a FOR.
This does not apply to a FOR with an output LIST. 12h

The only comparison operators which can be evaluated from an inversion are EQ and NE. All other comparison operators must be evaluated by a linear search through a set of records. If the container being compared is a member of an inner list, only the EQ comparison operator can be evaluated from an inversion. 12i

It is impossible to assign members of a LIST without setting up a FOR-loop (either explicitly or implicitly). For example, given the PORT is:

```
CREATE L1 PORT LIST (5)
    S1 STR (3);
```

The following assignment is illegal:

```
L1.S1='FOO';
```

because it treats the five members of S1 as if they were a single data item. 12j

Two outermost containers with the same name may not be open at the same time. This is true even

though the containers may have different pathnames in the directory. 12k

If an output PORT is punctuated, all assignments before each punctuation character must be completed before any assignments are made after the punctuation character. That is, the datacomputer cannot back up over punctuation in an output PORT. For example, given an output PORT of the form:

```
PP PORT LIST
    S1 STRUCT
        A1 STR (3),P=EOR
        A2 STR (3),P=EOR
    END
```

assignments must be made in the same order as the STRs appear in the STRUCT

..

```
A1='FOO';
A2='BAR';
```

will take effect correctly, but

```
A2='BAR';
A1='FOO';
```

will not.
Because of the internal paging of the datacomputer, STRUCTs containing long STRs (i.e. greater than 2560 ASCII characters) have a similar restriction, for example, the LIST

```
FF FILE LIST
    S1 STRUCT
        A1 STR (10000)
        A2 STR (10000)
        A3 STR (10000)
    END
```

may have assignments done only in the same order as they appear in the STRUCT.

121

IWWS 23-JAN-75 05:51 31693

DATALANGUAGE MANUAL

SEISMIC DESCRIMINATION GROUP

23 JAN 75

LINCOLN LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLGY
CAMBRIDGE, MASSACHUSETTS 01242

(J31693) 23-JAN-75 05:51;;;; Title: Author(s): Discrimination Group Seismic/IWWS; Distribution: /RMS2([INFO-ONLY]) RTL([INFO-ONLY]) SKS([INFO-ONLY]) ; Sub-Collections: NIC; Clerk: IWWS;
Origin: < IWWSS, DATALANGUAGE-MANUAL-0/10,NLS;1, >, 11-DEC-74 10:16 IWWS ;;;;####;

EJK 23-JAN-75 07:47 31694

Send PART of the Mail Form

The following is a copy of the sendmail form that I fill in and send whenever I want to get a message to one or more users. My usual mode of operation is to fill it in and then go to SENDMAIL subsystem and process the form. Following this I delete modifications and I have a virgin form ready to use the next time. 1

The last time I used the form I did not delete mods, because I wanted to check to see if all the message would arrive. The last time it did not. 2

SO here is the form with the message that you should have received, HELP!!! What did I do wrong??? 3

TITLE: Some Questions About The Care And Feeding Of NLS-8
COMMENT: The message is in three parts even though it is less than 2000 characters - I wonder if it will arrive
AUTHOR(S):EJK
DISTRIBUTE FOR ACTION TO: feed dls sgr
DISTRIBUTE FOR INFO-ONLY TO:
SUBCOLLECTION(S):
ACCESS STATUS: RADC
MESSAGE: I have some questions new and old about the use of NLS:
1. I have been unable to use the Accesslist feature on Col Krutz' initials file. The protection on the file is 770000 and I have tried to follow the instructions to set up an access list allowing access by RDK ARB and EJK. I can't make it work even though I think I have followed all the instructions. Should I change the Protection code to 777700? or what??
2. In the old NLS I could move things within my initials file by using a branch called contents which contains a listing of all the named branches in the file ie. (haveread) (mess). When asked by the system for the address to which something is to be moved, by typing <sp> and then bugging one of the branch names it functioned as though I had typed in the named branch. In the new NLS this does not work. I can't bug a branch name (except for the name at its real location) and have the thing bugged function as the name of a branch. I can however, type in a branch name. Comment?
3. When using the message form that I am using to send this, I find that if I get a lot of info in the message only the first 20 or 25% gets through. The rest gets lost, even though the total message is less than the 2000 character limit. Comment?? This happened on a message that I sent to a number of addressees re some missing terminals. Aside from thee inconvenience, it makes the sender (me) look like a moron. This message is long but less than 2000 characters. By the way does the form when filled in all have to be one statement, with CR's in it? Can I break it into several statements??
SEND THE MAIL. 4

EJK 23-JAN-75 07:47 31694

Send PART of the Mail Form

(J31694) 23-JAN-75 07:47;;;; Title: Author(s): Edmund J. Kennedy/EJK; Distribution: /FEED([ACTION]) SGR([ACTION]) DLS([INFO-ONLY]) ; Sub-Collections: RADC; Clerk: EJK;

JMB 23-JAN-75 08:55 31695

TRAINING REPORT--Mr. William Bangert [ARPA-Program Management Office]

Susan, Jim: This should go in the Users file that is rumored to exist. Please describe future conventions for this type of reporting.

TRAINING REPORT--Mr. William Bangert [ARPA-Program Management Office] 1

Hands-on DNLS Lesson 21-JAN-75 2

New Capabilities: 2a

Use of Environment: 2a1

Login with Line Processor 2a1a

Use of Mouse: buttons for CA, Bug, BC, CD, Viewspecs (though I don't think he mastered the latter) 2a1b

Familiarity with DNLS screen 2a1c

Jumping 2a2

to BUG 2a2a

to Item using Address 2a2b

to Origin, Next, Back, Return
[not too much practice with Successor, Predecessor, Up, Head, Tail, End, File Return--though these are in SGR's "Introduction to DNLS" course outline] 2a2c

Load file 2a3

He does not use his initial file, though he has now seen it upon entering DNLS. He simply then loads his destination file. 2a3a

DNLS editing--Extensive Practice with: 2a4

Insert Text, Word, Character, Statement 2a4a

Delete Text, Word, Character, Statement, Branch 2a4b

Copy Statement 2a4c

Replace Text, Word, Character, Statement 2a4d

Move Text, Word, Character, Statement 2a4e

Transpose Character, Word, Text, Statement, Branch 2a4f

Basic file-handling capabilities in TNLS (3rd level per SGR) transferred to DNLS 2a5

TRAINING REPORT--Mr. William Bangert [ARPA-Program Management Office]

Split Screens 2a6

Insert Edge and Jumping to fill window merely introduced--not much practice. This is where we left off. 2a6a

We also worked on appropriate (detailed) use of Archive and Interrogate processes. 2a7

APPLICATION NOTE: 2b

We worked on efficient editing techniques on one of his work files having columnated tables, which he created in TNLS. This was the reason for concentration on Editing, usually with the command-word "Text". 2b1

JMB 23-JAN-75 08:55 31695

TRAINING REPORT--Mr. William Bangert [ARPA-Program Management Office]

(J31695) 23-JAN-75 08:55;;;; Title: Author(s): Jeanne M. Beck/JMB;
Distribution: /UD([INFO-ONLY]) CKM([INFO-ONLY]) ;
Sub-Collections: SRI-ARC UD; Clerk: JMB; Origin: < BECK,
BANGERTTRAINING.NLS;3, >, 23-JAN-75 08:48 JMB ;;;;####;

FEED 23-JAN-75 09:19 31696

LIT typin bug

I tried to make the bug occur on he lp and couldn't. So I've moved it to the db branch in th feedback file. If it really does only occur on our taskers that are going away soon it prbably won't be fixed. Susan/FEED 1

FEED 23-JAN-75 09:19 31696

LIT typin bug

(J31696) 23-JAN-75 09:19;;;; Title: Author(s): Special Jhb
Feedback/FEED; Distribution: /KIRK([INFO-ONLY]) FEED([INFO-ONLY]
) ; Sub-Collections: SRI-ARC; Clerk: FEED;

FEED 23-JAN-75 09:29 31697

Problem with sendmailform commad

I've moved your message to the bugs branch. Applications doesn't have any programmer this monthe (so I hear) so I don't know when it'll be fixed. I'll let youknow.! Susan/FEED 1

FEED 23-JAN-75 09:29 31697

Problem with sendmailform commad

(J31697) 23-JAN-75 09:29;;;; Title: Author(s): Special Jhb
Feedback/FEED; Distribution: /POOH([INFO-ONLY]) FEED([INFO-ONLY]
) ; Sub-Collections: SRI-ARC; Clerk: FEED;

FEED 23-JAN-75 10:04 31698

Test of message with CTRL V's and CR's

This is a test to see if a message with CR's in it gets through - -
of course the ones in here are going to be CTL V CR's. O.K. here
goes.
The second line.
The third line.

1

FEED 23-JAN-75 10:04 31698

Test of message with CTRL V's and CR's

(J31698) 23-JAN-75 10:04;;;; Title: Author(s): Special Jhb Feedback/FEED; Distribution: /SGR([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: FEED;

LAC 23-JAN-75 11:24 31699

Suggestions and request for documentation

I would like to make a couple of suggestions for improving NLS. 1

Responses to the (Y/N): prompt should be displayed. It is useful to have some indication besides an action being taken or not, of what the system recieved.(Ie that you really typed what you thought.) Since it's a pretty much a Binary decision (ignoring the command repetition, which is easily detectible), it would probably be sufficient to indicate a "no" response only, if this is easier to implement. 1a

I would also like, when I recieve Journal mail referencing another document, to have the system have already assured the referenced document(s) is online and has a readable protection. I find it very frustrating to try to follow a link and find that I have to request an interogate and wait for the file to be loaded before I can find out what the reference had to say. One level(ie documents directly referenced, but not those referenced in turn- Idont want to chain in the whole tape library) would probably be sufficient. It would be useful also to check at sending time the access protection of all referenced files (journal or not), and flag to the author any which don't have at least general read access. 1b

Second, could you send me copies of the following documents: 2

L10 programming manual	2	2a
Output processors guide	4	2b
TNLS-8 quick reference card	15	2c

if there exists TNLS-8 version of ARC 19200 (TNLS Users guide, 28 Nov 73), i could use 5 of these. 2d

Thank you
/Larry Crain 3

LAC 23-JAN-75 11:24 31699

Suggestions and request for documentation

(J31699) 23-JAN-75 11:24;;;; Title: Author(s): Lawrence A. Crain/LAC; Distribution: /FEED([ACTION]) ; Sub-Collections: NIC; Clerk: LAC; Origin: < CRAIN, TEMP,NLS;1, >, 23-JAN-75 11:21 LAC
;;;;####;

new directories

request for new directories

DAP 23-JAN-75 11:44 31700

new directories

Please open two new directories as follows: 1

DAP 23-JAN-75 11:44 31700

new directories

(J31700) 23-JAN-75 11:44;;;; Title: Author(s): David A. Potter/DAP;
Distribution: /FEEDBACK([ACTION]) DAP([INFO-ONLY]) EJA2([
INFO-ONLY]) ; Sub-Collections: NIC FEEDBACK; Clerk: DAP;

JPC 23-JAN-75 14:00 31701

Specs & Requirements for Accounting System

Jim, this is a draft of the type of position paper we discussed. I'd appreciate any feedback from you concerning it. Joe

Requirements for Accounting System for IS 1

Background: 2

The financial picture of the IS division for a fiscal year consists of OiR's that represent efforts carried over form the prior fiscal year, new starts that might be in varying stages of procurement and planned efforts that might begin in that or subsequent years. During a current FY, a plan is conceived for expending the next fiscal year's funds, consisting of OiR's and planned efforts. This plan (with modifications) is implemented as we progress through the next FY. The cash flow of IS is the rate at which the funds we are authorized to expend during a fiscal year are paid to contractors. From the time the plan is conceived until actual obligation, estimated amounts can change one or more times during this period. 2a

Problem Identification 3

As the plan is implemented with modifications, there is no single source for getting an up-to-date report on the financial condition of the division. Projected spending for the end of the fiscal year is just a guess made by human intuition. When the plan changes in a certain area, there is usually a rippling effect as other efforts are affected indirectly. After a few of these changes, their ramification is beyound human comprehension and guesswork becomes totally inadequate. A way is needed to track an effort through its life cycle (from its conception in the planning stage, during the various phases of procurement and then through actual implementation) and to collectively represent selected efforts in such a way as to aid the management process. To do this, it must handle Initiated, Committed and Obligation money and dates as well as estimated expenditure rates that predict cash flow and the actual amount of money that the government is obligated for. 3a

System Objective: 4

An on-line accounting system will be built under NLS utilizing L10 programs. It would be capable of generating the finanacial status of division efforts in terms of when they were initiated, committed and obligated and for how much. Furthermore, it will be able to project future expenditures based on expenditure rate functions that will be assigned to each effort. The system will also support simulating the addition or deletion of efforts and letting the starting dates for efforts to slide. The transition

from planning to implementing will be represented. Progress reports will be generated that will deal with schedules and financial data. The accounting system will serve as a basic tool for program planning and to aid in the control of money that is spent during a fiscal year. The system should support both near-term and long-term planning. 4a

Scope and Boundaries: 5

The basic unit of the system will be a Purchase Request. PRs will form the core for tracking efforts and will be related to TPO plans. 5a

Money controlled by the accounting system will include OIR's (on-going efforts), new starts and planned efforts (estimated). This will give the system control of financial data for the fiscal year in the form of initiated, committed, obligated amounts and their dates. 5b

The basic unit of time for the system will be a month. 5c

Algorithms will be developed to project the expenditure rate for each effort. Its application will aid in the forecasting of actual money expenditures at any point in time during the fiscal year. 5d

There will be three types of expenditure rates: a straight linear function to deal with study-type contracts, a bell-shaped curve to handle hardware buys and a point function for off-the-shelf buys where the entire contract amount is spent when the contract is signed. The first two functions must be capable of incorporating no-cost extensions. 5d1

Estimated quantities in the system will be readjusted as necessary. For example, if an effort is initiated three weeks early, the estimated dates for commitment and obligation will be re-estimated. At the same time dollar amounts for the effort may also be changed. 5e

Actual expenditures will be kept track of for each effort. It will be computed from vouchers that have been paid, vouchers in progress and estimates of what the government is actually obligated for. 5f

A symbolic model of the system will be designed that will be suited for the prediction or determination of effects of changes in the actual system. 5g

This capability to perform simulations of the organization's financial picture will allow management to judge the impact of real or projected changes in the actual system. This will include addition or deletion of money to the total allocated project fund. Another possible exercise will involve letting the starting dates of new efforts slide in order to give management the ability to improve the cash flow of the organization by having certain efforts temporarily postponed and even inserting new efforts into the picture on a projected basis. If an effort is cancelled, its associated values will be deleted from the accounting system. 5g1

Efforts will be available on an individual basis, or grouped by project or division or TPO area. 5h

A graphics capability will developed that will be used in the formulation of reports. Simple plotting will be available as well as bar graphs composed of NLS statements. 5i

Standard Operating Procedures (SOPs) will be developed to control data collection and data entry. 5j

Assumptions 6

Feb. 1 will be the start date for this effort. 6a

Another person from ISIM will be identified to learn L10 and assist in the project. 6b

Branch administrators will be responsible for collecting the data for the system. This will involve initial entry of planning information, deciding which efforts to include and determining when to update the values associated with an effort. Tom Bucciero should be responsible for writing up the SOPs under engineer direction. 6c

Project engineers will be responsible for choosing the expenditure rate function that promises to most closely approximate their effort and informing the branch administrator of their decision. 6d

The accounting system may not capable of automatically providing relationships between various PRs in the system. 6e

Programming will be done at SRI so that adequate training and support can be available. 6f

Operational Scenerio: 7

Data will be collected at the branch level. It will consist of three types: (1) plans, (2) efforts under way and (3) updates to the existing database in the form of modifications to existing values or replacement of efforts in the system. 7a

Once this data is gathered, it can be sent to the data entry focal point. This will probably be Bobbie Carrier who will supervise its entry into the database and insure database integrity. The PSO will have the job of actually entering the data and verification programs may be written to check incoming data for inconsistencies (a more ambitious undertaking will be to have a data entry program that quizzes the inputter as to what kind of database change is being attempted so that the program can proceed to guide the entry of the data). 7b

When the database is validated, it will be available for queries. A Purchase Request may be specified by number and the following data items would be displayed: dates and amounts for initiation, committment and obligation, an estimated expenditure for that month, the projection of the expenditures for the duration of the effort and the actual expended amount for that effort based on the criteria that has already been established. A plot of this information will be available that could be displayed on a CRT or TTY or sent to the Printer according to the user's discretion. 7c

Multiple efforts could be considered at the same time. A summing capability will compute totals of the selected efforts for the projected expenditures and for the actual expenditures. These also could be plotted over a fiscal year. The number of entities that are selected would depend on the view the user wished to see of the database; whether by contract, by project or by division. 7d

The Dynamic system will provide the same kinds of information that the static system does except that temporary changes will be allowed. Efforts could be identified as sliding in the procurement cycles, incorporating delays or no-cost extensions, overruns or additions or deletions of efforts. The impact of these changes could then be viewed. To make any of these changes permanent, offical updates would have to be sent to the data entry point for inclusion to the database. 7e

Planning information in the Form of TPO areas would also be contained in the database. The associated bar graphs would be displayed (but may not be generated automatically). 7f

Specs & Requirements for Accounting System

Resource Requirements 8

Manpower	FEB	MAR	APR	MAY	JUN	JUL	AUG	
Cavano	.8	.8	.8	.8	.8	.8	.8	8b
Bucciero	.3	.3	.1	.1	.1	.1	.1	8c
Carrier		.1	.2	.2	.2	.2	.2	8d
Programmer	.2	.2	.2	.2	.2	.2	.2	8e
PSO	.1	.1	.2	.2	.2	.2	.2	8f
Other Br Admin			.1	.1	.1	.1	.1	8g

Milestones 9

Database for ISI branch	March	9a
Database for IS division	April	9b
Static System for ISI branch	June	9c
Static System for IS division	July	9d
Dynamic System for ISI branch	August	9e
Dynamic System for IS division	September 1	9f

Implementation 10

Design	Feb - Mar	8 weeks	10a
Training	April	4 weeks	10b
Programming	April - June	8 weeks	10c
Debug & Test	June - July	4 weeks	10d
Documentation	Feb - July		10e

JPC 23-JAN-75 14:00 31701

Specs & Requirements for Accounting System

(J31701) 23-JAN-75 14:00;;;; Title: Author(s): Joe P. Cavano/JPC;
Distribution: /JCN([ACTION]) ; Sub-Collections: RADC; Clerk: JPC;
Origin: < CAVANO, REQ.NLS;2, >, 23-JAN-75 13:56 JPC ;;;;####;

DAP 24-JAN-75 05:46 31702

new directories/2

for some reason 31700 did not send you any information; here's the full text.

new directories/2

TITLE: new directories
COMMENT: request for new directories
AUTHOR(S): DAP
DISTRIBUTE FOR ACTION TO: feedback
DISTRIBUTE FOR INFO-ONLY TO: dap,eja2

MESSAGE:Please open two new directories as follows:
Directory name: tryout
Password: ets
Protection: 777752

Directory name: rumar
Protection: 770000
Password: lib

The first directory will be a general directory, open to anyone at ETS who wants to experiment with the system; the second directory will be used by Ruth Ekstrom, Marlaine Lockheed, and Abby Harris for work on a project in which they are all involved. All other information (hardcopy address, etc.) is the same as for other ETS group members.

One last point: although I am now the Architect in the system's eyes (eyes?), McNally (BJM) also still has that designation. Please change that so that KWAC mail no longer drops into that mailbox; that directory will probably be the one used for Ernie's ARPA project, and I'd like to keep it short and neat.

Please let me know when the new directories are up. TRYOUT may already be in the works; I included a request for it in a msg to norton and bair 17-JAN-75 1019-PDT.

1

DAP 24-JAN-75 05:46 31702

new directories/2

(J31702) 24-JAN-75 05:46;;;; Title: Author(s): David A. Potter/DAP;
Distribution: /FEEDBACK([ACTION]) DAP([INFO-ONLY]) EJA2([
INFO-ONLY]) ; Sub-Collections: NIC FEEDBACK; Clerk: DAP;

JMB 24-JAN-75 07:28 31703

XGP (Re--25194,)

I'd love to show anyone the neat features and the hangups of the XGP, I don't know an awful lot about the program running it (dave bushi of keydata does) but i have had a lot of experience helping people here use it. Looking forward to seeing you Monday or Tuesday; maybe we could have lunch together or something? 1

JMB 24-JAN-75 07:28 31703

XGP (Re--25194,)

(J31703) 24-JAN-75 07:28;;;; Title: Author(s): Jeanne M. Beck/JMB;
Distribution: /DVN([ACTION]) ; Sub-Collections: SRI-ARC; Clerk:
JMB;

DAP 24-JAN-75 07:46 31704

thoughts about your visit

To begin with, I thought you might be interested in knowing that I have so far logged into RUTGERS-TIP as Czerniac, Smedlp, and George, all with no password. Neat system, huh?

As for your visit:

Len Swanson will be joining us for part of the time. His role at ETS is roughly comparable to that of the AKW Architect. He is very interested in OFFICE-1, and would like to do whatever he can to facilitate the growth of an ETS user community. I think it would be mutually helpful for him to join us in our early planning efforts.

Also, I'd like you to spend some time with Ruth Ekstrom, Marlaine Lockheed, and Abby Hoffman, for whom I have requested a directory (RUMAR). They have a 3000-item bibliography they want to put on the system. Basic question is how best to structure it so as to maximize usefulness and ease of retrieval?

Finally: the calculator subsystem. Is it realistic to think about making it function as a programmable desk calculator? Remember that not all knowledge work is verbal and conceptual in an abstract sense; some of it is pretty quantitative. A programmable calculator is a very useful tool, enabling one to calculate basic descriptive and inferential statistics without a programmer and a 360/65. This capability would considerably increase interest in the system here. Increased interest is not immediately necessary (we don't want people beating down my door to get on the system when we don't have room for them), but it's highly desirable in the long run.

Yours, Dave 1

DAP 24-JAN-75 07:46 31704

thoughts about your visit

(J31704) 24-JAN-75 07:46;;;; Title: Author(s): David A. Potter/DAP;
Distribution: /JCN([ACTION]) DAP([INFO-ONLY]) ; Sub-Collections:
NIC; Clerk: DAP;

FEED 24-JAN-75 11:00 31705

Status of NLS Suggestions

Thanks for the suggestions - I think they're good ideas. I've put the suggestion about Y/N in the design bugs section of feedback and the suggestion about archived files referenced in journal items in the design recommendations section of feedback. Susan/FEED 1

Status of NLS suggestions

(J31705) 24-JAN-75 11:00;;;; Title: Author(s): Special Jhb
Feedback/FEED; Distribution: /LAC([INFO-ONLY]) FEED([INFO-ONLY])
; Sub-Collections: SRI-ARC; Clerk: FEED;

FEED 24-JAN-75 11:37 31706

Insert Left and Other Neat Things

Ken, Thanks for the suggestion about insert left. It would be better to teach people that than to replace the first character with new text plus character as is currently done. Aside from the problem with syntax, there isn't any money for development type things submitted to feedback. So... unlesss it can be classified as NSW work itt will continue to sit in feedback as are many other good and easily accomplished suggestions and sometimes bugs. Susan/FEED 1

Insert Left and Other Neat Things

(J31706) 24-JAN-75 11:37;;;; Title: Author(s): Special Jhb Feedback/FEED; Distribution: /KEV([INFO-ONLY]) NPG([INFO-ONLY]) RWW([INFO-ONLY]) DCE([INFO-ONLY]) JCN([INFO-ONLY]) FEEDBACK([INFO-ONLY]) ; Sub-Collections: SRI-ARC NPG FEEDBACK; Clerk: FEED;

KM 24-JAN-75 13:25 31707

NLS Transfer

Sorry to take so long to respond but I have been off work for a couple of days, however consider the following :- 1

A smaller system implies either reduced capabilities for everybody or full capabilities for a reduced number of people or a combination of the two, 1a

I suggest we have two levels of login (as at one site on ARPANET I forget which) 1b

a reduced level available to every user which has mail-reading, access to NICE, 1b1

a full level which has access to everything for a restricted number of users, 1b2

If the first of these were to include any NLS (except that obtained indirectly through NICE) it would certainly be a restricted set of commands, 1c

This could well be the way to go anyway, since as you pointed out, even a full OFFICE-1 type system could not support all users in the Agency, and the number of users allowed non-restricted access can be adjusted when an upgrade occurs, 1d

As I understand it, NSW NLS will not be available until this Summer at the very earliest, and anyway, we will not have the PDP-11s around the network until Spring/Summer 1976. So I suggest we consider upgrading to NSW NLS at that time, 1e

Maintenance of NLS could be handled by C41 and their handling of old and writing of new USER PROGS is a good area of interest for them, 1f

On the more core versus more disk question - we need to make an estimate of the minimum amount of disk space with which we can live reasonably comfortably. Then increase (if necessary) disks to this level and any more money we should spend on more core, 1g

KM 24-JAN-75 13:25 31707

NLS Transfer

(J31707) 24-JAN-75 13:25;;;; Title: Author(s): Keith McCloghrie/KM;
Distribution: /JNH([INFO-ONLY]) TEH([INFO-ONLY]) ;
Sub-Collections: NIC; Clerk: KM;

MAP2 24-JAN-75 13:40 31708

SRI Utility Slot User Group (#35)

I will be maintaining a file which will describe the SRI Utility slot user community. It will include reference to all group directories and links to a file describing each user. Note also the list of important phone numbers. Feel free to access this file (using the following link) and let me know if any information is in error. Given the new slot allocation algorithim and our expanded use of the system you will undoubtedly need to get in touch with other users from time to time...

Link to SRI Slot users file: <PLACKO,SRIUSERS,1;dbn>

-- Mike

1

SRI Utility Slot User Group (#35)

(J31708) 24-JAN-75 13:40;;;; Title: Author(s): Michael A. Placko/MAP2; Distribution: /RAH([INFO-ONLY]) ; Sub-Collections: NIC; Clerk: MAP2;

RBp 24-JAN-75 13:45 31709

Answer for Patterson - FY78 Program

Colonel Krutz,

Below is what I suggest we send to Major Patterson regarding the information he requested concerning the FY78 program for P.E.63728F. Also, there are a few things which are becoming of concern to me because they may have an adverse effect on our credibility: 1

1. I really have no idea where the outyear figures were generated. When the Steering Group was in being, it recommended a $4.0M level. We now find ourselves in a position of having to justify these high numbers. I think we had better take a realistic look at the program element and recommend changes to the FYDP 's if necessary. 1a

2. I am beginning to see some of the efforts planned for 5550 being proposed to other users. One which comes to mind right off is the PSL for AFSC/ACD for which we have received obligation authority in FY75 for $95K. This is proposed as a FY76 start in the 5550 PMP. Since such an effort has been estimated to cost $250 - $350K, I have asked Major Ruple to think about phasing the effort so that 5550 funds can be used in FY76, to check into the need for forward financing approval for the funds he has received since he said he doubts that all of the funds can be expended this year, and have also asked him to look into the need for D&F approval of 5550 funds if this is what is to be accomplished. 1b

3. Our FY76 program is going to increase substantially. To me, this means that we must get an early start on our planned procurements so that we get most of our efforts on contract early in the year. I suggest you talk with DO and PM to get their agreement on IS now starting FY76 paperwork through the cycle up to committment. This should also help minimize any adverse effects if we have to move to Hanscom. 1c

While on the subject of getting the FY76 program on the road, I think we should get Major Eiden here as soon as possible to present his Requirements Analysis Task and either (1) commit or non-commit ourselves to supporting his efforts or (2) going with an approach we think to be better. This area seems to be one for which increasing significance is surfacing. 1c1

RBP 24-JAN-75 13:45 31709

Answer for Patterson - FY78 Program

Major Patterson,

Outlined below is the five year plan for 63728F. It is intended to provide you with the information you require regarding the FY78 plan. You have the descriptions of the proposed tasks in the PMP and in the added descriptions previously supplied for the FY77 planned program. The emphasis in FY77 & FY78 will be in Software C, R, & T. The five areas below will be emphasized: 2

1. Software Data Repository Implementation (pilot & full-blown). 2a

2. Language Control Facility Implementation. 2b

3. Modern Programming Practices Evaluations. 2c

4. Software Tools Acquisitions, Testing, & Evaluations. 2d

5. SEMANOL for COBOL & FORTRAN. 2e

5-YEAR PLAN FOR 63728F 3

	FY75	FY76	FY7T	FY77	FY78	
ARCHITECTURE						4
Assoc Proc	370	535	70	200	200	5a
Config Proc			-	500	800	5b
Dist Computation Fac			-	600	600	5c
S/W TECHNOLOGY						6
S/W C, R, & T	1270	3185	1130	4700	6500	6a
Reqmts Anal			-	500	1000	6b
Multisource Data Fusion				-	500	6c
ADP SYSTEM SUPPORT						7
Security	550	280	-			7a
H/W TECHNOLOGY	110	-				8
PROJECT 2108			-	1000	1000	9
TOTALS	2300	4000	1200	6500	10600	10

RBP 24-JAN-75 13:45 31709

Answer for Patterson - FY78 Program

(J31709) 24-JAN-75 13:45;;;; Title: Author(s): Roger B. Panara/RBP;
Distribution: /RDK([ACTION]) RBP([INFO-ONLY]) ; Sub-Collections:
RADC; Clerk: RBP; Origin: < PANARA, FY78ANSWER.NLS;2, >,
24-JAN-75 13:40 RBP ;;;; ####;

FGB 24-JAN-75 16:26 31710

More Directories & Idents for NSRDC

Jim,
Please do the following for me as quicklyas possible
I will tell you which ident go with what directories later.
Thanks. Frank 1

Please increase the disk space allocation of thefollowing directories: 2

BRIGNOLI 500 2a

AVRUNIN 500 2b

Please stablish the followng directories: 3

NAVIMP 300 Implementation Subgroup 3a

NAVAPS 300 Applications Subgroup 3b

NAVINFO 500 Information Exchange Subgroup 3c

NAVMINI 300 MinicomPuter Software 3d

Please delete the following directory: 4

NAVLIS 300 4a

Please establish the idents for the following: 5

Bob Archer Jr. 5a

Code 332 5a1

Naval Surface Weapons Center 5a2

White Oak Lab 5a3

Silver Spring, Md. 20910 5a4

Paul C. Bishop 5b

Code 731E 5b1

Naval Coastal Systems Lab 5b2

Panama City,,Florida 32401 5b3

I. Bornstein 5c

Code 4535 5c1

More Directories & Idents for NSRDC

Naval Undersea Center	5c2
San Diego, Ca 92152	5c3
David Brown	5d
Naval Coastal Systems Lab	5d1
Panama City, Fla 32401	5d2
Harold Doerfel	5e
Code 8501	5e1
Naval Air Development Center	5e2
Warminster, Pa 18974	5e3
D.F. Eliezer	5f
Code KXE	5f1
Dahlgren Lab	5f2
Naval Surface Weapons Center	5f3
Dahlgren, Va, 22884	5f4
J.M. Goertz	5g
Naval Electronics Lab. Center	5g1
San Diego, Ca, 92152	5g2
R. Hunzinger	5h
Naval Air Development Center	5h1
Warminster, Pa, 18974	5h2
J.G. Noel	5i
Code 5200	5i1
Naval Electronics Lab. Center	5i2
San Diego, Ca, 92152	5i3
John Smith	5j

Naval Research Lab,	5j1
Washington, D.C, 20375	5j2
E.P.Stemple	5k
Code KOE	5k1
Naval Surface Weapons Center	5k2
Dahlgren Lab	5k3
Dahlgren Va, 22448	5k4
R.A. Unger	5l
Naval Undersea Center	5l1
San Diego, Ca, 92152	5l2
Mary Lou Blessing	5m
Naval Surface Weapons Center	5m1
White Oak Lab	5m2
Silver Spring, Md, 20910	5m3
Robert A, Fleming	5n
Naval Electronics Lab, Center	5n1
San Diego, Ca, 92152	5n2
Connie Heitmeyer	5o
Naval Research Lab	5o1
Code 5403	5o2
Washington , D.C, 20375	5o3

FGB 24-JAN-75 16:26 31710

More Directories & Idents for NSRDC

(J31710) 24-JAN-75 16:26;;;; Title: Author(s): Frank G. Brignoli/FGB; Distribution: /JHB([ACTION]) FEED([INFO-ONLY]) JCN([INFO-ONLY]) ; Sub-Collections: NIC; Clerk: FGB;

DAP 27-JAN-75 05:56 31711

yesterday's problems

Thanks for your time and help yesterday. A series of QUITS got my job status straightened out; then a minor modification to the file, GOTO TENEX, RESET, NLS, LOAD FILE, UPDATE FILE OLD,,,and my pc finally joined its file.

So thanks again. I really appreciated being able to get some help on a Sunday afternoon with a problem that was more the fault of the phone company than anyone else (I always have connection problems when working at home that I never have from the office). That kind of response to a stranger's SOS enhances the feeling of community this system somehow gives.

Yours truly, Dave Potter 1

DAP 27-JAN-75 05:56 31711

yesterday's problems

(J31711) 27-JAN-75 05:56;;;; Title: Author(s): David A. Potter/DAP;
Distribution: /DCE([ACTION]) DVN([ACTION]) DAP([INFO-ONLY])
; Sub-Collections: NIC; Clerk: DAP;

addressing multiple statements

how-to-do-it???

DAP 27-JAN-75 06:04 31712

addressing multiple statements

Is there any way to simultaneously operate on a number of scattered statements (e.g.,3,7,14,21a,40,...)? For example, I would like to be able to:

1. Print statement at T/[A]: 3,7,8A,...
2. Insert word (to follow) A: 3,7,8a,...

I imagine you get the idea. Sounds simple, and it may well be; I haven't played around with it, thought for a change I'd ask first. 1

DAP 27-JAN-75 06:04 31712

addressing multiple statements

(J31712) 27-JAN-75 06:04;;;; Title: Author(s): David A. Potter/DAP;
Distribution: /FEEDBACK([ACTION]) DAP([INFO-ONLY]) ;
Sub-Collections: NIC FEEDBACK; Clerk: DAP;

MAP2 27-JAN-75 08:36 31713

SRI Utility Slot User Group (#35)

I will be maintaining a file which will describe the SRI Utility slot user community. It will include reference to all group directories and links to a file describing each user. Note also the list of important phone numbers. Feel free to access this file using the following link and let me know if any information is in error. Given the new slot allocation algorithim and our expanded use of the system you will undoubtedly need to get in touch with other users from time to time...

Link to SRI Slot users file: <PLACK□,SRIUSERS,1;dbn>

-- Mike

1

SRI Utility Slot User Group (#35)

(J31713) 27-JAN-75 08:36;;;; Title: Author(s): Michael A. Placko/MAP2; Distribution: /RA3Y([INFO-ONLY]) JCN([INFO-ONLY]) JHB([INFO-ONLY]) HEB([INFO-ONLY]) CAG2([INFO-ONLY]) KLM([INFO-ONLY]) PWO([INFO-ONLY]) SDP([INFO-ONLY]) BJR([INFO-ONLY]) ; Sub-Collections: NIC; Clerk: MAP2;

FEED 27-JAN-75 12:08 31715

Status of (25213,) and (25212,)

I've moved your suggestion about SID to the dr branch of feedback and your suggestion about the set external command to the db branch (sounded like it was a prompting problem?) Happy Monday to you!!
Feed/Susan

1

Status of (25213,) and (25212,)

(J31715) 27-JAN-75 12:08;;;; Title: Author(s): Special Jhb Feedback/FEED; Distribution: /KIRK([INFO-ONLY]) FEED([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: FEED;

KM 27-JAN-75 13:14 31716

testing teh journal mail

this is the message 1

KM 27-JAN-75 13:14 31716

testing teh journal mail

(J31716) 27-JAN-75 13:14;;;; Title: Author(s): Keith McCloghrie/KM;
Distribution: /TEH([INFO-ONLY]) KM([INFO-ONLY]) ;
Sub-Collections: NIC; Clerk: KM;

TEH 27-JAN-75 13:26 31717

testing journal mail from teh

this is the message 1

TEH 27-JAN-75 13:26 31717

testing journal mail from teh

(J31717) 27-JAN-75 13:26;;;; Title: Author(s): Tom E. Hassing/TEH;
Distribution: /KM([INFO-ONLY]) ; Sub-Collections: NIC; Clerk: TEH;

FGB 27-JAN-75 15:24 31718

test message

This is a test message. 1

FGB 27-JAN-75 15:24 31718

test message

(J31718) 27-JAN-75 15:24;;;; Title: Author(s): Frank G. Brignoli/FGB; Distribution: /AW([ACTION]) ILA([INFO-ONLY]) ; Sub-Collections: NIC; Clerk: FGB;

test

this is one 1

this is two 2

this is two a 2a

test

(J31719) 27-JAN-75 15:33;;;; Title: Author(s): Frank G.
Brignoli/FGB; Distribution: /ILA([ACTION]) AW([INFO-ONLY]) ;
Sub-Collections: NIC; Clerk: FGB;