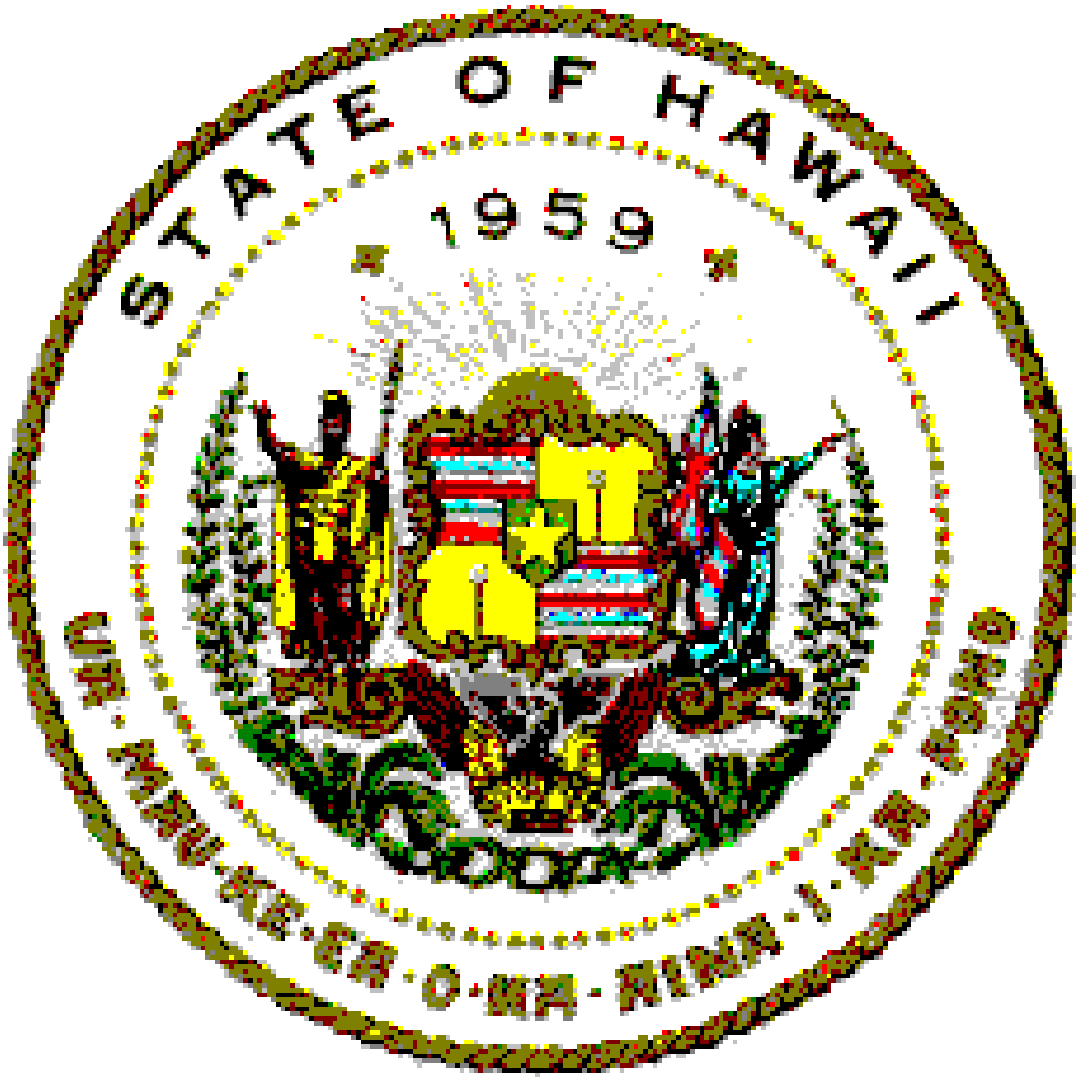

PANVALET USERS GUIDE

July 2001

TABLE OF CONTENTS

1 INTRODUCTION

This document, PANVALET USERS GUIDE, has been prepared by the Information and Communication Services Division (ICSD) to describe how the PANVALET system has been implemented in the State. The policies and procedures governing the use of the PANVALET system are documented in this manual. A complete description of what services are available and how they can be utilized is presented here. The policies and procedures describing the operations of the PANVALET system are contained in another ICS Division manual -- the PANVALET OPERATIONS MANUAL.

Section 2, General Information, gives a general overview of the PANVALET system and covers the functions of PAN#1 and the PAN#1 operating environment.

Section 3, Directly Available Services, covers the PANVALET concepts and conventions and gives a detailed description of the PANVALET services that are directly available to the programming staff. These are services that the programmer can utilize at any time by submitting his own job deck that calls for these services. Each service is discussed and followed by a description of the corresponding job deck.

Section 4, Indirectly Available Services, covers certain PANVALET services that are not directly available to the programming staff. These are services that the programmer can request by completing a PANVALET worksheet, punching the appropriate control cards, and submitting both the worksheet and the control cards to the PANVALET Librarian. The procedures are covered in detail in this section and it gives a detailed description of all PAN#1 commands and parameter options.

Section 5, Error Recovery and Backup of Test, gives examples of the applications of PAN#1 in typical user situations.

In summary, the PANVALET USERS' GUIDE is a working manual designed to show the programming staff what services are available and how to utilize these services. It does not cover the structure of the PANVALET system, its file requirements, its commands and options, and its operating characteristics. These important points are covered in detail in the PANVALET USER REFERENCE MANUAL OS prepared by Pansophic Systems, Incorporated. Therefore, to gain better knowledge and understanding of the PANVALET system, the programming staff is urged to read the PANVALET USER REFERENCE MANUAL OS, particularly Section 3 - "PANVALET Concepts and Conventions", before proceeding any further in this guide.

The user reference manual concentrates on describing the PAN#1 program which is the most important program as far as the users are concerned. This program is designed for

general use by the programming staff and performs the basic functions of random storage, maintenance, and direct access of all card image data sets in the PANVALET Libraries.

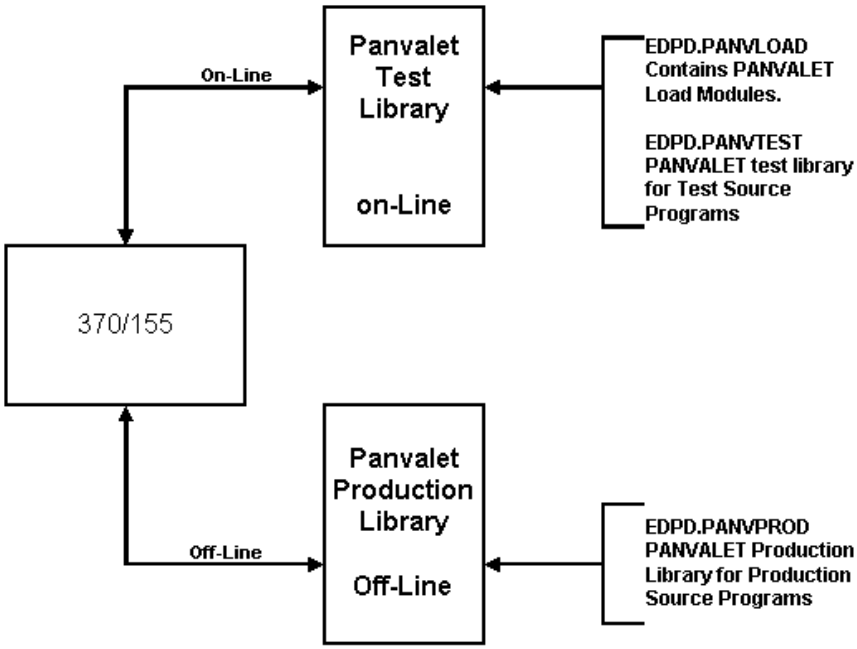

Figure 1. PANVALET Disk Packs

2 GENERAL INFORMATION

2.1 System Description

In order to understand the PANVALET system as implemented at the ICSD, it is necessary to know how the PANVALET Libraries have been set up.

There are two source program libraries and one load module library in our PANVALET system.

2.1.1 PANVALET Production Library

The PANVALET Production Library is stored off-line on a disk pack and mounted as needed. See Figure 1, PANVALET Disk Packs.

The Production Library contains the source programs for systems that are in production status. These are programs that are not under development or modification and therefore need not be stored on-line.

2.1.2 PANVALET Test Library

PANVALET Test Library is stored on an on-line disk pack. See Figure 1.

The PANVALET Test Library contains the source programs for systems that are in test status. These are programs that are under development or modification and therefore need to be stored on-line.

2.1.3 PANVALET System Load Modules

The PANVALET system consists of eight load modules. These load modules are located on the PANVALET Test Library disk pack and occupy two cylinders. See Figure 1

2.2 System Operation

2.2.1 Storing Source Modules

All source programs will be stored on either the PANVALET Test Library or the PANVALET Production Library. User Agencies outside of ICSD may have private PANVALET production source program libraries.

Only programs that are under development or modification will be stored on-line on the Test Library. All other programs will be stored off-line on the Production Library. Data sets will be monitored periodically. All data sets not in TEST status will be moved to the Production Library.

2.2.2 PANVALET Services Directly Available to Programming Staff

The PANVALET services for the programming staff can be used at any time. The programmer submits his own job decks that call for these services. Operations will handle these jobs like any other job. They will be run according to their job class and priority.

It is important to note that since these jobs can be run at any time, they can only refer to source modules that are stored on the on-line PANVALET Test Library. If any of these jobs refer to source modules on the off-line PANVALET Production Library, the job will abort.

These services will be covered in detail in Section 3 of this manual.

The following is a list of the PANVALET services that are directly available to the users:

a) Add a data set to the Test Library.

b) Add comments to the directory information block.

c) Copy a data set.

d) Eject to a new page on SYSPRINT.

e) Put an identification comment on the title page.

f) Select a portion of a data set to a work area.

g) Change the status of a data set.

h) Update a data set temporarily or permanently.

i) Change the user code.

j) Output a data set onto punched cards, the printer, or a work area.

k) Include a precoded data set within a major data set.

l) Create a superset with several subsets.

m) Resequence a data set to insert user sequence numbers.

n) Change the format/type of a data set.

2.2.3 PANVALET Services Not Directly Available to Staff

These are services that the programmer cannot request by submitting his own job deck. To request these services, the programmer must complete a PANVALET worksheet to have the appropriate PANVALET control cards sent to a member in the PDS: X.XZ.PANDATA. The worksheet with an authorization signature must be submitted to the ICSD Control Unit.

The Librarian will consolidate all of the requests that were submitted during the day. Then at the end of each day, all of the requests will be batched and processed. Thus, all requests for services submitted during the day will be processed and ready at the start of the next workday.

Two important points should be noted. First, the PANVALET services in this category will require a maximum lead time of only 24 hours. In most instances, the lead time will be less. For example, if a request were to be made at 2:30 p.m., it would be ready by the next morning.

Second, the services that were put in this category were carefully selected so that even the maximum 24 hours' lead time should not inconvenience the programming staff. With the minimum of advance planning, this once-a-day schedule should more than meet the programmers' needs except for very exceptional, emergency situations.

These services are covered in detail in Section 4 of this manual.
The following is a list of the PANVALET services that must be processed through the PANVALET librarian via the PANVALET worksheet.

a) Add a program to the Production Library.

b) Changing the status of a program on the Production Library.

c) Moving a data set from the Production Library to the Test Library.

d) Moving a data set from the Test Library to the Production Library.

e) Restoring a data set from the delete file tape to the Test Library.

f) Renaming a data set on the Test Library.

g) Changing the level number on a data set in the Test Library.

h) Changing the status of a program from "TEST" to "PROD" on the "TEST" Library.

2.2.4 PANVALET Run Schedule

* = optional - if requested, it will be run. X = job will be run.

FUNCTION	SCHEDULED DAYS				
PANVALET FUNCTIONS	M	T	W	T	F
BACKUP TEST LIBRARY	X	X	X	X	X
DIRECTORY LIST - PROD LIBRARY	X	X	X	X	X
MOVE data sets from PRODUCTION to TEST	*	*	*	*	*
MOVE data sets from TEST to PRODUCTION			*		
RESTORE deleted data sets to TEST					
LIBRARY (If not on PROD LIBRARY)	*	*	*	*	*
ADD data sets to PROD LIBRARY	*	*		*	*
DELETE data sets from TEST LIBRARY			*		
DELETE data sets from PROD LIBRARY		*			
BACKUP PROD LIBRARY			*		
DIRECTORY LIST - PROD LIBRARY			*		

3 DIRECTLY AVAILABLE SERVICES

This section gives a detailed description of the PANVALET services that are directly available to the programming staff. These are services that the programmer can utilize at any time by submitting his own job deck that calls for these services. Each function is explained briefly, followed by an illustration of a job deck to show the positions of the PANVALET control cards. References are made to the user reference manual whenever more detailed explanations may be helpful.

3.1 General Comments on PAN1

3.1.1 Maintaining PANVALET data sets

Files used in maintaining the PANVALET data sets:

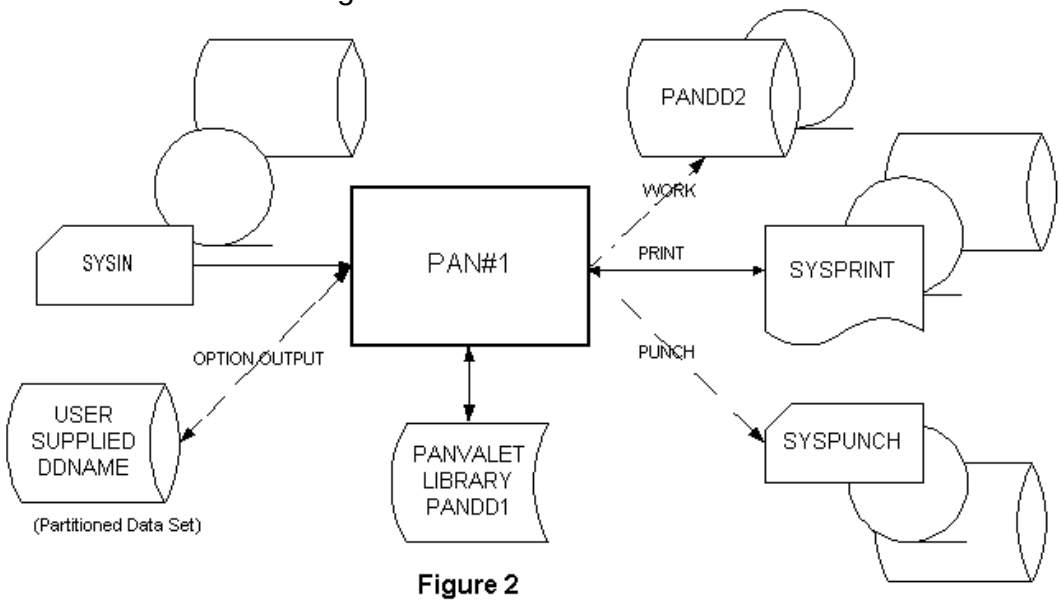

Figure 2

3.1.2 PANVALET JCL

JCL to Use PAN1

```
//jobname JOB (idno,pms),'name',MSGLEVEL=(0,0),CLASS=class
//stepname EXEC PAN1
//SYSIN DD *
                PANVALET Control Cards
/*
```

3.1.3 PANVALET Control Card Descriptions

1) JOB CARD:

//jobname JOB (idno,pms),'name',MSGLEVEL=(1,1),CLASS=class

where:

jobname - Follows the OS Naming Conventions PMSJaJn.

idno - User identification number.

pms - Is your system charge code.

name - 20 characters allowed for programmer's name and/or job description.

Class - Job classification is determined by the following table:

		I / O		
		No Mounts	Mounts	4 Mounts In any step
REGION	60K	B	F	J
	108K	C	G	J
	208K	D	H	J
	308K	E	I	J

2) EXEC CARD DESCRIPTION: //stepname EXEC PAN1

stepname- If used in a production job stream, follow the OS Naming Conventions PMSJaJnS where S starts with the alphabets.

3) GENERAL PANVALET CONTROL CARDS DESCRIPTION

PANVALET CONTROL CARD FORMAT

++COMMAND parameter1(,parameter2,...,parameter5)

++ - In columns 1 and 2 identifies a PANVALET control card.

The INCLUDE command is the only exception, the ++ starts in card column 8.

-- - In card columns 1 and 2 identifies a conditional PANVALET

control command which will be processed if the preceding command was successful.

COMMAND - The PANVALET command as listed on the PANVALET user reference card. The command follows immediately after the second plus (+) or (-).

parameters - Follows the command by at least one (1) space; multiple parameters are separated by commas, with no intervening spaces or extraneous characters.

```
++ COMMAND parameter1(,parameter2,.....parameter5)

++ - In card columns 1 and 2 identifies a PANVALET control card.
        The INCLUDE comand is the only exception, the ++ starts in cardcd column 8.

-- - In card columns 1 and 2 identifies a conditional PANVALET control command
        which will be processed if the preceding  command was successful.

COMMAND  - The PANVALET command as listed on the PANVALET user reference card
        The command followws mmediately after the second plus (+) or minus (-).

parameters - Follow the command by at least one (1) space; multiple parameters are
        se[parated by commas, with no intervvening space or extraneous characters
```

Figure 3

3.2 PAN#1 Functions

3.2.1 ADD a data set to the Test Library

(Reference: Pages 22-26 of the User Reference Manual)

Only INCLUDE modules or data sets that are being used should be on the Test Library. All other data sets should be stored on the Production Library.

The ++ADD command is used to add new data sets to the Test Library. The records in the input file which follow that statement will be written in the library until a record with a "++", "--", "/*", or "/&" in positions 1 and 2 is

encountered. The resulting data set will reside in the library with status of "TEST", "ENABLED", "ACTIVE", and with a modification "LEVEL" of 1.

Before adding a data set to the Test Library, check both the Test and Production Library directories for duplicate names. If none exists, proceed with the add function.

1) Case #1: The data set(s) to be added are on cards.

a. The job deck is assembled as follows:

```
//jobname JOB (idno,pms)
//stepname EXEC PAN1
//SYSIN DD *
        (cards for 1st data set follow)
++ADD dataset-name,lang-type,option,idno
Card Deck
        (cards for each additional data set follow)
++EJECT
++ADD dataset-name,lang-type,option,idno
Card Deck
/*
```

b. Control Card Description

JOB and EXEC cards are defined in.

++ADD card:

++ADD dataset-name,lang-type,option,idno

dataset-name - The name of the data set to be added to the Test Library. Follow the 10-character OS Naming Convention: PMSJaJnSN"S"VaVn.

lang-type - Must be specified.

List of allowable language types:

	C.C. Stored:
ANSCOBOL .	 7-72

AUTOCODER	 6-75
BAL (or ALC)	 1-72
COBOL	 7-72
COBOL-72	 1-72
FORTRAN	 1-72
PL/1 (or PL/I)	 1-72
RPG	 6-74
OBJECT	 1-72
JCL	 1-72
DATA	 1-80
OTHER	 1-80
USER180	 1-80
USER780	 7-80

option - Specify SEQ or LIST. If no listing is desired, omit parameter.

idno - Specify the user identification number of the person responsible for the data set.

2) Case #2: The data set(s) to be added is a sequential data set on disk or tape. (Reference: ADD - pages 22-26 of the User Reference Manual).

a. The job deck is assembled as follows:

```
//jobname JOB (idno,pms)
//stepname EXEC PAN1
//SYSIN DD *
      (cards for 1st seq. data set )
++ADD dataset-name,lang-type,option,idno
//DD DSN=dsname,UNIT=type,VOL=SER=volume,DISP=SHR
//  DD  *
      (cards for each additional sequential data set)
++EJECT
++ADD dataset-name,lang-type,option,idno
//  DD  DSN=dsname,UNIT=type,VOL=SER=volume,DISP=SHR
/*
```

Note that a number of data sets may be added with one job deck. Set a maximum of 5.

The JCL cards must be arranged in the order illustrated; otherwise, results are unpredictable.

b. Control Card Description

JOB and EXEC cards are defined section 3.1.3 PANVALET Control Card Descriptions.

++ADD card is defined section 3.2.1 ADD a data set to the Test Library.

Data set card:

// DD DSN=dsname,UNIT=type,VOL=SER=volume,DISP=SHR

dsname - The name of the data set to be added to the Test Library.

type - Unit type must be 2400, 23l4, or 3330.

volume - Volume serial number of the disk pack or tape.

NOTE: UNIT=type & VOL=SER=volume - Required only if the data set is not cataloged.

3) Case #3: The data set(s) to be added is a PDS member of a Partitioned Data Set (PDS). Reference: ADD pages 22-26, OPTION - page 30.1 of the User Reference Manual)

a. The job deck is assembled as follows:

```
//JOBNAME JOB (idno,pms)
//stepname EXEC PAN1
        (location of PDS)
//PDS DD DSN=dsname,UNIT=type,VOL=SER=volume,DISP=SHR
//SYSIN DD *
        (cards for 1st member of the PDS)
++OPTION INPUT,PDS
++ADD dataset-name1,lang-type,option,idno
        (cards for each additional member of the PDS)
++EJECT
```

++ADD dataset-name1,lang-type,option,idno
/*

Note that any number of members from a given PDS may be added with one job deck.

The JCL cards must be arranged in the order illustrated; otherwise, results are unpredictable

b. Control Card Description

JOB and EXEC cards are defined section 3.1.3 PANVALET Control Card Descriptions.

++ADD card is defined section 3.2.1 ADD a data set to the Test Library.

Exception: The dataset-name is the name of the PDS member that is to be added to the Test Library.

PDS card:

//PDS DD DSN=dsname,UNIT=type,VOL=SER=volume, DISP=SHR

dsname - Name of partitioned data set.

type - Unit type as 23l4 or 3330.

volume - Volume serial number of the disk pack.

c. As a final step convert the name of the data sets to conform to the OS Naming Conventions.

Punch the following PANVALET control card for each module name that is to be changed and submit it to the PANVALET Librarian.

++RENAME name1,name2

where:

name1 - The current module name.

name2 - New module name following the OS Naming Conventions.

3.2.2 ADD a COMMENT to a Library

Add a COMMENT to a library data set directory block.
(Reference: Page 27 of the User Reference Manual)

The COMMENT command enables the users to specify user-defined information to be associated with a library data set. This user-defined information becomes a data set's user-comment record.

The user-comment records are reported only on a PANVALET directory.

1) The job deck is assembled as follows:

```
//jobname JOB (idno,pms)
//stepname EXEC PAN1
//SYSIN DD *
++COMMENT name,description
/*
```

2) Control Card Description

JOB and EXEC cards are defined section 3.1.3 PANVALET Control Card Descriptions.

++COMMENT card: ++COMMENT name,description

name - Ten-character data set name as it appears on the directory.

description - Fifty characters of information are allowed to describe your data set. If specified, it replaces the information contained in the existing user-comment record. If omitted, the user-comment record is deleted from named data set.

3.2.3 COPY a data set

COPY a library data set on the Test Library.
(Reference: Page 27 of the User Reference Manual)

The COPY command is used to create a duplicate data set with a new name. The data set that is to be copied must exist in the library. The new data set will have a new name and is automatically placed in "TEST" and "ENABLED" status. In all other aspects, the copy is identical to the original data set, including the level number.

1) The job deck is assembled as follows:

```
//jobname JOB (idno,pms)
//stepname EXEC PAN1
//SYSIN DD *
++COPY name1,name2
/*
```

2) Control Card Description

JOB and EXEC cards are defined section 3.1.3 PANVALET Control Card Descriptions.

++COPY: ++COPY name1,name2

name1 - Name of the original data set as it appears on the directory.

name2 - New name of the duplicate data set to be created.

3.2.4 EJECT

EJECT to a new page on the printout.
(Reference: Page 28 of the User Reference Manual)

When a ++EJECT command is encountered, the printer file will advance to a new page. The command may be used in a PAN1 job step wherever one desires to have the printout to start on a new page.

3.2.5 IDENTIFICATION Comment

Put an IDENTIFICATION comment on title page of a PANVALET listing. (Reference: Page 28 of the User Reference Manual)

The ID command is used to print user identification on a separate page. When this command is executed, the output print file is advanced to the next form, a job identification page is written, and the print file is advanced to the following form. The user identification may be any combination of up to 75 characters.

This command can be used to cancel an existing temporary library data set which was created by a temporary update and can be used to turn off the OPTION INPUT command.

1) The job deck is assembled as follows:

```
//jobname JOB (idno,pms)
//stepname EXEC PAN1
//SYSIN DD *
++ID  identification
other PANVALET functions
/*
```

2) Control Card Description

JOB and EXEC cards are defined in section 3.1.3 PANVALET Control Card Descriptions.

++ID card: ++ID identification

identification - 75 byte string to be printed on the title page.

This command can be used in place of the ++EJECT if you desire a new page on the printout and a descriptive title for reference.

3.2.6 INSERT

This command will not be used at the present time.

3.2.7 Change the LEVEL number

The LEVEL number of a data set on the Test Library is a controlled function. This function is not effective at this time.

See section 4.3 Type II – Indirectly Available Services if you would like to have this service performed. (If absolutely necessary.)

3.2.8 RENAME a data set

RENAME a data set on the Test Library is a controlled function. See section 4.3 Type II – Indirectly Available Services if you would like to have this service performed.

3.2.9 SELECT a portion of a data set

SELECT a portion of a data set to an output file.
(Reference: Page 32 of the User Reference Manual)

This command is used to write a part of a data set onto an output file. The data set must be in the library. Through a series of "select" statements, partial data sets may be copied from the library and assembled into a composite data set on the output file.

1) The job deck is assembled as follows:

```
//jobname JOB (idno,pms)
//stepname EXEC PAN1 //SYSIN DD *
        (1 or more of each of the following SLECT cards)
++SELECT PRINT,dataset,level,from-stmt,to-stmt
++SELECT PUNCH,dataset,level,from-stmt,to-stmt
++SELECT WORK,dataset,level,from-stmt,to-stmt
/*
```

2) Control Card Descriptions

JOB and EXEC cards are defined in section 3.1.3 PANVALET Control Card Descriptions.

```
++SELECT:
                [ WORK   ]
++SELECT        [ PUNCH ], dataset,level,from-stmt,to-stmt
                [ PRINT  ]
        [ WORK, PUNCH, PRINT ]   - Output file.

        dataset  - Name as it appears on the directory.

        level    - Present level number of the data set.

        from-stmt - The statement number of first statement
                    of the selected portion of the data set.
```

to-stmt - The statement number of last statement of the selected portion of the data set.

If more than one data set is written to "WORK," the data sets will be concatenated.

3.2.10 Change the STATUS of a data set

To change the STATUS of a data set.
(Reference: Page 33 of the User Reference Manual)

Every data set in the library is described as either Production or Test and either Enabled or Disabled, and either Active or Inactive status. When a data set originally enters the library, it is put in "TEST," "ACTIVE," and "ENABLED" status.

1) The job deck is assembled as follows:

```
/ /jobname JOB (idno,pms)
//stepname EXEC PAN1
//SYSIN DD *
++STATUS dataset,status
/*
```

2) Control Card Descriptions

JOB and EXEC cards are defined in section 3.1.3 PANVALET Control Card Descriptions.

++STATUS card: ++STATUS dataset,status

dataset - Name as it appears on the directory.

status - The new status to be affixed to the data set.

[ACTIVE, INACTIVE, DISABLE, or ENABLE]

A production data set cannot be restored to "TEST" status. To do so, a COPY command must first be issued to create a "TEST" copy of the "PRODUCTION" data set.

3) Changing the status of a data set to PROD status is a controlled function. See section 4.3 Type II – Indirectly Available Services if you would like to have this service performed.

3.2.11 UPDATE a data set

To UPDATE a data set on the PANVALET Test Library.
(Reference: Pages 34-35.3 of the User Reference Manual)

The ++UPDATE command is used to update a data set that is in the library. The function of this statement is to identify the data set; the actual changes are accomplished through the use of one or more ++C, ++D and ++R commands following the "UPDATE" command. Only data sets that are in “TEST” and “ENABLED” status may be updated. A "PRODUCTION" data set may be temporarily updated using the "TEMP" parameter. The successful completion of an "UPDATE" is the only time the "DATE OF LAST MAINTENANCE" of a data set is altered, following the initial "ADD" to the library.

1) The job deck is assembled as follows:

```
 //jobname JOB (idno,pms)
//stepname EXEC PAN1
//SYSIN DD *
++UPDATE dataset[(,level) or (,0)][(,TEMP) or (,ALL)]
```
(1. Insert Statements)
```
++C n1
insert cards
```
(2. Delete Statements)
```
++C n2,n3
```
(3. Delete & Insert Statement)
```
++C n4,n5
insert cards
/*
```

Other "UPDATE" and CHANGE Subcommands:

```
++C  S1,(beg-col,fld)
++C  SeqXXY
++D  S2(,S3)
```

++R S4,(S5),/scan-fld/replace-fld/(,)(beg-col)(,end-col)

2) Control Card Descriptions

JOB and EXEC cards are defined in section 3.1.3 PANVALET Control Card Descriptions.

++UPDATE dataset [(,level) or (,0)][(,TEMP) or (,ALL)]

dataset - Name of the data set to be updated.

Level, or 0 - Present level number of the data set. This level will be increased by one each time the data set is permanently updated.

A "0" (zero) modification level may be specified for UPDATE...ALL commands only. This signifies that no level check occurs. However, the level will be incremented following a successful update.

,ALL The ALL command is used to indicate that all statements presently in a data set are to be replaced by the statements which follow the command in the input stream. All present rules for the UPDATE function remain unchanged; ++C commands are not allowed when this option is specified. The replacement statements are not printed, however, as in a normal update; a ++WRITE PRINT of the data set may be supplied for this purpose.

,TEMP The "TEMP" parameter is used to indicate that the changed copy of the data set is only to exist until: (1) a ++ID is encountered, (2) end of job, or (3) encountering any PAN#1 command that requires additional library space.

DO NOT execute two TEMP updates on the same data set in one job step.

++C sequence1[,sequence2]

This command is used for adding or deleting one or more consecutive statements beginning at any designated point in the data set. All ++C statements that apply to any single data set must immediately follow the "UPDATE" statement for that data set.

The first sequence number specified [sequence1] designates the exact point of change of the data set. Any records which follow this statement will be inserted into the data set immediately following the point of change until a record is encountered with a "++", "/*", "--", or "/&" in positions 1 & 2. (See example in 1) Job Deck under 3.2.11 Update a data set, *1. Insert Statements*.)

The second sequence number [sequence2] is optional. If specified, it will cause the deletion of all records in the current data set from the first sequence number to the second sequence number, inclusive. The second sequence number, if specified, must be equal to or greater than the first sequence number. (See example in 1) Job Deck under 3.2.11 Update a data set, *2. Delete Statements*. and 3*. Delete & Insert Statements*.)

++C seq1,(beg-col,fld)

To update specific columns within a statement:

seq1 - Designates the statement number to be modified.

beg-col - Specifies the column at which the change is to start.

fld - Specifies field to be inserted.

Example:

++C 101,(72,X) places an X in column 72 of statement 101.

Inserting a closed parenthesis to an updated field, a double closed parenthesis is

e.g.,

++C 223,(16,0(R2)),0) places the string "0(R2),0" into statement 223 beginning at column 16.

NOTES:

1. Only one column update card is allowed on a single statement.

2. Data is placed in the record in overlay mode. Attempts to update beyond the data field for the format will terminate the entire update.

3. Statements before and after the update are printed on the report.

4. Data cards following a column update ++C card will be inserted after the column update statement.

++C SEQxxy

When this parameter is specified on the first ++C command that follows an UPDATE command, the update will take place based on sequence numbers contained in the card images themselves. xx specifies the left end of the sequence field and y specifies the length of that field. The sequence field can contain only alphanumeric data and must be in ascending collating sequence. Data cards supplied with the input stream for insertion or replacement into the library data set must also be in ascending collating sequence based on the specified sequence field. Other ++C commands that are used in the same UPDATE with this parameter to accomplish range deletes must contain sequence numbers of length y.

++D seq1(,seq2)

The delete subcommand to delete statements from a data set. If one operand is used, only that sequence number is deleted. If two sequence numbers are used, the range of statements from the first sequence number through and including the second sequence number is deleted. In either case, all records, if any, that follow this subcommand will be inserted into the data set in place of the deleted statements.

++R seq1,(seq2),/scan-field/replace-field/(,) (beg-col)(,end-col)

The replacement update will perform a scan on the data set for the "scan" field specified and, if found, replace the "scan" field with the "replace" field. If only seq1 is specified, only that record will be scanned. If both sequence numbers are specified, that range of records will be scanned.

The scan can be limited to a portion of all records in sequence range specified. If beg-col is specified, the scan begins with that column and ignores all columns before that. If end-col is specified, all columns after it are ignored. Either/both may be specified.

The "scan" and "replace" fields need not be of the same length. If the "replace" field is shorter, the remaining positions of the record will be adjusted to the left to compensate. Conversely, if the "replace" field is longer, the remaining positions of the record will be adjusted to the right. If this forces any characters off the record (truncation), an error condition will occur and the entire update will be suppressed. Truncation, therefore, is not allowed.

If "scan" field is not specified, the "replace" field will overlay the beginning of the records specified (or as limited by beg-col). This is in effect a FILL of specific character string in desired columns. No adjustment right will occur; the overlayed characters are lost.

If the "replace" field is not specified, the "scan" field will be deleted for all occurrences. Adjustment to the left will occur. If no occurrences of the "scan" field are found, an error condition will exist and the entire update will be suppressed.

3.2.12 Change the USER

Change the USER identification code.
(Reference: Page 35 of the User Reference Manual)

This command is used to alter the user code of a library data set. The user code should reflect the identification number of the user presently maintaining the data set

1) Job deck is assembled as follows:

```
//jobname JOB (idno,pms)
//stepname EXEC PAN1
//SYSIN DD *
++USER dataset,user-code1,user-code2
/*
```

2) Control Card Descriptions

JOB and EXEC cards are defined in section 3.1.3 PANVALET Control Card Descriptions.

++USER card: ++USER dataset,user-code1,user-code2

dataset - Name of the data set whose user code is to be changed.

user-code1- Current user code related to the data set.

user-code2 - The new user code desired; must be numeric 0-9999. If omitted when a data set is initially added to the PANVALET Library, the user code will be set to "0" and will appear as spaces on the PANVALET Library directory list.

3.2.13 WRITE a data set

WRITE a data set onto an output file. (Reference: Pages 36-37 of the User Reference Manual)

This command is used to write an entire data set onto an output file. The data set must be in the library.

1) Job deck is assembled as follows:

```
//jobname JOB (idno,pms)
//stepname EXEC PAN1
//SYSIN DD *
++WRITE  [PRINT PUNCH or WORK],
        dataset,PREFIX=namekey
 /*
```

2) Control Card Descriptions

JOB and EXEC cards are defined in section 3.1.3 PANVALET Control Card Descriptions.

++WRITE card:

++WRITE [PRINT PUNCH or WORK],data set

PRINT, PUNCH, WORK - The description of the output file onto which the data set is to be written.

dataset - Name of the data set.

If more than one data set is written to "WORK", the data sets will be concatenated.

namekey - Using PREFIX=namekey causes data sets with a particular namekey of up to nine characters to be written.

Data set(s) may be written to a partitioned data set. The ++OPTION OUTPUT command will reroute the output from the

sequential work area to the PDS specified. The name of the PDS number will be equal to the first eight characters of the PANVALET name used.

1) The job deck is assembled as follows:

```
//JOB
//step  EXEC  PAN1
//ddname1  DD  DSN=PDS1,DISP=(,KEEP)
//ddname2  DD  DSN=PDS2,DISP=(,KEEP)
//sysin  DD  *
++OPTION OUTPUT,ddname1
++write work
++OPTION OUTPUT,ddname2
++write work
```

2) Control Card Descriptions

JOB and EXEC cards are defined in section 3.1.3 PANVALET Control Card Descriptions.

PDS card(s).

```
//ddname  DD  DSN = dsname,UNIT=type,
        VOL =SER=volume,DISP = disp,DCB= (_____)
```

ddname = Name on job control card that describes the PDS.

dsname = Name of partitioned data set.

type = 2314 or 3330.

volume = Volume serial number of the disk pack.

disp = (,KEEP) if it is a new data set.

++OPTION OUTPUT,ddname

ddname - Refers to the ddname in the JCL designating the output file.

3.2.14 INCLUDE a data set

INCLUDE a precoded data set within another data set. (Reference: Page 28 of the User Reference Manual)

The ++INCLUDE command may be stored as an embedded statement within a library data set in the PANVALET Library. During retrieval of a library data set to a WORK or PUNCH output file, a ++INCLUDE statement encountered will cause the data set specified by the INCLUDE to be located and retrieved from the PANVALET Library and written to the output file in place of the ++INCLUDE statement. Statements in the data set following the ++INCLUDE then continue to be retrieved to the output file as normal.

The command ++INCLUDE must begin in position 8 of the input record. This is the only command in the PANVALET system that does not begin in position 1. At least one blank must follow the command. The name parameter must follow the blank. The name parameter must be a valid data set name (1-10 alphanumeric characters) and may appear anywhere in positions 18 through 72, inclusive, in the input record.

If the specified name is invalid or not found in the library, the statement itself will be outputted and assumed not to be a valid command.

++INCLUDE can be used within another INCLUDE data set. Up to 6 nested includes are allowed.

If a data set containing ++INCLUDE command is in "PRODUCTION" status, all data sets to be included also must be in "PRODUCTION" status. However, "PRODUCTION" or "TEST" data sets may be ++INCLUDED by data sets in TEST status.

1) Control Card Description

++INCLUDE name:

++ - Denotes a PANVALET control card; must start in card column 8.

name - Name of the data set to be included.

2) Updating COBOL COPY Statements

PANVALET's ++INCLUDE statements simply copy a precoded module and include it into the program.

If the 01 level name in a program overrides the existing 01 level name in the copylib module, the following steps must be done:

Remove 0l level from copylib library module.

Update the source program by replacing the copy statement with:

01 level name.

++INCLUDE PANV.dataset-name

where:

level-name is the name used in the copy statement.

PANV.dataset-name is the 10-character name of the copylib module as listed in PANVALET.

If a "REPLACING" option is used in the copy statement, the following steps must be done.

If replacing refers to a 01 level name in the copylib module that is not the first data name coded (e.g., if two 01 levels are coded within the same copylib module and "replacing" references the second 01 level name), then:

Separate the copylib module and create 2 individual modules removing the 01 level.

Update the source program by replacing the copy statement with:

01 level name1.
++include module 1
01 level name2.
++include module 2

where:

level name1 is the original 01 level name used in the program.

module 1 is the name of the first copylib module as listed in PANVALET.

level name2 is the name used in the replacing option to override the second 01 level name of the original copylib module.

module 2 is the name of the second copylib module as listed in PANVALET.

If replacing is used to alter a dataname in the copylib module whose level number is greater than 01, then

If the 01 level name in a program overrides the 01 level name in the copylib module, remove 01 level from copylib module.

Update the source program by replacing the copy statement with:

01 level name. (01 level required only if copylib module starts with level number greater than 01.)

++include module-name 66 new-data-name renames old-data-name

3.2.15 FORMAT

Reference: Pages 29-30 of the User Reference Manual)

1) This command is used to change Format type of a data set.

Job deck is assembled as follows:

```
//jobname  JOB  (idno,pms)
//stepname  EXEC  PAN1
//sysin  DD  *
++FORMAT  dataset,type
/*
```

2) Control Card Description

JOB & EXEC cards are defined in section 3.1.3 PANVALET Control Card Descriptions.

++FORMAT card:

++FORMAT data set,type

dataset - Name of the data set in the library.

type - Format type to be changed to.

3.2.16 RESEQ

(Reference: Page 31 of the User Reference Manual)

1) This command is used to insert user sequence numbers in a data set.

Job deck is assembled as follows:

```
//jobname  JOB  (idno,pms)
//stepname  EXEC  PAN1
//sysin  DD  *
++RESEQ name,beg-col,length,beg-seq,incr
/*
```

2) Control Card Description

JOB and EXEC cards are defined in section 3.1.3 PANVALET Control Card Descriptions.

++RESEQ name,beg-col,length, beg-seq,incr

name - Name of the data set (no supersets).

beg-col - Position in record where sequence numbers are to be placed.

length - Length of the sequence field (maximum is 8).

beg-seq - Initial sequence number to be generated.

incr - Amount each successive sequence number is to be incremented.

3.2.17 SUPERSETS AND SUBSETS

An introductory definition; a superset is a library data set comprised of subsets where each subset identifies a group of data records within the superset.

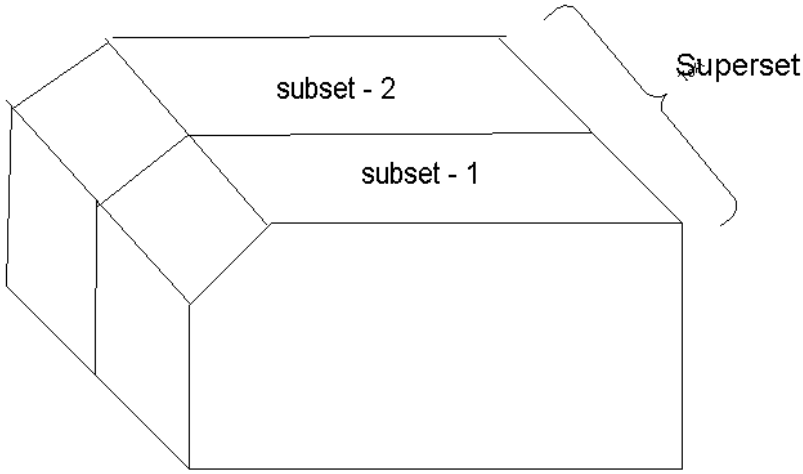

Figure 4

Two very significant capabilities of supersets and subsets are:

(1) Supersets and subsets will significantly increase library storage utilization by their ability to consolidate many small data sets into larger data sets defined as supersets. This should be of particular interest to users who are currently storing or planning to store an appreciable volume of JCL decks or similar small data sets.

(2) Supersets and subsets can be very useful in production applications where batches of keypunch data are involved. Supersets, being library data sets, would represent the card-image data files to application procedures. Subsets within a superset might represent batches of keypunch data. The advantage obtained in using supersets and subsets for the storage of card-image files is in the simplicity of creating and retrieving them and the level of documentation that will be provided.

Supersets should not be used for consolidating source modules or other similar data sets which would require updating via the ++UPDATE command. The current procedure to apply changes to a subset is to completely replace a subset with a new subset. See copying Supersets and Subsets.

Three PAN#1 commands are provided for defining and controlling supersets and subsets:

1) Control Card Description

The ALLOCATE Command

This command is used to define a new superset by entering into the library directory the data set "name" and "attributes." The format and function of the ALLOCATE command are very similar to that of the ADD command with two exceptions: no data can follow this statement and the data set defined is a superset and not a normal data set.

++ALLOCATE {superset-name} ,format–type ,NOFORMAT, user-code

++ALLOCATE The command to allocate a new superset to the library. The name and data set attributes of the superset are entered into the library directory, but no data set is created until the first subset is attached. The status of the superset will be "TEST,ENABLE." Only another PAN#1 command or the input stream delimiter (/*) can follow this statement; that is, no data records are input to the ++ALLOCATE command.

{superset-name} The name of the new superset and cannot be the name of any data set currently in the library. It may consist of any combination of up to 10 alphanumeric characters (A-Z and 0-9). (Use OS Naming Conventions.)

Format-type The types are:

,AUTOCODER	,BAL (or ALC)
,COBOL	,COBOL-72
,FORTRAN	,PL/I (or PL/1)
,RPG	,OBJECT
,JCL	,DATA
,USER780	,USER180
,OTHER	

The format and internal description of the superset or, more appropriately, the "format" of the subset(s) that will be attached to the superset. In all respects, the "formatting" of supersets and subsets is the same as the "formatting" performed when a data set is created by the ++ADD command. This is an optional parameter, and if omitted, the superset will be described as "UNSPECIFIED."

,NOFORMAT An optional parameter to override the language formatting that would otherwise be done according to the "internal description" of the superset. At the time a subset is attached, the entire card-image (80 columns) of the subset's data records will be stored into the superset. Of course, the data records will still be compressed by stripping off blanks. When the superset or an individual subset is retrieved and written onto the output PRINT file, generated sequence numbers will be inserted into positions 84-88 of each print record, positions 1-80 contain the original card-image. If written onto the output PUNCH or WORK file, no generated sequence numbers will be inserted into the individual records.

user-code The optional parameter for user-supplied information. The user-code may be any numeric value from 0 to 9999, and if omitted, zero is assumed. This field might be used to represent a project number, application code,

installation number, programmer number, etc. This field has no effect on the data set or the PANVALET Library but will be printed on the PANVALET Directory for future reference by the user. The user-code assigned to a superset can be changed at any time by the ++USER command.

The ATTACH Command

++ATTACH superset-name.subset-name ,{(LIST) or (SEQ)]

++ATTACH The command to attach a subset to a superset. The superset must currently be in the library and have a status of "TEST,ENABLE." All data records in the input stream following this statement until a record with a "++", "--", "/*", or "/&" in positions 1 and 2 is encountered, will be considered as belonging to the subset and will be written to the superset.

A subset cannot be attached to a library data set that was not originally defined as a superset.

{superset-name. subset-name} The parameter to identify the superset (superset name) to which the subset is to be attached and the name to be given to the subset (subset name). Subset names must be unique within a superset and can be any combination of alphabetic (A-Z) and numeric (0-9) characters of up to 10 positions. The two names must be separated by a period (.), thereby setting the maximum length of this parameter to 21 positions.

,LIST or ,SEQ An optional parameter to write the subset onto the output print file and optionally sequence check the input records as the subset is being added to the superset. Each input record is printed in its entire 80 column

card-image along with its new sequence numbers assigned by PANVALET.

LIST will perform only a print of the subset without sequence checking. SEQ specifies both the printing of the subset and the sequence checking of the input records according to the "internal description" of the superset. Any input records out-of-sequence will have an error message printed to the right of the record in error but will still be written to the superset.

The "DETACH" Command

++DETACH superset-name.subset-name

++DETACH The command to detach a subset from a superset. This command causes the subset data and the subset directory entry to be deleted from the superset as well as the library (the subset is physically gone from the library). The superset must have a status of "TEST,ENABLE" before a subset can be detached.

superset-name.subset-name This parameter identifies the subset within the superset to be detached. The two names must be separated by a period (.).

EXAMPLE: The following input stream to PAN#1 will create a superset named PAYROLLJCL, which is to initially contain two payroll JCL procedures:

```
++ALLOCATE PAYROLLJCL,JCL
++ATTACH PAYROLLJCL.PR001
$/
$/ JCL procedure for job named PR001
$/
++ATTACH PAYROLLJCL.PR002
$/
$/ JCL procedure for job named PR002
```

$/

One of the advantages of supersets and subsets as shown above is the simplicity in which small data files such as job control decks, test data, and program sub-routines are added to or deleted from the library. A superset can have up to 4095 subsets and, provided this value is not exceeded, there is no limit to the number of times the ++ATTACH and ++DETACH commands can operate on a superset.

Another advantage is the extended naming schemes possible with having two names instead of one, i.e., PAYROLLJCL.PR001 or PRTESTDATA.PR001.

Supersets and subsets are not necessarily limited to the control and storage of small data files; a superset can be as large as the number of data blocks available (maximum being 64K blocks) and each subset can have up to 65,535 statements.

2) Extended PAN#1 Commands for Supersets and Subsets

The following are PAN#1 commands that can be used to reference supersets and subsets.

Some of these commands have an extended format where a qualified name in the form "superset-name.subset-name" is used instead of the more familiar form "name." Otherwise, the format as well as the function of these commands is identical whether referencing a superset or a normal library data set.

Copying Supersets and Subsets

```
                {superset-name-1                {,superset-name-2
++COPY          {superset-name.subset-name      {,name
```

{name {,superset-name
.subset-name

The ++COPY command can be used to copy supersets or subsets to data sets or to copy data sets to subsets.

Supplying the parameters “superset-name-1, superset-name-2” will cause an exact copy of the original superset (name-1 to be made and given a new name (name-2).

Supplying the parameters "superset-name.subset-name, name" will take an existing subset and copy it into the library as a data set. For example:

++COPY SUPERJCL.SUBJCL1,PANJCL1

will create a new PANVALET data set called PANJCL1 using the data contained in subset SUBJCL1. The new data set's attributes (format, user code) are the same as in the originating superset, except that the level number will be 1 and status will be TEST, ENABLE.

Supplying the parameter "name,superset-name. subset-name" will take an existing data set and attach it with the subset-name to the superset. For example:

++COPY PANJCL1,SUPERJCL.SUBJCL1

will perform an attach or reattach of SUBJCL1 using the data contained in data set PANJCL1 (PANJCL1 cannot be a superset or subset).

Status of a Superset

```
                                { ,ACTIVE   }
                                { ,INACTIVE }
++STATUS   superset-name       { DISABLE   }
                               {,ENABLE    }
                                { ,PROD     }
```

Once a superset exists and is allowed to remain in a TEST and ENABLE status, subsets can be continually ATTACHed and DETACHed. By changing the status of a superset to "PROD", a superset becomes protected and therefore cannot be modified by future additions or deletions of subsets.

A superset can be tagged for subsequent deletion in PAN#2 by changing the status to DISABLE or INACTIVE (it can also be deleted by name).

Renaming Supersets or Subsets

The ++RENAME command can be used to change the name of a superset or the name of one of its subsets.

++RENAME superset-name-1 ,superset-name-2

To change a superset name (refer to the above format), specify in the first parameter the current superset name and in the second parameter the new name. The new name cannot be equal to any other library data set (or superset) name currently in the library.

++RENAME superset-name.subset-name-1,subset-name-2

The above format of the ++RENAME command is used to change a subset name. A qualified name (superset name followed by a period (.) followed by the current subset name) is entered in the first parameter. The second parameter only requires the new subset name that cannot be the name of another subset within the same superset.

User-comment

++COMMENT superset-name, up to 50 positions of user-defined information

The ++COMMENT command is a new command and is used to generate a user-comment record that is to contain user descriptive information that is to be associated with a library data set (or superset). A superset can have a

user-comment record added, changed, or deleted by submitting a ++COMMENT command.

User-code

++USER superset-name ,user-code-1 ,user-code-2

The ++USER command, just as it applies to normal library data sets, can be used to alter or eliminate a superset's user-code.

Retrieving Supersets and Subsets

```
                PRINT                       ,/*
++WRITE         PUNCH  ,super-set name      ,/&
                WORK   ,superset-name.subset-name,
                              up to 10 consecutive non-blanks
                                { superset-name }
'embedded'  ++INCLUDE  { superset-name.subset-name }
```

An individual subset or an entire superset (all of its subsets) may be retrieved via the ++WRITE or ++INCLUDE commands. The "name" parameter may be a qualified name (superset-name.subset-name) whereby a single subset is retrieved, or it can be just the superset-name that causes retrieval of all the subsets within the named superset.

When retrieving a superset, each of its subsets will be retrieved in subset name collating sequence.

If applicable to the format of the superset (COBOL, PL/1, BAL, etc), the sequence numbering of statements will be generated at the subset level; that is, the first statement of each subset will always begin with one whether retrieving a single subset or collectively (superset). However, if the "NOFORMAT" parameter was specified in the ++ALLOCATE command, then the original sequence numbers contained in the input data at the time the subset was added (ATTACHed) are retained and used.

The "embedded" ++INCLUDE command, as used in a normal library data set, can reference another library data

set, a superset, or an individual subset. This command can also be embedded as a statement within a subset and can reference any one of the following:

(a) another subset within the same superset
(b) a subset of another superset
(c) a superset
(d) a normal library data set

EXAMPLE: To create a superset containing record descriptions on the payroll master file to be used as systems application documentation.

```
++ALLOCATE PRFILE,DATA
++ATTACH PRFILE.PRREC01
        data records
++ATTACH PRFILE.PRREC02
        data records
++COMMENT PRFILE,PAYROLL MASTER FILE RECORD
            FORMATS
++WRITE PRINT,PRFILE
++WRITE PUNCH,PRFILE.PRREC02
```

3) The PANVALET Directory Listing

Perhaps the most important question concerning supersets and their subsets is how will they appear on a directory listing and be distinguished from normal data sets.

A superset will be easy to identify on a directory because of two distinct characteristics: the absence of a level number and the presence of a non-blank value printed under the column heading, "No. of Subsets".

When subsets are requested, they will be listed immediately following the associated superset with each subset-name prefixed by a period (.) and in alphanumeric sequence by name.

3.3 Examples Using PANVALET Functions

A combination of PAN1 control cards may be submitted in the same step.

Examples:

3.3.1 ADD a COBOL program to the Test Library

To add a COBOL program to the Test Library and compile after

```
//jobname  JOB  (idno,pms),'programmer-compile',
                MSGLEVEL=(1,1),CLASS=C //stepname1 EXEC PAN1
//SYSIN DD *
++ADD ABCD111SA1,COBOL,LIST,9999
        SOURCE DECK
++WRITE WORK,ABCD111SA1
/*
//stepname2 EXEC COBUC,PARM.COB='BUF=8K'
//COB.SYSIN DD DSN=&WORK,DISP=(OLD,DELETE)
/*
```

3.3.2 COPY a data set, ADD a comment

Copy a data set, add a comment, make a temporary update to the old data set, compile.

NOTE: "--WRITE WORK,__". The "--" in columns 1 and 2 indicates a conditional command that is processed only if the update is successful. This is highly recommended when doing a compile and load function. If the update is not successful, the job will terminate after the compile step and no program will be loaded to EDPD.LINKLIBT.

```
//jobname JOB (idno,pms),'programmer-control',
                MSGLEVEL=(1,1),CLASS=C
//stepname1 EXEC PAN1
//SYSIN DD *
++COPY ABCD111SA1,ABCD111SZ9
++COMMENT ABCD111SZ9,COPY OF ABCD111SA1-EDIT RUN
++UPDATE ABCD111SA1,1,TEMP
++C 2,2
        PROGRAM ID. NEWNAME
--WRITE WORK,ABCD111SA1
/*
//stepname2 EXEC COBUC,PARM.COB='BUF=8K'
```

```
//COB.SYSIN DD DSN=&WORK,DISP=(OLD,DELETE)
/*
```

3.3.3 Sample Use of PAN#1 ++ADD Command

```
++ADD PAY61,COBOL,SEQ,0120

        IDENTIFICATION DIVISION.
        PROGRAM-ID. 'PAY61'
        ++INCLUDE ENVIR
           '
           '
   STOP RUN.
++ADD ENVIR,COBOL,LIST
        ENVIRONMENT DIVISION.
            '
            '
        INPUT-OUTPUT SECTION.
/*
```

The above commands added two programs to the PANVALET Library. The first ++ADD adds a data set "PAY61" with a COBOL format, listing the source input, sequence checking the appropriate columns and giving this data set a user-code of 120. The second ++ADD adds a data set called "ENVIR" with a COBOL format and lists the data set without sequence checking.

```
++ADD  ABCOBJ,OBJECT

                ABC  OBJECT MODULE
```

PAN#1 will format the object module "ABC" by removing card columns 73-80 and regenerating sequence numbers upon retrieval in card columns 73-77. If the user had wished to retain columns 73-80 in this case, he would have coded: "++ADD ABCOBJ,OBJECT,NOFORMAT."

3.3.4 Sample Use of PAN#1 ++UPDATE Command

```
++UPDATE PAY61,1
++C 31
            MOVE SPACES TO PRINT-REC.
```

The above ++UPDATE command will modify data set "PAY61," which must be at modification Level 1. The ++C command indicates that the following source statement is to be inserted after sequence number 31. The updated data set "PAY61" will be at Level 2 upon successful completion.

```
++UPDATE PAY62,9
++C 101,101
++C 321,321
        CLC  WORK(2),=C'99'
        BE   ENDRTN
```

The above ++UPDATE command will modify data set "PAY62" which is at Level 9. The first ++C command indicates that sequence line 101 is to be deleted. The second ++C command indicates that sequence line 321 is to be deleted and replaced with two new source statements. The updated data set "PAY62" will be at Level 10 upon successful completion.

```
++UPDATE    ABC,4,ALL

              OBJECT MODULE
```

This command will replace the entire data set "ABC" with the following object module. The updated data set "ABC" will be at Level 5 upon successful completion. No listing of the input module will be produced. If the user desires a list, he must use the ++WRITE command.

```
++UPDATE  PAY61A,10
++C  2,(20,'PAY61A')
++C  31,(32,.)
```

The above ++UPDATE command will modify data set "PAY61A" which is at Level 10. The first ++C command indicates that sequence line 2 is to be column updated by placing the 8-character field "PAY61A" (the quotes are part of the field) beginning in column 20. The second ++C command performs a column update by placing a period in column 32 of sequence line 31. The updated data set "PAY61A" will be at Level 11 upon successful completion.

3.3.5 Sample Use of PAN#1 ++WRITE Command

```
++WRITE PRINT,PAY57
```

The above ++WRITE command followed by the "PRINT" parameter will print the data set named PAY57 on the system print unit.

++WRITE WORK,PAY57,/*

The above ++WRITE command followed by the work parameter and /* optional parameter will put the data set "PAY57" onto a work file (tape or disk) followed by a /* delimiter.

3.3.6 Sample Use of PAN#1 ++RENAME Command

++RENAME PAY57TST1,PAY57PROD

The above ++RENAME command changes the data set name from "PAY57TST1" to "PAY57PROD." The data set will now be referenced by the new name "PAY57PROD." The name "PAY57PROD" must be unique.

3.3.7 Sample Use of PAN#1 ++COPY Command

++COPY PAY57PROD,PAY57TST

The above ++COPY command creates a duplicate data set of "PAY57PROD" named "PAY57TST." The new name must be unique. The new data set, "PAY57TST," will have a status of TEST and ENABLE. Modifications may be applied to "PAY57TST."

++COPY TUESDATA,SUPDATA.TUESDAY

The above ++COPY command will enter a subset called "TUESDAY" in superset "SUPDATA" using the regular PANVALET data set "TUESDATA" as input.

```
++COPY  SUPDATA.THURSDAY,XTHURS
++UPDATE  XTHURS,1
++C  8,(40,RECEIPTS=286)
--COPY  XTHURS,SUPDATA.THURSDAY
```

The above sequence of command illustrates how changes can be incorporated into a subset since the ++UPDATE command is not valid with the superset-subset structure.

The first ++COPY creates a new PANVALET member called "XTHURS" from the subset "THURSDAY." The original superset remains unchanged. The new data set is created at Level 1, is placed in TEST, ENABLED status, and has the same language format as the originating superset.

The ++UPDATE command can be executed against the new member since it is a regular PANVALET data set, not a superset.

The last --COPY will replace the subset called "THURSDAY" with the newly updated member called "XTHURS," provided the ++UPDATE completed successfully. The data set "XTHURS" is not changed by the --COPY.

```
++UPDATE PAY61A,11
++D  27
++R  28,40,/FILE-FD/FILE-NAME/
```

The above update will delete statement 27 from the data set and replace all occurrences of "FILE-FD" with "FILE-NAME" in statements 28 through 40. The replacement update will adjust to the right all characters so that only the seven specified scan characters will be deleted.

3.3.8 Sample Use of PAN#1 ++EJECT Command

```
++EJECT
```

The above ++EJECT command advances the print form to the top of a new page. The command does not print on the output print unit.

3.3.9 Sample Use of PAN#1 ++ID Command

```
++ID  JOHN JONES  PHONE X379
```

The above ++ID command advances the print form to the top of a new page, prints "JOHN JONES PHONE X379" on a PANVALET identification page then advances to the following page.

3.3.10 Sample Use of PAN#1 ++USER Command

```
++ USER  ABC,360,370
```

This command will change the user code of data set "ABC" from 360 to 370. The current user code must always be provided.

3.3.11 Sample Use of PAN#1 ++STATUS Command

++ STATUS ABC,INACTIVE

The above command places data set "ABC" into an "INACTIVE" status. An "INACTIVE" status data set is quite similar to a "DISABLED" status data set. It may be relocated by PAN#2 to a protection file for temporary storage off-line. It may then be restored into the library at a future date.

3.3.12 Sample Use of PAN#1 ++SELECT Command

++ SELECT PRINT,PAY57PROC,15,1,135

The above ++SELECT command followed by the "PRINT" parameter will selectively print statements 1-135 of the data set "PAY57PROD" at Level 15. The level number must be the current level of the data set named.

3.3.13 Sample Use of PAN#1 ++COMMENT Command

++ COMMENT PAYTAX022, STATE TAX RTN EFFECTIVE 12/16/00

The above command will create or replace a user-comment to data set "PAYTAX022." This user-comment will appear on the directory listing.

3.3.14 Sample Use of PAN#1 ++ALLOCATE Command

++ ALLOCATE SUPDATA,DATA

The above command defines a special type of data set called a superset. No data is entered into the member with this command. Subsets are entered in this data set with the ++ATTACH or ++COPY commands. All records entered will be stored in DATA format.

3.3.15 Sample Use of PAN#1 ++ATTACH Command

++ ATTACH SUPDATA.MONDAY

DATA FILE

This command will enter a subset called "MONDAY" into the superset called "SUPDATA."

3.3.16 Sample Use of PAN#1 ++RESEQ Command

```
++ RESEQ  PAY75B,65,6,10,5
```

The above will insert sequence numbers in positions 65-70 of each statement in PAY75B. The first sequence number will be 10 with each subsequent field being incremented by 5.

3.3.17 Sample Use of PAN#1 ++FORMAT Command

```
++ FORMAT  PAY57,PL/1
```

The above changes PAY57 to PL/1 format specifications. All statement change level stamps will be lost.

3.3.18 Sample Use of PAN#1 ++OPTION Command

```
++ OPTION INPUT,DDN
++ADD  PROG1,BAL,LIST
++UPDATE  PROG2,17,ALL
```

The above ADD command will attempt to receive its input from a PDS described by the ddname "DDN." PDS member "PROG1" will be read as input for the ADD. Also, the UPDATE...ALL will attempt to receive its input from "DDN" and member "PROG2" will be read as input.

```
++OPTION OUTPUT,NEWPDS
++WRITE  WORK,PAY57
++WRITE  WORK,PAY58
++WRITE  WORK,PAY59
```

The above data sets (PAY57, PAY58, and PAY59) will be written as individual members to a PDS described by the ddname "NEWPDS." The same result would occur with:

```
++OPTION OUTPUT,NEWPDS
++ADD  CALLIT,JCL
     ++INCLUDE  PAY57
     ++INCLUDE  PAY58
```

```
    ++INCLUDE  PAY59
++WRITE  WORK,CALLIT
```

Only one PDS member will be created with:

```
++OPTION OUTPUT,NEWPDS,NEWMEM
++SELECT WORK,CALLIT,1,1,9999
```

It will have the PDS member name of NEWMEM.

3.4 Scanning the Library for a "Character String"

3.4.1 Use of PAN#8 Library Statements

PAN#8, at the discretion of management, may be available for use by the programming staff. This program, referred to as "The Library Scan Program", is a tool of the PANVALET system which allows a program to be searched for the presence of a specific field. An option within the scan program allows the search field to be replaced, data to be inserted, or data to be deleted. If no search field is specified, data can be deleted from or generated into a data set, data sets of one language type, or the entire library.

In the case of security controlled data sets, either a ++CONTROL or ++ACCESS statement will be required to allow scanning.

PAN#8 is used to:

1) Search an individually named data set for the occurrences of a character string supplied by the user. Each occurrence of the character string is listed showing the entire card-image statement, statement number, and name of the library data set in which the character string was found, and level number of the data set.

2) JCL to Use PAN#8:

```
//jobname JOB (idno,pms)
// stepname EXEC PANSCAN
//PANIN DD *
++SCAN name,'scan-field',beg-stmt,end-stmt,beg-col,end-col
++SCAN  name
++REP  /scan-field/replace-field/
```

/*

3) Files Used in the SCAN Function

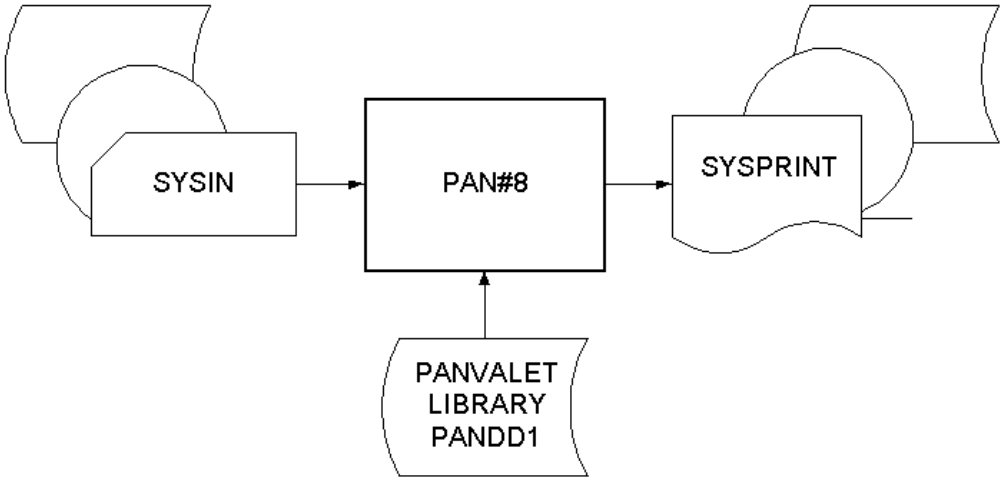

Figure 5

4) Control Card Description

The "SCAN" Statement

++SCAN {name} {,'scan-field'}

[{,} [beg-stmt] {,} [end-stmt] {,} [{beg-col,end-col}]]

NOTE: The above library statement must fit on one card (may extend to column 72).

++SCAN The command to invoke a search of a specified area of the library (according to the first parameter) for statements containing the user-supplied character string.

{name} This parameter is required and specifies the area of the library to be searched.

If the name of a data set is specified, then just that data set is searched for the character string.

{'scan-field'} A character string with a beginning and ending delimiter which will be used as an argument for searching

programs for an exact match. The scan field may be any length from 1 to 61 character exclusive of the delimiter characters. The delimiter characters may be any character and may not be contained within the character string. (Arbitrarily, apostrophes were used as delimiters in the parameter format shown on the left.)

{,beginning-stmt} Optional parameter (must be numeric) to specify the statement number of the first statement to begin the search within a data set. Those statements with a lower statement number are bypassed without inspection.

{,ending-stmt} Optional parameter to specify by number the last statement to be searched. If specified, must be greater than the "beginning-stmt" number. Statements with a higher statement number are bypassed.

{,beginning-column} The optional parameter to specify by number the first column of each statement to begin the search. Columns to the left of the specified column number are ignored.

{,ending-column} An optional parameter to specify the last column number to be searched, and if specified, must be greater than the "beginning-column" number. Columns to the right of this column are not inspected. The number of columns to be searched (inclusive of the beginning and ending columns) must be equal to or greater than the length of the "scan-field."

The four optional parameters are numeric values and are positional parameters. If used, they must be coded in the above order and each separated by a comma. Two commas must be used to indicate the omission of a parameter.

The user may supply as many different ++SCAN statements in any one execution of PAN#8 as desired. Each statement is processed independently.

The "REP" Statement

++REP /scan-field/replace-field/

NOTE: The above library statement must fit on one card (may extend to column 72).

++REP The command to replace a specified character string (scan-field) with a specified character string (replace-field) within a library area delimited by the last previously encountered ++SCAN command. This command is subject to all of the same restrictions and functions as the PAN#1 ++UPDATE command. The level number may be incremented more than once in a single execution of PAN#8 if a program is changed more than once. (See PANVALET User Reference Manual.)

/scan-field/replace-field/ Two character strings with beginning and ending delimiters which will be used as an argument for searching library data sets for an exact match and replacement. The combined length of the 2 fields must not exceed 63 characters exclusive of delimiter characters and must be short enough to fit within ++SCAN column delimiters for the language type being processed. Any delimiting character may be used as long as it is not contained within either character string. (Arbitrarily, slashes were used as delimiters in the parameter format shown above.)

The character strings may be of equal or unequal length as required.

If the scan-field is longer than replace-field, all data within the language type column delimiters and located to the right of the scan-field will be shifted leftward to be immediately to the right of the replace-field. Blank filling on the right will be done up to the right side column delimiter. If end-col is specified, data to the right of it remains the same and is not shifted.

If the scan-field is shorter than the replace-field, all data within the language type column delimiters and located to the right of the scan-field will be shifted rightward as far as necessary to allow the insertion of the replace-field. However, if the rightward shift would result in loss of non-blank characters at the right side column delimiter, an

informational message "REP NOT DONE - truncation will occur" will be printed immediately below the record being processed. No change will be made up to that particular statement and processing will resume with the next record.

If the scan-field and the replace-field are of equal length, the replace will always be done subject to the above restrictions.

Either argument, but not both, may be eliminated by placing two delimiter characters immediately next to each other. When this is done, two important variants of the ++REP command are invoked.

//replace-field/ When the scan-field is eliminated, the replace-field is inserted at the beginning column specified either by the preceding ++SCAN command or by the default beginning column for the language being processed.

/scan-field// When the replace-field is eliminated, the scan-field is simply deleted whenever it is found and all data to the right (within the columns specified on the SCAN statement) is shifted left.

5) Examples Using ++SCAN and ++REP

Scanning the Library for a Specified Field.

The following examples illustrate some of the possible uses of the ++SCAN statement:

Example 1 - A source program named NIM600V5 is to be scanned for all occurrences of the character string 'PUT~~bbb~~SYSPRINT':

```
++SCAN  NIM600V5,'PUT~~bbb~~SYSPRINT'
/*
```

Example 2 - Search two programs named NIM400V2 and NIM500V4 for all occurrences of the operand X'FF'. In this case, the apostrophes appear within the character string so $ are used for delimiters:

```
++SCAN  NIM400V2,$X'FF'$
```

```
++SCAN  NIM500V4,$X'FF'$
/*
```

Example 3 - Search an assembler program named BAL for DCB file definitions. To void such fields as DCBLRECL, the character string DCB must be preceded and followed by a blank. Do not scan past column 30 for DCB because they probably are not the DCB macro instruction.

```
++SCAN  BAL,'␣DCB␣',,,,30
   /*
```

Note the use of the four consecutive commas to indicate the omission of the first three positional parameters.

Replacing Character Strings within the Library

The following examples illustrate some of the possible uses of the ++REP statement:

Example 1 - An assembler language source program name ORDENT36 is to have certain macros replaced by other macros. The search will be limited to columns 2-30 to eliminate changing statement labels and comments.

```
++SCAN  ORDENT36,,,,2,30
++REP  /GETMAIN/DFHGETMAIN/
++REP  /FREEMAIN/DFHFREEMAIN/
++REP  'OPEN'DFHOPEN'
++REP  $CLOSE$DFHCLOSE$
```

Example 2 - An assembler program named A40MCD26 contains an MZ as two characters of each statement label. Readability could be improved by deleting these two characters.

```
++SCAN  A40MCD26
++REP  /MZ//
```

4 INDIRECTLY AVAILABLE SERVICES

This section describes those services that are indirectly available to

programmers through the PANVALET Librarian or the PANVALET Operations Officer. To request these services, the programmer must complete a worksheet, punch the appropriate PANVALET control cards, and submit both the worksheet and the control cards to either the Librarian or the Operations Officer.

Requests submitted to the PANVALET Librarian will be consolidated during the day. Then, at the end of each day, all requests will be batched and processed. Thus, all requests for services submitted during the day will be processed and ready at the start of the next workday.

This section describes each function and gives instructions on how to call for each service.

4.1 General Information

4.1.1 Types of Indirect Services

There are two types of indirect services. The first may be requested by the programmer whenever he needs them. To request these services, the programmer must complete PANVALET Worksheet I, punch the appropriate PANVALET control cards, and submit them to the PANVALET Librarian. The Librarian will consolidate all requests submitted during the day. Then, at the end of the day, all requests will be batched and processed.

The second type of indirect services is PAN1 functions that are controlled to protect the Test Library and to monitor the effectiveness of the system. Therefore, to request these services, the programmer MUST complete PANVALET Worksheet II and get the approval of the Application Systems Development Services Branch Chief. If the Branch Chief approves the request and signs the worksheet, then punch the appropriate PANVALET control cards and submit the worksheet and cards to the PANVALET Librarian.

4.1.2 Emergency Requests for Services

Under certain very special circumstances, the services listed on PANVALET Worksheets I and II may require immediate action. The procedure to obtain immediate action is to get the request authorized by the Application Systems Development Services Branch Chief or his designated representative.

Approval for emergency service will be indicated by the authorized signature in the space provided at the bottom of PANVALET Worksheets I and II.

Note that Section 4 (move from PRODUCTION to TEST) of PANVALET Worksheet I is duplicated on PANVALET Worksheet II.

For routine transfers, use PANVALET Worksheet I.

For emergency transfers (from the Production to the Test Library), use PANVALET Worksheet II only.

4.2 Type I - Indirectly Available Services

These services are requested on PANVALET Worksheet I via the PANVALET Librarian. The following is a description of these services.

4.2.1 ADD a data set to the Production Library

To add a source deck to the Production Library, complete Section I of PANVALET Worksheet I. Supply the appropriate cards as listed below.

1) Case #1: Data set is a card deck.

Cards to submit:

```
++ADD name,lang-type,LIST,idno
Source deck
```

where:

name - Name of the data set to be added to the Production Library.

lang-type - Must be specified, one of the following:

AUTOCODER	BAL (or ALC)
COBOL	COBOL-72
FORTRAN	PL/1 (or PL/I)
RPG	OBJECT
JCL	DATA
OTHER	

Idno - Specify user identification number of the person responsible for the data set.

2) Case #2: Data set is a member of a Partitioned Data Set (PDS)

Cards to submit:

```
//PDS DD DSN=dsname,UNIT=type,
       VOL=SER=volume,DISP=SHR
++OPTION INPUT,PDS
++ADD member-name,lang-type,LIST,idno
++RENAME member-name,new-name
```

The parameters of the ADD control cards are defined above.

```
++RENAME member-name,new-name
```

member-name - Is the name as it is used on the PDS.

new-name - This will be the name of the data set in the PANVALET Library. Follow the OS Naming Conventions, PMSJaJnSN"S"VaVn.

Only one //PDS card and one ++OPTION card need be submitted. Submit as many ++ADD and ++RENAME cards as required, one for each member to be transferred from the PDS to the PANVALET Production Library.

3) Case #3: Data set is a sequential data set on disk or tape.

Cards to submit:

```
++ADD name,lang-type,LIST,idno
//DD DSN=dsname,UNIT=type,
        VOL=SER=volume,DISP=SHR
```

The parameters of the ADD control card are defined above.

dsname - Data set name of the sequential file.

type - 2314, 3330 or 2400.

volume - Volume serial number of the disk pack or tape.

Submit a set of ++ADD and // DD cards for each data set to be added.

4.2.2 DELETE a data set from the Production Library

To delete a data set in the Production Library, complete Section II of PANVALET Worksheet I and submit with required items to the PANVALET Librarian. You will submit a card to disable the data set on the Production Library. Once a week a job will be submitted by the PANVALET Librarian to delete the disabled data sets from the library and save them on a backup tape.

Cards to submit:

```
++STATUS name,DISABLE
```

where:

name - Name of the data set to be deleted.

4.2.3 MOVE a data set from the Production to Test Library

When moving a data set from the Production Library to the Test Library, complete Section III of PANVALET Worksheet I and submit required items to the PANVALET Librarian.

At the completion of the MOVE, a copy of the data set will be on the Test Library. The original and the copy of the data set on the Production Library are disabled and will be deleted later.

Cards to submit:

PT old-name new-name

where:

PT - Identifies PROD to TEST Library move (c.c. 1-2)

old-name - Name as it exists on the library (c.c. 5-14)

new-name - Name of the duplicate data set to be created (c.c. 20-29).

When copying a data set, the new-name is formed by incrementing the last digit (10th character) by one, from one to nine.

For example:

++COPY PMS1A1SA1,PMSA1A1SA2

Use as the ninth and tenth characters A1-A9, B1-B9, ... Z1-Z9 and start again with A1. The first eight characters always remain the same.

4.2.4 Move a data set from the Test to Production Library

When moving a data set from the Test Library to the Production Library, complete Section IV of PANVALET Worksheet I and submit required items to the PANVALET Librarian.

At the completion of the MOVE, a copy of the data set will be on the Production Library and will be in the Production status. The copy of the data set on the Test Library is disabled and will be deleted.

Cards to submit:

TP name

where:

TP - Identifies TEST to PROD Library move (c.c. 1-2).

name - Name of the data set being moved (cc. 5-14).

4.2.5 RESTORE a data set to Test Library

When restoring a data set from the Delete File tape to the Test Library, complete Section V of PANVALET Worksheet I and submit required items to the PANVALET Librarian.

Check the Delete File Directory listing for the contents of the backup tapes and locate the data set desired. The restore will add the data set back to the on-line Test Library provided that a data set with the same name does not exist on the library.

Cards to submit:

++RESTORE name

where:

name - Ten-character name of the data set.

4.3 Type II - Indirectly Available Services

These services are requested on PANVALET Worksheet II via the PANVALET Librarian and require the approval of the Application Systems Development Services Branch Chief. The following is a list of these services.

4.3.1 Changing the LEVEL number

When changing the level number of a data set on the PANVALET Library, complete Section I of PANVALET Worksheet II, and if authorized, supply the required items to the PANVALET Librarian.

Cards to submit:

++LEVEL name,old-level,new-level

where:

name - Name of the data set.

old-level - Current level number.

new-level - New level number 1.

4.3.2 RENAME a data set on the PANVALET Library

When renaming a data set on the PANVALET Library, complete Section II of PANVALET Worksheet II, and if authorized, supply the required items to the PANVALET Librarian.

Card to submit:

++RENAME old-name,new-name

where:

old-name - Current name of data set as it exists on library.

new-name - Name to be changed to.

4.3.3 Change the STATUS of a data to "PROD."

When changing the status of a data set to PRODUCTION, complete Section III of PANVALET Worksheet II, and if authorized, supply the required items to the PANVALET Librarian.

Cards to submit:

++STATUS name,PROD

where:

name - Name of the data set.

5 ERROR RECOVERY AND BACKUP of TEST

5.1 Error Recovery if PANVALET Test Library if Destroyed

5.1.1 Notification of Analysts and Programmers

1) The Librarian will notify the stenos in the Applications Branch and Administration Section.

2) The stenos, in turn, will notify the programmers and analysts.

Call project leaders.

Post notice on bulletin boards.

3) The project managers must notify any non-EDP personnel (contractors, consultants, etc.) with whom they are working.

5.1.2 PANVALET Jobs

1) The PANVALET Librarian will restore the Test Library to its original condition (i.e., its condition at the beginning of the first shift).
2) Users

PANVALET jobs that were run prior to destruction of the Test Library - Users must resubmit all jobs that updated the PANVALET Test Library between the start of the first shift and the time that the Test Library was destroyed.

Pending PANVALET jobs - All jobs in the input queue that will utilize data sets on the PANVALET Test Library that were updated during the day by a previous job prior to the destruction of the Test Library must be placed in "Hold Status" until the job step that updated the data set has been resubmitted and rerun. Therefore, users who have such jobs pending at the computer center must notify the PANVALET Librarian to put those jobs in the hold status as soon as possible.

3) As soon as they are notified, operators will place these jobs in the "Hold Status" until the resubmitted jobs have been run. They will then release these jobs from the "Hold Status."

5.2 Backup

5.2.1 Frequency and Effect on the User

1) Test Library - The Test Library will be backed up every morning, Mon thru Fri.

All permanent updates made during the day should be kept for recovery purposes until the next backup is made on the following

morning. After each backup, a directory is printed. When the directory arrives at its destination, update cards may be discarded.

2) Production Library - Backup of the Production Library is scheduled for every Wednesday providing there has been some activity affecting the library.

5.2.2 Generations

1) There are five generations of the Test Library.

2) There are two generations of the Production Library.

P A N V A L E T W O R K S H E E T I

INSTRUCTIONS

A. For each function:
   1. Arrange sets of control cards in ascending sequence by data set name
   2. Control cards must be in the order shown below.
   3. Specify the number of control card sets being submitted in the space provided at the right.

For urgent requests, check below:

B. Submit this form and PANVALET control cards to PANVALET Librarian.

C. c For more details, refer to the PANVALET User's Guide.

PROGRAMMER:	EXTENSION:	DATE:	NO. OF SETS

I. ADD a source program to the Production Library.

II. A. If source program storage medium is CARDS, the items required are:
++ADD name,lang-type,LIST,user-id number
Card Deck

I. B. If the source program storage medium is a PARTITIONED DATA SET, the items required are:
//PDS DD DSN=dsname,UNIT=type,VOL=SER=volume,DISP=SHR
++ADD name,lang-type,LIST,user-id number
++OPTION INPUT,PDS
++RENAME name,name2

I. C. If the source program storage medium is a SEQUENTIAL DATA SET, the items required are:
++ADD name,lang-type,LIST,user-id number
// DD DSN=dsname,UNIT=type,VOL=SER=volume,DISP=SHR

II DELETE a source program from the Production Library.
++STATUS name,DISABLE

III. MOVE a source program from the Production Library to the Test Library.

1	5-14	20-29
PT	old-nameXX	new-nameXX

IV. MOVE a source program from the Test Library to the Production Library.

1	5-14
TP	old-nameXX

V. RESTORE a source program from the Deleted Data Set tape to the Test Library. (Use this function only to recover data sets not on the libraries)

++RESTORE name
++STATUS name,ENABLE

If immediate service is necessary, state reason for request and obtain authorization before submittal to PANVALET Librarian.

Reason for urgency.

Request for EMERGENCY services approved by

Signature

8/74

FOR EMERGENCY USE

PROGRAMMER
EXTENSION
DATE

PANVALET WORKSHEET II

INSTRUCTIONS

A. Fill in requested information:
   1. Programmer's name, extension number, and date.
   2. For each function requested, specify the number of data sets to be affected in the space provided on the right.
B. For controlled functions, go to Section I below. (Authorization NOT required.)
C. For emergency transfer, go to Section II below. (REQUIRES authorization.)

I. CONTROLLED FUNCTIONS	NO. OF DATA SETS
I. LEVEL number modification on the TEST Library. ++LEVEL dataset-name,old-level,new-level	
II. RENAME datasets on the TEST Library. ++RENAME old-name,new-name	
II. STATUS change from 'TEST' to 'PROD' on the Test Library. +STATUS dataset-name,PROD	

PUNCH required control card(s) (label the decks with the functions) and submit the form and the card(s) to the PANVALET Librarian at the computer center.

II. EMERGENCY TRANSFER

A. State the reason for this request.
B. Obtain authorization from DEVELOPMENT BRANCH CHIEF (or other authorized personnel).
C. If request is granted, punch control cards (label the decks with 'MOVE P-T') and submit form and cards to the PANVALET Librarian at computer center.

			NO. OF DATA SETS
MOVE a source program from the Production Library to the TEST Library.			
1	5-14	20-29	
PT	old-nameXX	new-nameXX	

Reason for emergency request:
X
Date

Signature of authorized personnel

8/74

6 COMMANDS THAT REQUIRE CAREFUL ATTENTION

Adding data sets from sequential or partitioned data sets:

The JCL cards must be arranged in the order illustrated; otherwise, results are unpredictable.

Commands that use the WORK area:

If more than one write to work command is processed in a single job step, the two data sets will be concatenated. This can be beneficial for some jobs but for compilations use only ONE ++WRITE WORK,---.

Temporary updates:

1) To the same data set:

   Do not attempt two or more temporary updates to the same data set in one step. In a specific instance when two temporary updates were attempted, the directory pointers for the data set were altered in such a way that data set became inaccessible.

2) To two or more data sets:

   When two data sets are temporarily updated, the following occurrences take place. The first update uses space on the library to store the temporarily updated data set. When the second temporary update is executed, the resulting data set is stored in the same area allocated to store the results of the first temporary updated data set overriding the results of the first temporary update.

   An example where this may occur is:

   Temporary update INCLUDE module.

   Temporary update data set.

   Write data set to work.