GC26-3902-1
File No. S370-32

Systems

OS/VS2 MVS Utilities

Release 3.8

Includes Selectable Units:

Data Management	**VS2.03.808**
System Security Support	**5752-832**
Data Management Support	**5752-860**

IBM

Second Edition (September 1983)

This is a reprint of GC26-3902-0 incorporating changes released in the following Technical Newsletters:

GN26-0926 (dated 31 January 1979)
GN26-0930 (dated 29 June 1979)
GN26-0982 (dated 10 October 1980).

This edition applies to Release 1.0 of Data Facility/Device Support, Program Product 5740-AM7, as well as to Release 3.8 of OS/VS2 MVS, and to any subsequent releases until otherwise indicated in new editions or Technical Newsletters.

Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 and 4300 Processors Bibliography,* GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Publications are not stocked at the address given below. Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments has been provided at the back of this publication. If the form has been removed, address comments to IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California 95150. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

HOW TO USE THIS PUBLICATION

This publication describes how to use the OS/VS utility programs. To use this book, you should be familiar with VS terms and concepts.

In addition to the preface you are now reading, a table of contents, and a list of figures, this publication has the following major parts:

Title	**Function**
"Summary of Amendments"	an abstract of the major technical changes reflected in this and previous editions.
"Introduction"	a summary of the utility programs and information on the differences among system, data set, and independent utility programs. This chapter contains basic information about how the programs are executed and about the utility control statements used to specify program functions. New or infrequent users of the utility programs should give particular attention to this chapter.
"Guide to Utility Program Functions"	a table, arranged in alphabetic order, of utility program functions and the programs that perform them. This table enables you to find the program that can do what you need to have done.
Utility Programs	individual chapters for each utility program arranged in alphabetic order. For a discussion of the organization of these chapters, see "Organization of Program Descriptions" below.
"Appendix A: Exit Routine Linkage"	information about linking to and returning from optional user-supplied exit routines. This appendix should be read only if you plan to code or use an exit routine. If you are coding an exit routine, this appendix provides linkage conventions, descriptions of parameter lists, and return codes. If you are using an existing exit routine, you may be interested in the meaning of return codes from the exit routine.
"Appendix B: Invoking Utility Programs from a Problem Program"	description of the macro instructions used to invoke a utility program from a problem program rather than executing the utility program by job control statements or by a procedure in the procedure library. This appendix should be read only if you plan to invoke a utility program from a problem program.
"Appendix C: DD Statements for Defining Mountable Devices"	a review of how to define mountable volumes to ensure that no one else has access to them. For a definitive explanation of this subject, see *OS/VS2 JCL,* GC28-0692.

"Appendix D: Processing User Labels"	description of the user-label processing that can be performed by IEBGENER, IEBCOMPR, IEBPTPCH, IEHMOVE, IEBTCRIN, and IEBUPDTE. This appendix should be read only if you plan to use a utility program for processing user labels.
"Index"	a subject index to this publication.

Organization of Program Descriptions

Program descriptions are all organized, as much as possible, in the same way to enable you to find information more easily. Most programs are discussed according to the following pattern:

- Introduction to and description of the functions that can be performed by the program. This description typically includes an overview of the program's use, definitions of terms, illustrations, etc.
- Functions supported by the utility and the purpose of each function.
- Input and output (including return codes) used and produced by the program.
- Control of the program through job control statements and utility control statements. Explanations of utility control statement parameters are presented in alphabetic order in tabular format, showing applicable control statements, syntax, and a description of the parameters. Any general information, restrictions, and relationships of a given utility control statement to other control statements are described in the sections concerning the statements or in the section for restrictions.
- Examples of using the program, including the job control statements and utility control statements.

Required Publications

The reader should be familiar with the following publications:

- *OS/VS Message Library: Utilities Messages,* GC38-1005, which contains a complete listing and explanation of the messages and codes issued by the utility programs.
- *OS/VS2 JCL,* GC28-0692, which contains a complete explanation of the job control statements available for the operating system.
- *OS/VS2 MVS Data Management Services Guide,* GC26-3875, which describes the input/output facilities of the operating system. It contains information on record formats, data set organization, access methods, direct access device characteristics, data set disposition, space allocation, and generation data sets.
- *OS/VS2 Supervisor Services and Macro Instructions,* GC28-0683, which contains information on how to use the services of the supervisor. Among the services of the supervisor are program management, task creation and management, virtual storage management, and checkpoint and restart.
- *OS/VS2 MVS Data Management Macro Instructions,* GC26-3873, which contains a description of the WRITE SZ, LINK, and RETURN macro instructions, and contains the format and contents of the DCB.

Related Publications

The additional publications referred to in this publication are:

- *OS/VS2 Data Areas,* SYB8-0606, which contains a complete description of the control blocks used by the operating system.
- *IBM System/370 Principles of Operation,* GA22-7000, which contains a description of system structure; of the arithmetic, logical, branching, status switching, and input/output operations; and of the interruption system.
- *OS/VS Mass Storage System (MSS) Services: General Information,* GC35-0016, which contains information on the copy or restore of a staging volume.
- *OS/VS2 Access Method Services,* GC26-3841, which contains information on generation data groups and SMF record types 63 and 67.
- *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide,* GC26-3838, which contains information on cataloging VSAM data sets.
- *OS/VS2 System Programming Library: Data Management,* GC26-3830, which contains information on data set password protection.
- *IBM 50 Magnetic Data Inscriber Component Description,* GA27-2725, which contains information on the MTDI cartridge used by the IBM 2495 Tape Cartridge Reader (TCR) when used by the IEBTCRIN utility program.
- *Using OS Catalog Management with the Master Catalog: CVOL Processor,* GC35-0010, contains VSAM catalog management details.
- *OS/VS2 MVS CVOL Processor,* GC26-3864, has detailed information regarding CVOL processing with the MVS VSAM master catalog.
- *OS/VS Message Library: VS2 Utilities Messages,* GC26-3920, contains error messages and corrective actions.
- *OS/VS2 Conversion Notebook,* GC28-0689, contains information on program authorization (APF) in the OS/VS2-MVS environment.
- *Device Support Facilities,* GC35-0033, describes initialization and maintenance of direct access storage devices (DASD).
- *Data Facility/Device Support: User's Guide and Reference,* SC26-3952, has detailed information on processing DASD volumes with indexed VTOC.
- *Data Facility/Data Set Services: User's Guide and Reference*, SC26-3949, describes DASD utility functions such as dump or restore, and reduction or elimination of free space fragmentation.

Utilities Not Explained in This Book

There are several specialized utilities not discussed in this book. The following list shows their names, functions, and what book contains their explanation.

Utility	Function	Reference
IEBIMAGE	Allows the user to define, modify, print, or link modules for use with the IBM 3800 Printing Subsystem.	*IBM 3800 Printing Subsystem Programmer's Guide,* GC26-3846.
IDCAMS	Allows users to define, manipulate, or delete VSAM data sets, define and manipulate VSAM Catalogs, and copy, print, or convert SAM and ISAM data sets to VSAM data sets.	*OS/VS2 Access Method Services,* GC26-3841
IAPAP100	Examines user data and drive characteristics of IBM 3344 and 3350 DASD for errors. If AP-1 detects an error, it prints a message to the operator, and diagnostic information to the system's printer.	*OS/VS and DOS/VS Analysis Program–1 (AP-1) User's Guide,* GC26-3855
Device Support Facilities	To be used for the initialization and maintenance of DASD volumes.	*Device Support Facilities,* GC35-0033.
Data Facility/ Data Set Services	Describes DASD utility functions such as dump/restore and reduction of free space fragmentation.	*Data Facility/Data Set Services: User's Guide and Reference,* SC26-3949

CONTENTS

FIGURES

SUMMARY OF AMENDMENTS

OS/VS2 MVS DASD Support

Major Technical Changes

- Two utilities have been added to those not explained in this book. A description of them is contained in the following manuals:
  - *Device Support Facilities,* GC35-0033. This utility is used for the initialization and maintenance of direct access storage devices (DASD). It supersedes IBCDASDI and IEHDASDR for these functions. In addition, it supports the IBM 3375 Direct Access Storage, the IBM 3380 Direct Access Storage, and volumes with indexed VTOC.
  - *Data Facility/Data Set Services: User's Guide and Reference,* SC26-3949, describes DASD utility functions such as dump or restore, and reduction or elimination of free space fragmentation.
- The IBM 3375 Direct Access Storage and the IBM 3380 Direct Access Storage are not supported by IBCDASDI, IBCDMPRS, or IEHDASDR. Refer to *Device Support Facilities* for information on initialization and maintenance of such DASD volumes. Refer to *Data Facility/Data Set Services: User's Guide and Reference* for information on additional support of such DASD volumes, such as dump or restore, and reduction or elimination of free space fragmentation.
- DASD volumes with indexed VTOC are not supported by IBCDASDI or IEHDASDR. Refer to *Device Support Facilities* for information on initialization and maintenance of such DASD volumes. IEHLIST supports volumes with indexed VTOC. Refer to *Data Facility/Device Support: User's Guide and Reference* for additional information.

OS/VS2 Support for the IBM 3203-5 Printer

Major Technical Changes

Throughout this manual the IBM 3203-5 Printer is correctly specified as "3203-4."

OS/VS2 MVS Data Management (VS2 Release 3.8)

Major Technical Changes

Miscellaneous maintenance changes have been made throughout.

November 1977 Edition

Major Technical Changes

- Separate manual created for *OS/VS1 Utilities*, GC26-3901.
- IEHUCAT description deleted.
- Numerous technical descriptions expanded throughout.
- Statement of non-support of the 3036 console by the Independent (standalone) utilities.

- OS/VS2 MVS System Security Support (SU32).
- OS/VS2 MVS Data Management Support (SU60).

Major Editorial Changes

- All chapters revised to include a tabular description of utility control card parameters.
- A Device Support section included in the Introduction portion of the manual.
- Specific device support information added to the IBCDASDI and IEHDASDR chapters.
- 3330, 3340, 3344, 3350 grouped as Buffered-Log DASD throughout.

OS/VS2 MVS Data Management (VS2.03.808)

CVOL

Support for OS CVOLs in MVS has been added to the IEHLIST, IEHMOVE, and IEHPROGM sections of the manual.

RACF

The user will be checked for proper access authorization to all data sets accessed by the utilities, except for the independent utility programs which operate outside of the operating system. Special cases are noted in the IEHMOVE, IEHDASDR, and IEHPROGM sections of the manual.

For information about RACF, refer to the *OS/VS2 MVS Resource Access Control Facility (RACF): General Information Manual.*

VS2 Release 3

Major Technical Changes

Mass Storage System (MSS). Added a restriction to IBCDASDI since it cannot be used to format MSS staging volumes. Also, IBCDMPRS cannot be used to dump or restore a staging volume.

Editorial Changes

- Added a restriction to IEHATLAS since it cannot be used on MSS virtual volumes or virtual devices.
- Added an example to IEHDASDR that dumps to tape and then restores a 3330 volume that contains a VSAM user catalog.
- Added an example to IEHDASDR that describes initialization of an MSS staging volume.
- Added an example to IEHMOVE that describes the MOVE DSGROUP control statement.
- Added descriptions of the MOVE and COPY DSGROUP statements to the IEHMOVE section.

VS2 Release 2

Major Technical Changes

- Incorporated changes as listed in VS1 Release 3, with the exception of the IEHUCAT chapter which does not run under VS2 Release 2.
- Added recognition of JES2 control statements to IEBEDIT.
- IEHLIST for VS2 Release 2 does not support listing catalog entries. It is suggested that this entire chapter be carefully reviewed if running under VS2 Release 2.
- IEHMOVE for VS2 Release 2 does not support moving or copying groups of cataloged data sets, catalogs, nor portions of catalogs. Neither can it move nor copy the SYSCTLG data set, ISAM data sets, or VSAM data spaces. Before running IEHMOVE under VS2 Release 2, it is suggested that this chapter be reviewed in its entirety.
- IEHPROGM for VS2 Release 2 does not support the following functions:

  1. Building and deleting indexes and their aliases.
  2. Connecting and releasing two volumes.
  3. Building and maintaining generation indexes.

  It does catalog and uncatalog non-VSAM data sets. Before running IEHPROGM under VS2 Release 2, it is suggested that this chapter be reviewed in its entirety.
- IEHDASDR for VS2 Release 2 processed both VSAM and non-VSAM data sets. However, additional restrictions are encountered when processing password-protected VSAM data sets.

Major Editorial Changes

- Added the syntax and explanation of the LABEL statement of IEBUPDTE.
- Added a section on dumping and restoring unlike devices to the IEHDASDR chapter.
- Added a data security suggestion to the IEHINITT program.
- Added an example showing labelling a tape volume at 6250 bpi to the IEHINITT chapter.
- All syntax presentations have been modified to show the required comma between keyword parameters.
- All examples have been adjusted to show exact character punch positions.
- Many notes have been added, to more readily bring to the user's attention any special or unusual requirements or restrictions of the individual programs.
- The SMF Type 21 record format in the IFHSTATR chapter has been modified to more accurately reflect the contents of this record. A paragraph on suggested use of the information gathered by SMF, and available from record type 21, has been added.
- The appendix explaining generation data groups has been removed from this publication. The information on generation data groups is now located in *OS/VS2 MVS Data Management Services Guide,* GC26-3875, and *OS/VS2 Access Method Services,* GC26-3841.

INTRODUCTION

OS/VS provides utility programs to assist in organizing and maintaining data. Each utility program falls into one of three classes of programs, determined by the function performed and the type of control of the utility.

System utility programs are used to maintain and manipulate system and user data sets. Entire volume manipulation, for example, copying or restoring, is also provided. These programs must reside in an authorized library and are controlled by JCL statements and utility control statements.

They can be executed as jobs or can be invoked as subroutines by *authorized* programs. The invocation of utility programs and the linkage conventions are discussed in "Appendix B: Invoking Utility Programs from a Problem Program."

Refer to Figure 1-1 for a list of system utility programs and unique notes when using them.

System Utility	**Purpose**
• IEHATLAS	to assign alternate tracks and recover usable data records when defective tracks are indicated.
• IEHDASDR	to initialize and label direct access volumes, to assign alternate tracks when defective tracks are indicated, or to dump or restore data.
• IEHINITT	to write standard labels on tape volumes.
• IEHLIST	to list system control data.
• IEHMOVE	to move or copy collections of data.
• IEHPROGM	to build and maintain system control data.
• IFHSTATR	to select, format, and write information about tape errors from the IFASMFDP tape or the SYS1.MAN data set.

When using system utility programs,

- Each data set to be used by programs other than IEHPROGM, IEHMOVE, and IEHLIST must be defined on a DD statement specifying the data set name. When updating activity is being performed by IEHPROGM, IEHMOVE, or IEHDASDR in a multiprogamming environment, other tasks should not be allowed to access the data set being updated. (Refer to "Appendix C: DD Statements for Defining Mountable Devices" for precautions to be taken.)
- DD statements defining mountable devices must specify that volumes mounted on those devices cannot be shared.
- Mountable volumes must not be made available to the system until the user is requested by the system to mount the specified volumes.
- A reader procedure must be used that will direct input and output data sets to volumes other than those which are to be modified by a system utility program.
- When executing a SCRATCH operation, the data set or volume being scratched must not be used by a program executing concurrently.

Figure 1-1. System Utility Programs

Data set utility programs are used to reorganize, change, or compare data at the data set and/or record level. These programs are controlled by JCL statements and utility control statements.

These utilities manipulate partitioned, sequential, or indexed sequential data sets provided as input to the programs. Data ranging from fields within a logical record to entire data sets can be manipulated.

Data set utility programs can be executed as jobs or can be invoked as subroutines by a calling program. The invocation of utility programs and the linkage conventions are discussed in "Appendix B: Invoking Utility Programs from a Problem Program."

Refer to Figure 1-2 for a list of data set utility programs.

Data Set Utility	**Purpose**
• IEBCOMPR	to compare records in sequential or partitioned data sets.
• IEBCOPY	to copy, compress, or merge partitioned data sets, to select or exclude specified members in a copy operation, and to rename and/or replace selected members of partitioned data sets.
• IEBDG	to create a test data set consisting of patterned data.
• IEBEDIT	to selectively copy job steps and their associated JOB statements.
• IEBGENER	to copy records from a sequential data set or to convert a data set from sequential organization to partitioned organization.
• IEBISAM	to place source data from an indexed sequential data set into a sequential data set in a format suitable for subsequent reconstruction.
• IEBPTPCH	to print or punch records that reside in a sequential or partitioned data set.
• IEBTCRIN	to construct records from the input data stream that have been read from the IBM 2495 Tape Cartridge Reader.
• IEBUPDTE	to incorporate changes to sequential or partitioned data sets.

Figure 1-2. Data Set Utility Programs

Independent utility programs are used to prepare devices for system use when the operating system is not available. They operate outside of, and in support of, the operating system, are controlled by utility control statements, and cannot be invoked by a calling program. They do not support, however, the 3036 display console or the 3066 console.

Refer to Figure 1-3 for a list of independent utility programs.

Independent Utility	**Purpose**
• IBCDASDI	to initialize a direct access volume and to assign alternate tracks.
• IBCDMPRS	to dump and restore the data contents of a direct access volume.
• ICAPRTBL	to load the forms control and Universal Character Set buffers of a 3211 after an unsuccessful attempt to IPL, with the 3211 printer assigned as the output portion of a composite console.

Figure 1-3. Independent Utility Programs

The selection of a specific program is dependent on the nature of the job to be performed. For example, renaming a data set involves modifying system control data. Therefore, a system utility program can be used to rename the data set. In some cases, a specific function can be performed by more than one program. Figure 1-6 at the end of this chapter is provided to help you find the program that performs the function you need.

The IEHDASDR system utility program can be used with volumes containing VSAM and/or non-VSAM data sets. The other utility programs that manipulate data sets and are contained in this manual cannot be used with VSAM data sets. Information about VSAM data sets can be found in *OS/VS2 Access Methods Services.*

Two utilities, IEHMOVE and IEBCOPY, do not support Virtual Input/Output (VIO) data sets.

Device Support

Except where noted, all of the following devices are supported by all Utility programs. Restrictions and peculiar device support will be noted in the individual Utility sections.

The table below indicates specific devices supported, and the notation to be used to reference them. The term *Buffered-Log DASD* includes all DASD except 2314/2319 and 2305 devices.

	Device-id Notation	**Devices**
DASD:	2314	2314, 2319
	2305	2305 Model 1 & 2
	3330	3330, 3333 and 3350 in 3330 compatibility mode
	3330-1	3330-MOD11, 3333-MOD11 and 3350 in 3330-MOD11 compatibility mode
	3340	3340, 3344 (both 35 & 70 megabyte models)
	3350	3350 Native mode
	3375	3375
	3380	3380
	3330V	3850 MSS Virtual Volumes
Tape:	2400	2400 (all models)
	3400	3400 (all models)
	2495	2495 (IEBTCRIN only)

Control

System and data set utility programs are controlled by job control statements and utility control statements. Independent utility programs are controlled by utility control statements; because these programs are independent of the operating system, job control statements are not required. The job control statements and utility control statements necessary to use utility programs are provided in the major discussion of each utility program.

Job Control Statements

A system or data set utility program can be introduced to the operating system in different ways:

- Job control statements can be included in the input stream.
- Job control statements, placed in a procedure library or defined as an inline procedure, can be included by means of the EXEC job control statement.
- A utility program can be invoked by a calling program.

If job control statements are placed in a procedure library, they should satisfy the requirements for most applications of the program; a procedure, of course, can be modified or supplemented for applications that require additional parameters, data sets, or devices. The data set utility IEBUPDTE can be used to enter a procedure into a procedure library; see "IEBUPDTE Program."

A job that modifies a system data set (identified by SYS1.) must be run in a single job environment; however, a job that uses a system data set, but does not modify it, can be run in a multiprogramming environment. The operator should be informed of all jobs that modify system data sets.

DD statements should ensure that the volumes on which the data sets reside cannot be shared when update activity is being performed.

Utility Control Statements

Utility control statements are used to identify a particular function to be performed by a utility program and, when required, to identify specific volumes or data sets to be processed.

The control statements for the utility programs have the following standard format:

label operation operand

The *label* symbolically identifies the control statement and, with the exception of system utility program IEHINITT, can be omitted. When included, a name must begin in the first position of the statement and must be followed by one or more blanks. It can contain from one to eight alphameric characters, the first of which must be alphabetic.

The *operation* identifies the type of control statement. It must be preceded and followed by one or more blanks.

The *operand* is made up of one or more keyword parameters separated by commas. The operand field must be preceded and followed by one or more blanks. Commas, parentheses, and blanks can be used only as delimiting characters.

Comments can be written in a utility statement, but they must be separated from the last parameter of the operand field by one or more blanks.

Continuing Utility Control Statements

Utility control statements are coded on cards or as card images and are contained in columns 1 through 71. A statement that exceeds 71 characters must be continued on one or more additional cards. A nonblank character must be placed in column 72 to indicate continuation. A utility statement can be interrupted either in column 71 or after any comma.

The continued portion of the utility control statement must begin in column 16 of the following statement. (Job control language continuations can begin in any column from 4 through 16, and do not require a nonblank character in column 72 for continued operand fields.) Comments can be placed on any card containing a complete or partial statement. However, when a card is included for the sole purpose of continuing a comment, the continuation must begin in column 16.

Note: The IEHPROGM, IEBCOPY, IEBPTPCH, IEBGENER, IEBCOMPR, and IEBDG utility programs permit certain exceptions to these requirements (see the applicable program description).

The utility control statements are discussed in detail, as applicable, in the remaining chapters.

Restrictions

- Unless otherwise indicated in the description of a specific utility program, a temporary data set can be processed by a utility program only if the user specifies the complete name generated for the data set by the system (for example, DSNAME=SYS68296.T000051.RP001.JOBTEMP.TEMPMOD).
- Standard utility programs do not normally support VSAM. Refer to the various program descriptions for certain exceptions.

Notational Conventions

A uniform system of notation describes the format of utility commands. This notation is not part of the language; it simply provides a basis for describing the structure of the commands.

The command-format illustrations in this book use the following conventions:

- Brackets [ ] indicate an optional parameter.
- Braces { } indicate a choice of entry; unless a default is indicated, you must choose one of the entries.
- Required parameters will not have brackets or braces surrounding them.
- Items separated by a vertical bar (|) represent alternative items. No more than one of the items may be selected.
- An ellipsis . . . indicates that multiple entries of the type immediately preceding the ellipsis are allowed.
- Other punctuation (parentheses, commas, spaces, etc.) must be entered as shown. A space is indicated by ƀ.
- **Boldface** type indicates the exact characters to be entered. Such items must be entered exactly as illustrated.
- *Italic* type specifies fields to be supplied by the user.
- Underscored type indicates a default option. If the parameter is omitted, the underscored value is assumed.

keyword=device=list

The term *keyword* is replaced by VOL, FROM or TO.

The term *device* is replaced by either a generic name, for example, 3330; or a substitute for a generic name, for example DISK, if this substitute has been generated into your system. For direct access devices, the term *list* is replaced by one or more volume serial numbers separated by commas. When there is more than one, the entire *list* field must be enclosed in parentheses.

For tape, the term *list* is replaced by either one or more volume serial number-comma-data set sequence number pairs. Each pair is separated from the next pair by a comma. When there is more than one pair, the entire *list* field must be enclosed in parentheses; for example: FROM=2400=(tapeA,1,tapeB,1).

Special Referencing Aids

Two special referencing aids are included in this publication to help you:

1. Locate the right utility program.
2. Locate the right example.

To locate the right utility program, refer to Figure 1-6 in "Guide to Utility Program Functions," which immediately follows this section. Figure 1-4 shows a portion of the table. The figure shows that you can use IEHINITT to label a magnetic tape volume or IEHLIST to list a volume table of contents.

Task	Definition of Task	Utility Program
Label	magnetic tape volumes	IEHINITT
List	a password entry	IEHPROGM
	a volume table of contents	IEHLIST
	partitioned directories	

Figure 1-4. Locating the Right Program

To locate the right example, use the figure—called an "example directory"—that precedes each program's examples. Figure 1-5 shows a portion of the example directory for IEHMOVE. The figure shows that IEHMOVE Example 1 is an example of moving a sequential data set and that IEHMOVE Example 2 is an example of copying a sequential data set.

Operation	Device	Comments	Example
MOVE Sequential	3330 Disk, 2314 Disks	Source volume is demounted after job completion. Two mountable disks.	1
COPY Sequential	3330 Disk, 2314 Disks	Three cataloged sequential data sets are to be copied. The 2314s are mountable.	2

Figure 1-5. Locating the Right Example

Guide to Utility Program Functions

Figure 1-6 shows a list of tasks that the utility programs can be used to perform. The left-hand column shows tasks that you might want to perform. The middle column more specifically defines the tasks. The right-hand column shows the utility programs that can be used for each task. Notice that in some cases more than one program may be available to perform the same task.

Task		**Utility Program**
Add	a password	IEHPROGM
Analyze	tracks on direct access	IEHDASDR, IBCDASDI
Assign alternate tracks	to a direct access volume	IEHDASDR, IBCDASDI
	to a direct access volume and recover usable data	IEHATLAS
Catalog	a data set	IEHPROGM
Change	data set organization	IEBUPDTE
	logical record length	IEBGENER
	volume serial number of direct access volume	IEHDASDR
Compare	a partitioned data set	IEBCOMPR
	sequential data sets	IEBCOMPR
Compress-in-place	a partitioned data set	IEBCOPY
Construct	records from MTST and MTDI input	IEBTCRIN
Convert to partitioned	a sequential data set created as a result of an unload	IEBCOPY
	sequential data sets	IEBUPDTE, IEBGENER
Convert to sequential	a partitioned data set	IEBUPDTE, IEBCOPY
	an indexed sequential data set	IEBISAM, IEBDG
Copy	a direct access volume	IEHDASDR, IBCDMPRS, IEHMOVE
	a partitioned data set	IEBCOPY, IEHMOVE
	a volume of data sets	IEHMOVE
	an indexed sequential data set	IEBISAM
	dumped data from tape to direct access	IEHDASDR, IBCDMPRS
	job steps	IEBEDIT
	members	IEBGENER, IEBUPDTE, IEBDG
	selected members	IEBCOPY, IEHMOVE
	sequential data sets	IEBGENER, IEHMOVE, IEBUPDTE
	to tape	IBCDMPRS
Create	a library of partitioned members	IEBUPDTE
	a member	IEBDG
	a sequential output data set	IEBDG
	an indexed sequential data set	IEBDG
	an output job stream	IEBEDIT
Delete	a password	IEHPROGM
	records in a partitioned data set	IEBUPDTE
Dump	a direct access volume	IEHDASDR, IBCDMPRS
Edit	MTDI input	IEBTCRIN
Edit and convert to partitioned	a sequential data set	IEBGENER, IEBUPDTE
Edit and copy	a job stream	IEBEDIT
	a sequential data set	IEBGENER, IEBUPDTE

Figure 1-6 (Part 1 of 3). Tasks and Utility Programs

Task		**Utility Program**
Edit and list	error statistics by volume (ESV) records	IFHSTATR
Edit and print	a sequential data set	IEBPTPCH
Edit and punch	a sequential data set	IEBPTPCH
Enter	a procedure into a procedure library	IEBUPDTE
Exclude	a partitioned data set member from a copy operation	IEBCOPY, IEHMOVE
Expand	a partitioned data set	IEBCOPY
	a sequential data set	IEBGENER
Format	a 3350 to 3330-1, 3330-11, or 3350 mode	IEHDASDR, IBCDASDI
Generate	test data	IEBDG
Get alternate tracks	on a direct access volume	IEHDASDR, IBCDASDI, IEHATLAS
Include	changes to members or sequential data sets	IEBUPDTE
Initialize	a direct access volume	IEHDASDR, IBCDASDI
Insert records	into a partitioned data set	IEBUPDTE
Label	magnetic tape volumes	IEHINITT
List	a password entry	IEHPROGM
	a volume table of contents	IEHLIST
	contents of direct access volume on system output device	IEHDASDR
	number of unused directory blocks and tracks	IEBCOPY
	partitioned directories	IEHLIST
Load	a previously unloaded partitioned data set	IEBCOPY
	an indexed sequential data set	IEBISAM
	an unloaded data set	IEHMOVE
	UCS and FCB buffers of a 3211	ICAPRTBL
Merge	partitioned data sets	IEHMOVE, IEBCOPY
Modify	a partitioned or sequential data set	IEBUPDTE
Move	a volume of data sets	IEHMOVE
	partitioned data sets	IEHMOVE
	sequential data sets	IEHMOVE
Number records	in a new member	IEBUPDTE
	in a partitioned data set	IEBUPDTE
Password protect	add a password	IEHPROGM
	delete a password	IEHPROGM
	list passwords	IEHPROGM
	replace a password	IEHPROGM
Print	a sequential data set	IEBGENER, IEBUPDTE, IEBPTPCH
	partitioned data sets	IEBPTPCH
	selected records	IEBPTPCH
Punch	a partitioned data set member	IEBPTPCH
	a sequential data set	IEBPTPCH
	selected records	IEBPTPCH
Read	Tape Cartridge Reader input	IEBTCRIN
Reblock	a partitioned data set	IEBCOPY
	a sequential data set	IEBGENER, IEBUPDTE

Figure 1-6 (Part 2 of 3). Tasks and Utility Programs

Task		**Utility Program**
Recover	data from defective tracks on direct access volumes	IEHATLAS
	tracks flagged as defective on some DASD	IEHDASDR, IBCDASDI
Rename	a partitioned data set member	IEBCOPY, IEHPROGM
	a sequential or partitioned data set	IEHPROGM
	moved or copied members	IEHMOVE
Renumber	logical records	IEBUPDTE
Replace	a password	IEHPROGM
	data on an alternate track	IEHATLAS
	identically named members	IEBCOPY
	logical records	IEBUPDTE
	members	IEBUPDTE
	records in a member	IEBUPDTE
	records in a partitioned data set	IEBUPDTE, IEBCOPY
	selected members	IEBCOPY
	selected members in a move or copy operation	IEBCOPY, IEHMOVE
Restore	a dumped direct access volume from tape	IBCDMPRS, IEHDASDR
Scratch	a volume table of contents	IEHPROGM
	data sets	IEHPROGM
Uncatalog	data sets	IEHPROGM
Unload	a partitioned data set	IEHMOVE, IEBCOPY
	a sequential data set	IEHMOVE
	an indexed sequential data set	IEBISAM
Update	in place a partitioned data set	IEBUPDTE
Write	IPL records and a program on a direct access volume	IBCDASDI, IEHDASDR

Figure 1-6 (Part 3 of 3). Tasks and Utility Programs

IBCDASDI PROGRAM

IBCDASDI is an independent utility used to initialize direct access volumes for use and to assign alternate tracks on direct access storage volumes. IBCDASDI jobs can be performed continuously by stacking complete sets of control statements.

Initializing a Direct Access Volume

IBCDASDI can be used to initialize a direct access volume by two methods;

A non-QUICK DASDI will:

1. Unassign all alternate tracks
2. Rewrite the home address and/or record zero (HA/R0) on all tracks
3. Test flagged defective tracks and recover them if no errors are detected
4. Assign defective tracks to new, alternate tracks
5. Perform all other functions of QUICK DASDI

A QUICK DASDI will:

1. Write IPL records on track 0 (records 1 and 2)
2. Write volume labels on track 0 (record 3) and provide space for additional records, if requested (reads alternate tracks and decreases the total count of the alternates by one when an alternate is found defective or assigned)
3. Construct and write a volume table of contents (VTOC)
4. Write an IPL program, if requested, on track 0
5. Optionally, check for tracks that have been previously designated as defective (flagged) and have had alternate tracks assigned
6. Optionally, write a track descriptor record (record 0) and erase the remainder of each track. May also attempt to reclaim any track that has the defective bit on in the flag byte of the home address.

Assigning an Alternate Track

IBCDASDI can be used to: (1) test a track* and, if necessary, assign an alternate or (2) bypass testing and automatically assign an alternate.

If testing is performed, an alternate track is assigned for any track found defective. If the defective track is an unassigned alternate, it is flagged to prevent its future use. The alternate track address is made known to the operator.

If a track is tested and not found to be defective, no alternate is assigned. The operator is notified by a message.

If testing is bypassed, an alternate track can be assigned for the specified track or its alternate, whether it is defective or not. If the specified track is an unassigned alternate, it is flagged to prevent its future use.

***Only 2314 and 3350 (native) devices are tested before alternate tracks are assigned.**

Executing IBCDASDI

IBCDASDI is loaded as card decks or as card images on tape. Control statements for the requested program can follow the last card or card image of the program, or can be entered on a separate input device. To execute IBCDASDI:

1. Place the object program deck in the reader or mount the tape reel that contains the object program.
2. Load the object program from the reader or tape drive by setting the load selector switches and pressing the console LOAD key. When the program is loaded, the wait state is entered and the console lights display the hexadecimal value FFFF.
3. Define the control statement input device in one of the following ways:
   a. Press the REQUEST key of the console typewriter* and, in response to the message "DEFINE INPUT DEVICE", enter "INPUT=*xxxx,cuu*". The *xxxx* is the device type, *c* is the channel address, and *uu* is the unit address. The device type can be 1402, 2400, 2501, 2540, or 3505.
   b. If the console typewriter is not available, enter at storage location 0110 (hexadecimal): 2*cuu* for a 2400 9-track tape unit; or 0*cuu* for a 2540 Card Read Punch, 2501 card reader, 3410 tape, or 3420 tape. Press the console INTERRUPT key.
4. Control statements are printed on the message output device. At the end of the job, "END OF JOB" is printed on the message output device, and the program enters the wait state.

Input and Output

IBCDASDI uses as input a control data set which consists of utility control statements.

IBCDASDI produces as output an initialized direct access volume and a message data set.

Control

Use IEHDASDR for online initialization of all supported DASD.

IBCDASDI is controlled by utility control statements. Because IBCDASDI is an independent utility, operating system job control statements are not used.

---

* Terminal screen consoles must be in "printer keyboard" mode.

Utility Control Statements

All utility control statements/operands must be preceded and followed by one or more blanks.

IBCDASDI utility control statements in the order in which they must appear are:

Statement	**Use**
JOB	Indicates the beginning of an IBCDASDI job.
MSG	Defines an output device for operator messages.
DADEF	Defines the volume to be initialized.
VLD	Contains information for constructing an initial volume label and for allocating space for additional labels.
VTOCD	Contains information for controlling the location of the volume table of contents.
IPLTXT	Separates utility control statements from any IPL program text statements.
GETALT	Assigns an alternate track on a volume.
END	Indicates the end of an IBCDASDI job.
LASTCARD␢	Ends a series of stacked IBCDASDI jobs.

Figure 2-1. IBCDASDI Utility Control Statements

JOB Statement

The JOB statement indicates the beginning of an IBCDASDI job.

The format of the JOB statement is:

[*label*] **JOB** [*user-information*]

MSG Statement

The MSG statement defines an output device for operator messages. It follows the JOB statement and precedes any function definition statements.

The format of the MSG statement is:

[*label*] **MSG** **TODEV=***xxxx*
,TOADDR=*cuu*

DADEF Statement

The DADEF statement defines the direct access volume to be initialized.

The format of the DADEF statement is:

[*label*] **DADEF** **TODEV=***xxxx*
,TOADDR=*cuu*
[,IPL={YES | NO}]
,VOLID={ *serial* **| SCRATCH}**
[,FLAGTEST={NO | YES}]
[,PASSES=*n* **]**
[,BYPASS={YES | NO}]
[,MODEL=*n* **]**

VLD Statement

The VLD Statement contains information for constructing an initial volume label and for allocating space for additional labels.

The format of the VLD statement is:

[*label*] **VLD** **NEWVOLID**=*serial*
[,VOLPASS={0 | 1}]
[,OWNERID=*xxxxxxxxxx* **]**
[,ADDLABEL=*n* **]**

VTOCD Statement

The VTOCD statement contains information for controlling the location of the volume table of contents (VTOC).

The format of the VTOCD statement is:

[*label*] **VTOCD** **STRTADR**=*nnnnn*
,EXTENT=*nnnn*

IPLTXT Statement

The IPLTXT statement separates utility control statements from IPL program text statements. It is required only when IPL text is included.

The format of the IPLTXT statement is:

IPLTXT

IPLTXT must be preceded by at least one blank space.

When IPL text is included, **END** must start in column 2. See "END Statement" below.

GETALT Statement

The GETALT statement is used to assign an alternate track on a volume. Any number of alternate tracks can be assigned in a single job by including a GETALT statement for each track.

Note: A GETALT statement that applies to a 3330, 3330-1, or 3340/3344 device causes an alternate track to be assigned automatically without testing.

The format of the GETALT statement is:

[*label*] **GETALT** **TODEV**=*xxxx*
,TOADDR=*cuu*
,TRACK=*cccchhhh*
,VOLID=*serial*
[,FLAGTEST={NO | YES}]
[,PASSES=*n* **]**
[,BYPASS={YES | NO}]
[,MODEL=*n* **]**

The GETALT function should not be used immediately after a RESTORE operation that did not complete successfully. Before using GETALT in such a case, reinitialize the volume, if possible.

END Statement

The END statement denotes the end of job. It appears after the last function definition statement.

The format of the END statement is:

[*label*]**END** [*user-information*]

END must be preceded and followed by at least one blank.

END must start in column 2 if IPLTXT is included.

LASTCARD Statement

The LASTCARD statement is required only when an IBCDASDI job or a series of stacked IBCDASDI jobs is followed by other statements on the control statement input device. The LASTCARD statement must follow the last END statement applying to an IBCDASDI job.

The format of the LASTCARD statement is:

LASTCARD

LASTCARD must be preceded by at least one blank space.

Operands	Applicable Control Statements	Description of Operands/Parameters
ADDLABEL	VLD	**ADDLABEL=*n*** specifies the total number of additional labels for which space is to be allocated. The value of *n* can be 1 through 7. **Default:** 0
BYPASS	DADEF	**BYPASS=YES** specifies that no check is to be made for defective tracks. If 2314: write standard R0 on each track. No check will be made for defective tracks. If Buffered-log DASD: the BYPASS parameter is not applicable. **Default:** NO IF 2314: If FLAGTEST=NO write HA and 7294 byte R0, then test (read R0). Write HA and standard R0 on each track. If FLAGTEST=YES, write 7294 byte R0 then test (read R0). Write standard R0 on each track.
	GETALT	**BYPASS=YES** Applicable only to 2314 and 3350 (native and compatibility modes); causes an alternate track to be assigned without testing the track to be flagged. **Default:** BYPASS=NO Test the track to be flagged and assign an alternate only if the test results are in error (data check).
EXTENT	VTOCD	**EXTENT=*nnnn*** specifies the length (number of tracks) of the VTOC.

Device	VTOC Entries per Track
2314	25
2319	25
2305-1	18
2305-2	34
3330	39
3330-1	39
3340/3344	22
3350	47

Figure 2-2. VTOC Entries per Track

Operands	Applicable Control Statements	Description of Operands/Parameters
FLAGTEST	DADEF	**FLAGTEST={NO | YES}** If Buffered-log DASD: the FLAGTEST parameter is not applicable. If 2314: FLAGTEST=NO specifies that all tracks will be tested whether flagged defective or not. Write HA on each track if BYPASS=NO is also specified. **Default:** YES If 2314: check for and maintain all flagged (defective) tracks by assigning alternates.
	GETALT	If Buffered-log DASD: the FLAGTEST parameter is not applicable. If 2314: FLAGTEST=NO specifies previously flagged tracks will be tested before assigning alternates (see BYPASS). **Default:** YES If 2314: previously flagged tracks will remain flagged.
IPL	DADEF	**IPL={YES | NO}** specifies that an IPL program is to be written on the volume. An IPL initialization program must be written on a device to be used for system residence. **Default:** No IPL program is written.
MODEL	DADEF GETALT	**MODEL=***n* specifies a decimal model number (1 or 2). This parameter corresponds to the 2305-1 and 2305-2, respectively. **MODEL** is required when a 2305 is to be initialized.
NEWVOLID	VLD	**NEWVOLID=***serial* specifies a one- to six-character volume serial number.
OWNERID	VLD	**OWNERID=***xxxxxxxxxx* specifies a one- to ten-character field that identifies the owner of the volume. **Default:** no identification given.

Operands	Applicable Control Statements	Description of Operands/Parameters
PASSES	DADEF	**PASSES=*n*** For 2314: in checking for defective tracks. (n=1:255) For 3330: If PASSES=0, do a QUICK DASDI. If PASSES=1, write R0 on each track. If PASSES>1, write R0 on each track 'n' times. No surface analysis is performed. For 3340: If PASSES=0, do a QUICK DASDI. If PASSES>=1, test all flagged (defective) tracks and recover (unflag) those that test okay. Write R0 on each track. For 3350: If PASSES=0, do a QUICK DASDI. If PASSES>=1, write HA/R0 on each track. Test all flagged tracks and recover (unflag) those with no errors.
	GETALT	For 2314: specifies the number of passes per track to be made in checking for defective tracks. (n=1:255) For Buffered-log DASD: the PASSES parameter is not applicable.
STRTADR	VTOCD	**STRTADR=*nnnnn*** specifies the one- to five-byte *decimal* track address, relative to the beginning of the volume, at which the VTOC is to begin. The VTOC cannot occupy track 0 or any alternate track. To improve performance when reading from and writing to the VTOC, it is recommended that every VTOC end on the last track of a cylinder (a cylinder boundary). This means that you should determine the starting address for the VTOC by subtracting the number of tracks allocated to the VTOC from the nearest larger track that ends on a cylinder boundary. For example, if the VTOC requires 5 tracks on a 3336 disk pack, which has 19 tracks per cylinder, the starting track should be specified as track 14, so that the VTOC will end on track 18 (the last track of the first cylinder).
TOADDR	MSG DADEF GETALT	**TOADDR=*cuu*** specifies the channel number, *c,* and unit number, *uu,* of the message output device (MSG), or the direct access device (for DADEF and GETALT).
TODEV	MSG	**TODEV=*xxxx*** specifies the type of device to receive messages. All supported tape drives (see Introduction - Device Support) and the following unit-record devices: 1403, 1443, 1052, 3203-4, 3210, 3215, 3211 and 3800.

Operands	Applicable Control Statements	Description of Operands/Parameters
	DADEF GETALT	specifies the type of DASD device (see Introduction - Device Support for proper device notation).
TRACK	GETALT	**TRACK=***ccccchhhh* specifies the hexadecimal address of the track for which an alternate is requested, where *cccc* is the cylinder number and *hhhh* is the head number.
user-information	JOB END	[*user-information*] specifies user explanation of action.
VOLID	DADEF GETALT	**VOLID=**{*serial* | **SCRATCH**} specifies the volume serial number of the volume to which an alternate track is to be assigned. If *serial* does not match the volume serial number found on this volume, the operator is notified and the job is terminated. SCRATCH specifies that no volume serial number check is to be made.
VOLPASS	VLD	**VOLPASS=**{**<u>0</u>** | **1**} specifies the value of the volume security bit. 0 specifies that the volume is not security protected. 1 specifies that the volume is security protected.

Restrictions:

- IBCDASDI should not be used to format Mass Storage System staging volumes because the disk format written by this utility is incompatible with the disk format required for staging volumes. IBCDASDI may be used to initialize a pack that has been formatted for use as a staging pack. You must use the DADEF option, PASSES=1, to re-initialize a staging pack for normal system use.
- IBCDASDI does not support volumes with indexed VTOC, the IBM 3375, or the IBM 3380. Refer to *Device Support Facilities* for information on initialization and maintenance of such DASD volumes.

IBCDASDI Examples

The examples that follow illustrate some of the uses of IBCDASDI. See the IBCDASDI utility control statement descriptions for complete device dependent information. Figure 2-3 can be used as a quick reference guide to IBCDASDI examples. The numbers in the "Example" column point to examples that follow:

Note: Examples which use *disk* in place of actual device-ids, must be changed before use. See the Device Support section, in the Introduction to this manual, for valid device-id notation.

Operation	Comments	Example
Initialize	A disk volume is to be initialized with surface analysis. (2305 and 2314 only)	1
Initialize	A disk volume is to be initialized without surface analysis. (2305 and 2314 only)	2
Initialize	A disk volume to be used as the system residence volume is to be initialized. An IPL program is included in TXT format.	3
Initialize	A 3350 volume is to be formatted for compatible 3330-11 mode and initialized	4
Initialize	A 3340 volume is to be initialized. Flagged (defective) tracks are to be tested and recovered if no (data check) errors occur.	5
Assign alternate tracks	Three alternate tracks are to be assigned on a disk volume.	6

Figure 2-3.IBCDASDI Example Directory

IBCDASDI Example 1

In this example, a 2305 volume is initialized with surface analysis.

```
                                                                72
INIT    JOB   'INITIALIZE 2305'
        MSG  TODEV=1403,TOADDR=00E
      DADEF  TODEV=2305,TOADDR=140,VOLID=SCRATCH,FLAGTEST=NO, C
        MODEL=2
        VLD  NEWVOLID=111111
      VTOCD  STRTADR=40,EXTENT=8
        END
```

The control statements are discussed below:

- JOB initiates the IBCDASDI job.
- MSG defines the 1403 on channel 0, unit 0E, as the output message device.
- DADEF specifies that a 2305 volume on channel 1, unit 40, is to be initialized. No check is to be made for previously flagged tracks.

- VLD specifies 111111 as the volume serial number of the volume to be initialized.
- VTOCD specifies the starting address and length in tracks of the volume table of contents.

IBCDASDI Example 2

In this example, a disk volume is initialized. No surface analysis is performed with the initialization.

```
INIT  JOB  INITIALIZE DISK
       MSG TODEV=1403,TOADDR=00E
     DADEF TODEV=disk,TOADDR=140,VOLID=SCRATCH,BYPASS=YES
       VLD NEWVOLID=230500
     VTOCD STRTADR=1,EXTENT=7
       END
```

The control statements are discussed below:

- DADEF specifies that a disk volume is to be initialized and specifies the channel and unit number. No check is to be made for the volume serial number or for defective tracks.
- VLD specifies the volume serial number of the volume to be initialized.
- VTOCD specifies that the volume table of contents is to begin on track 1 and is to extend over seven tracks. The VTOC terminates on the last track of the first cylinder.
- END specifies the end of the IBCDASDI job.

IBCDASDI Example 3

In this example, a disk volume is initialized for later use as a system residence volume. An IPL program is included in standard TXT format.

```
INIT      JOB  'INITIALIZE DISK'
           MSG TODEV=1403,TOADDR=00E
         DADEF TODEV=disk,TOADDR=150,IPL=YES,VOLID=SCRATCH
           VLD NEWVOLID=P10000,OWNERID=BROWN,ADDLABEL=2
         VTOCD STRTADR=2,EXTENT=7
        IPLTXT
```

(IPL program text statements)

```
           END
```

The control statements are discussed below:

- DADEF specifies that a disk volume is to be initialized and specifies the channel number and unit number. An IPL program is to be included.
- VLD specifies a volume serial number and owner identification for the volume to be initialized. It also specifies that space is to be allocated for two additional labels.
- VTOCD specifies that the volume table of contents is to begin on track 2 and is to extend over nine tracks.
- IPLTXT specifies the beginning of IPL program text statements.
- END specifies the end of IPL program text statements. Because IPL text is included, END begins in column 2.

IBCDASDI Example 4

In this example, a 3350 volume (in 3350 or 3330 format) will be reformatted to compatible 3330-1 format. HA and R0 fields will be rewritten. Each flagged (defective) track encountered will be recovered.

```
                                                              72
INIT   JOB   'INITIALIZE 3350 TO 3330-1 FORMAT'
       MSG   TODEV=1403,TOADDR=00E
     DADEF   TODEV=3330-1,TOADDR=360,VOLID=SCRATCH,            C
             PASSES=1
       VLD   NEWVOLID=333011
     VTOCD   STRTADR=7675,EXTENT=19
       END
```

The control statements are discussed below:

- DADEF specifies that a 3350 in 3330-1 compatibility mode is to be reformatted to 3330-1 format and initialized. Flagged (defective) tracks will be tested and recovered (unflagged) if no errors occur.
- VLD specifies 333011 as the volume serial number.
- VTOCD specifies a one cylinder VTOC in the center of the 3330-1 volume.

IBCDASDI Example 5

In this example, a 3344 volume will be initialized. Flagged (defective) tracks will be tested and recovered (unflagged) if no errors occur. R0 will be rewritten on each track.

```
                                                              72
INIT   JOB   'INITIALIZE 3344'
       MSG   TODEV=1403,TOADDR=00E
     DADEF   TODEV=3340,TOADDR=259,VOLID=SCRATCH,              C
             PASSES=1,BYPASS=NO
       VLD   NEWVOLID=3340AA
     VTOCD   STRTADR=2,EXTENT=10
       END
```

The control statements are discussed below:

- DADEF specifies a 3340 volume is to be initialized.
- VLD specifies 3340AA as the volume serial number.
- VTOCD specifies starting address and length of the volume table of contents.

IBCDASDI Example 6

In this example, three alternate tracks are assigned to a disk volume, without reinitialization of the volume. The check for a defective track is bypassed when the first two of the three tracks are assigned.

```
                                                              72
ALTRK       JOB  ASSIGN ALTERNATE TRACKS ON DISK
            MSG  TODEV=1052,TOADDR=009
STMT1    GETALT  TODEV=disk,TOADDR=150,VOLID=P20000,           C
                 BYPASS=YES,TRACK=006F0001
STMT2    GETALT  TODEV=disk,TOADDR=150,VOLID=P20000,           C
                 BYPASS=YES,TRACK=00910004
STMT3    GETALT  TODEV=disk,TOAADR=150,                        C
                 TRACK=004B0007,VOLID=P20000
            END
```

The control statements are discussed below:

- The first and second GETALT statements bypass the check for defective tracks.
- The third GETALT statement causes the check for a defective track to be made because BYPASS is not included.

IBCDMPRS PROGRAM

IBCDMPRS is an independent utility used to dump and restore data on direct access volumes.

The data contents of a direct access volume (all data except the home address) can be dumped to supported DASD or tape volumes and restored to a direct access volume that resides on the same type of device as the source volume. Both the source volume and the volume to which data is to be restored must have been initialized according to operating system specifications. IBCDMPRS is useful for preparing transportable copies and backup copies of direct access volumes.

IBCDMPRS cannot be used to dump or restore a staging volume. For further information see *OS/VS Mass Storage System (MSS) Services: General Information.*

Executing IBCDMPRS

IBCDMPRS is loaded as a card deck or as card images on tape. Control statements for the requested program can follow the last card or card image of the program, or can be entered on a separate input device. To execute IBCDMPRS:

1. Place the object program deck in the reader or mount the tape reel that contains the object program.
2. Load the object program from the reader or tape drive by setting the load selector switches and pressing the console LOAD key. When the program is loaded, the wait state is entered and the address portion of the current PSW is set to X'FFFF'.
3. Define the control statement input device in one of the following ways:
   a. Press the REQUEST key of the console typewriter and, in response to the message "DEFINE INPUT DEVICE", enter "INPUT=*xxxx,cuu*". The *xxxx* is the device type, *c* is the channel address, and *uu* is the unit address. The device type can be 1402, 2400, 2501, 2540, or 3505.
   b. If the console typewriter is not available, enter at storage location 0110 (hexadecimal): 2*cuu* for a 2400 9-track tape unit; or 0*cuu* for a 2540 Card Read Punch, 2501 card reader, 3410 tape, or 3420 tape. Press the console INTERRUPT key.
4. Control statements are printed on the message output device. At the end of the job, "END OF JOB" is printed on the message output device, and the program enters the wait state with the address portion of the current PSW set to X'EEEE'.

Input and Output

IBCDMPRS uses as input:

- A control data set, which contains utility control statements.
- A data set to be dumped to tape or to be restored to a direct access volume.

IBCDMPRS produces as output:

- A data set dumped to tape or a data set restored to a direct access volume.
- A message data set.

Control

IBCDMPRS is controlled by utility control statements. Because IBCDMPRS is an independent utility, operating system job control statements are not used.

Utility Control Statements

All utility control statement operands must be preceded and followed by one or more blanks.

IBCDMPRS utility control statements are:

Statement	Use
JOB	begin an IBCDMPRS job.
MSG	Defines an output device for operator messages.
DUMP	Identifies the volume to be dumped and the receiving volume.
VDRL	Specifies the upper and lower track limits of a partial dump.
RESTORE	Identifies the source volume whose data is to be restored and the receiving volume.
END	Indicates the end of an IBCDMPRS job.

Figure 3-1. IBCDMPRS Utility Control Statements

JOB Statement

The JOB statement indicates the beginning of a job.

The format of the JOB statement is:

[*label*] **JOB** [*user-information*]

MSG Statement

The MSG statement defines an output device for operator messages. It follows the JOB statement and precedes any function definition statements.

The format of the MSG statement is:

[*label*] **MSG** **TODEV**=*xxxx*
,TOADDR=*cuu*

DUMP Statement

The DUMP statement is used to identify both the source volume whose contents are to be dumped and the receiving volume. The data contents of the entire source volume are dumped, including any data on alternate tracks. If both the source and receiving volumes reside on the same type of direct access device, the receiving volume is an exact replica of the source volume.

Dump time can be minimized by selecting devices assigned to different channels. For example:

```
DUMP FROMDEV=3330,FROMADDR=150,TODEV=2400,TOADDR=282
```

The format of the DUMP statement is:

```
[label] DUMP  FROMDEV=xxxx
              ,FROMADDR=cuu
              ,TODEV=xxxx
              ,TOADDR=cuu
              [,VOLID=serial [, serial ]]
              [,MODE=mm ]
              [,MODEL=n ]
```

VDRL Statement

The VDRL (volume dump/restore limits) statement is used to specify the upper and lower limits of a partial dump. If a track within these limits has had an alternate assigned to it, the data on the alternate track is included in the dump. When the VDRL statement is used, it must be preceded by a DUMP statement and must be followed by an END statement.

The format of the VDRL statement is:

```
[label] VDRL  BEGIN={nnnnn | 0}
              [,END=nnnnn ]
```

RESTORE Statement

The RESTORE statement is used to identify both the source volume whose data contents are to be restored and the receiving volume.

Note: IBCDMPRS can be used to restore a tape created by IEHDASDR. Conversely, IEHDASDR can be used to restore a tape created by IBCDMPRS.

Restore time can be minimized by selecting devices assigned to different channels. For example:

```
RESTORE FROMDEV=2400,FROMADDR=282,TODEV=3330,TOADDR=150
```

The format of the RESTORE statement is:

```
[label] RESTORE  FROMDEV=xxxx
                 ,FROMADDR=cuu
                 ,TODEV=xxxx
                 ,TOADDR=cuu
                 ,VOLID=serial
                 [,MODE=mm ]
                 [,MODEL=n ]
```

END Statement

The END statement marks the end of job. It appears after the last function definition statement.

The format of the END statement is:

```
[label] END [user-information ]
```

Operands	Applicable Control Statements	Description of Operands/Parameters
BEGIN	VDRL	**BEGIN=**{*nnnnn* | 0} specifies a one- to five-byte relative decimal track address that identifies the first track to be dumped. **Default:** ?
END	VDRL	**END=***nnnnn* specifies the relative decimal track address of the last track to be dumped. If only one track is to be dumped, this address is the same as the beginning address. **Default:** the last track of the volume, excluding those tracks reserved as alternates, is assumed to be the upper limit.
FROMADDR	DUMP RESTORE	**FROMADDR=***cuu* specifies channel number, *c,* and unit number, *uu,* of the source device.
FROMDEV	DUMP RESTORE	**FROMDEV=***xxxx* specifies the type of the source device.
MODE	DUMP RESTORE	**MODE=***mm* specifies the bit density for data written to the receiving tape volume. This parameter must match the mode specified when data was written to the source volume. MODE should not be specified if the source or receiving volumes are not tape or if MODE was not specified when data was written to the source volume. This parameter is applicable to tape units with density selections of 800, 1600, and 6250 bits per inch. Valid modes for 7-track page are shown in Figure 3-2. (Only those modes that set the data converter on are accepted.) For 9-track tape with density selections of 800, 1600, and 6250 bits per inch, the mode settings are CB, C3, and D3, respectively. If the receiving device is not a tape unit, the MODE parameter is ignored. If the receiving device is a tape device but no mode is specified, the data is written at the highest density supported by the device.

Mode (mm)	Density (bits per inch)	Translator	Data Converter	Parity
13	200	Off	On	Odd
53	556	Off	On	Odd
93	800	Off	On	Odd

Figure 3-2. Valid 7-Track Tape Unit Modes in IBCDMPRS

Operands	Applicable Control Statements	Description of Operands/Parameters
MODEL	DUMP RESTORE	**MODEL=***n* specifies a decimal model number (1 or 2) for a 2305. This parameter is applicable only when a 2305 is specified. **Default:** 2305-1 is assumed.

Operands	**Applicable Control Statements**	**Description of Operands/Parameters**
TOADDR	MSG DUMP RESTORE	**TOADDR=***cuu* specifies the channel number, *c,* and unit number, *uu,* of the message output device (MSG) or the receiving device (DUMP and RESTORE).
TODEV	DUMP RESTORE	**TODEV=***xxxx* specifies the type of the receiving device. For RESTORE, this device type must be compatible with the device originally contained in the volume. If the receiving device is a tape unit and no MODE parameter is specified, the data is written at the highest density supported by the device. (For 7-track tape, the default mode is 93.)
	MSG	**TODEV=***xxx* specifies the type of device to receive messages. All supported tape drives (see Introduction-Device Support) and the following unit-record devices: 1403, 1443, 1052, 3203-4, 3210, 3215, 3211, and 3800.
user-information	JOB END	[*user-information*] specifies user explanation of action, and comments
VOLID	DUMP RESTORE	**VOLID=***serial* [,*serial*]... specifies the volume serial numbers of the receiving volumes. VOLID is required when the receiving volume has a standard label. If *serial* does not match the volume serial number found on the receiving volume, the operator is notified and the job is terminated. If VOLID is not specified and the receiving volume contains a volume serial number, the operator is notified.

Restrictions: IBCDMPRS does not support volumes with indexed VTOC, the IBM 3375, or the IBM 3380. Refer to *Data Facility/Data Set Services: User's Guide and Reference* for information on this support.

IBCDMPRS Examples

The examples that follow illustrate some of the uses of IBCDMPRS. Figure 3-3 can be used as a quick reference guide to the examples. The numbers in the "Example" column point to examples that follow.

Operation	Comments	Devices	Example
DUMP	A direct access volume is to be dumped to a tape volume.	disk, tape	1
RESTORE	A data set dumped to tape is to be restored to a direct access volume.	disk, tape	2

Figure 3-3. IBCDMPRS Example Directory

Note: Examples which use *disk* or *tape,* in place of actual device-ids, must be changed before use. See the Device Support section, in the Introduction to this manual, for valid device-id notation.

IBCDMPRS Example 1

In this example, a direct access volume is dumped to a tape volume:

```
                                                                       72
DUMP        JOB  DUMP DISK ONTO TAPE
            MSG  TODEV=3210,TOADDR=009
           DUMP  FROMDEV=diskFROMADDR=150,                              C
                 TODEV=tape,TOADDR=280
 END
```

IBCDMPRS Example 2

In this example, dumped data is restored to a direct access volume:

```
                                                                       72
RESTORE     JOB  RESTORE DISK FROM TAPE
            MSG  TODEV=3210,TOADDR=009
        RESTORE  FROMDEV=tape,FROMADDR=280,TODEV=disk,                  C
                 TOADDR=150,VOLID=PZ1111
 END
```

ICAPRTBL PROGRAM

ICAPRTBL is an independent utility used to load the Universal Character Set (UCS) buffer and the forms control buffer (FCB) for an IBM 3211 or 3203-4 Printer.

ICAPRTBL is used when the 3211/3203-4 is assigned as the output portion of a composite console and an unsuccessful attempt has been made to initialize the operating system because the UCS and FCB buffers contain improper bit patterns. ICAPRTBL is used to properly load the buffers so the operating system can be initialized.

Note: When an operable console printer keyboard is available, the buffers are loaded under the control of the operating system.

Executing ICAPRTBL

ICAPRTBL must be loaded from a card reader. Control statements must follow the last card of the program. Only one printer can be initialized each time the program is executed.

To execute ICAPRTBL:

1. Mount the correct train on the printer and ready the printer.
2. Place the object program deck and the control cards in the card reader. Ready the reader and press the END OF FILE key.
3. Load the object program from the reader by setting the load selector switches and pressing the console LOAD key.

Wait state codes will be displayed in the address portion of the PSW for normal termination and for input/output, system or control card errors. Code B01 is issued for normal termination; B02 through B07 are issued for control card errors; B0A through B0C are issued for system errors; and B11 through B1D are issued for input/output errors. Figure 4-1 shows these codes and their meanings.

Code	Meaning
B01	Visually check the train image printed on the 3211/3203-4.
B02	Missing control card or control card out of order.
B03	Incorrect JOB statement.
B04	Incorrect DFN statement.
B05	Incorrect UCS statement.
B06	Incorrect FCB statement.
B07	Incorrect END statement.
B0A	External interrupt.
B0B	Program check interrupt.
B0C	Machine check interrupt.
B11	Reader not online.
B12	Reader not ready.
B13	Reader unit check (display low main storage location 2 through 7 for sense information).
B14	Reader channel error.
B15	No device end on reader.
B19	Printer not online.
B1A	Printer not ready.
B1B	Printer unit check (display low virtual storage location 2 through 7 for sense information).
B1C	Printer channel error.
B1D	No device end on printer.

Figure 4-1. ICAPRTBL Wait-State Codes

Input and Output

ICAPRTBL uses as input utility control statements that contain images to be loaded into the Universal Character Set and/or Forms Control Buffer. ICAPRTBL produces as output properly loaded UCS and FCB buffers.

Control

ICAPRTBL is controlled by utility control statements. Because ICAPRTBL is an independent utility, operating system job control statements are not used.

Utility Control Statements

All utility control statement operands must be preceded and followed by one or more blanks.

ICAPRTBL utility control statements are:

Statement	Use
JOB	Indicates the beginning of an ICAPRTBL job.
DFN	Defines the address of the 3211 or 3203-4.
UCS	Contains an image of the characters to be loaded into the UCS buffer.
FCB	Defines the image to be loaded into the FCB.
END	Indicates the end of an ICAPRTBL job.

Figure 4-2. ICAPRTBL Utility Control Statements

JOB Statement

The JOB statement indicates the beginning of an ICAPRTBL job.

The format of the JOB statement is:

[*label*] **JOB** [*user-information*]

DFN Statement

The DFN statement is used to define the address of the 3211 or 3203-4, to specify that lowercase letters are to be printed in uppercase when the lowercase print train is not available, and to identify UCS and FCB image-ids.

The format of the DFN statement is:

DFN ADDR=*cuu,* **[FOLD={Y | N}] [,DEVT={3211 | 3203-4}] [,UCS=***ucsname* **]**
[,FCB=*fcbname* **]**

UCS Statement

The UCS statement contains an image to be loaded into the UCS buffer.

The format of the UCS statement is:

[*ucsname*] **UCS** *ucs-image*

FCB Statement

The FCB statement defines the image to be loaded into the forms control buffer. The FCB statement may precede or follow the UCS statement.

The format of the FCB statement is:

[*fcbname*] **FCB** **LPI={6 | 8}**
,LNCH=((*l, c***)[,(***l, c***,)...])**
,FORMEND=*x*

END Statement

The END statement signals the end of the ICABPRTBL job.

The format of the END statement is:

[*label*] **END** [*user-information*]

Operands	Applicable Control Statements	Description of Operands/Parameters
ADDR	DFN	**ADDR=***cuu* specifies the channel number, *c,* and unit number, *uu,* of the 3211.
DEVT	DFN	**DEVT={3211 | 3203-4}** specifies the device type for which the ADDR parameter addresses.
FCB	DFN	**FCB=**{*fcbname* **| STD | STD2**} specifies a one- to eight-character name of the image loaded into the forms control buffer. The actual image loaded into the buffer is not affected by this name, but serves as a meaningful reference when printed on the printer, *fcbname* should be the same as the FCB image being used.
FOLD	DFN	**FOLD={Y | N}** specifies whether lowercase letters are to be printed as uppercase letters when the lowercase print train is not available. These values can be coded: **Y** specifies that lowercase letters are to be printed as uppercase letters when the lowercase print train is not available. **N** specifies that lowercase letters are not to be printed as uppercase letters.
FORMEND	FCB	**FORMEND=***x* specifies the number of lines (maximum 180) on the printer form. For an 11 inch form, spacing six lines per inch, *x* must be 66.
LNCH	FCB	**LNCH=((***l,c* **)[,(***l,c* **)...])** specifies the channels of the FCB image. Each set of parentheses must contain the line number (1-180), a comma, and the channel number (1-12) to be assigned to that line. One or all of the 12 channels may be assigned in any order. Each set must be separated by commas and the entire group surrounded by parentheses.
LPI	FCB	**LPI={6 | 8}** specifies the number of lines per inch that will be printed on the document. These values can be coded: **6** specifies that six lines per inch will be printed. **8** specifies that eight lines per inch will be printed.

Operands	Applicable Control Statements	Description of Operands/Parameters
UCS	DFN	UCS={*ucsname* | AN | A11} is a one- to eight-character alphameric name of the image loaded into the UCS buffer. This name is printed on the printer to serve as a reference to the print train being used. **AN** is the default for 3203-4 devices. **A11** is the default for 3211 devices.
ucs-image	UCS	*ucs-image* specifies characters to be loaded into the UCS buffer. The characters must be contained in columns 16 through 71. The first UCS statement contains the first 56 characters; subsequent statements contain continuations of the image to be loaded into the UCS buffer.
user-information	JOB END	[*user-information*] specifies user explanation of action and comments.

ICAPRTBL Examples

The examples that follow illustrate some of the uses of ICAPRTBL. Figure 4-3 can be used as a quick reference guide to the examples. The numbers in the "Example" column point to examples that follow.

Devices	**Example**
3211	1
3211	2
3203-4	3
3203-4	4

Figure 4-3. ICAPRTBL Example Directory

ICAPRTBL Example 1

In this example, a 3211 UCS image (A11) and an FCB image are loaded into the UCS and FCB buffers.

```
                                                                         72
      JOB LOAD A11 IMAGE
      DFN ADDR=002,FOLD=N
A11   UCS       1<.=IHGFEDCBA*$-RQPONMLKJ%,&ZYXWVUTS/@#0987654321<.=IHGF
                EDCBA*$-RQPONMLKJ%,&ZYXWVUTS/@#0987654321<.=IHGFEDCBA*$-
                RQPONMLKJ%,&ZYXWVUTS/@#0987654321<.=IHGFEDCBA*$-RQPONMLK
                J%,&ZYXWVUTS/@#0987654321<.=IHGFEDCBA*$-RQPONMLKJ%,&ZYXW
                VUTS/@#0987654321<.=IHGFEDCBA*$-RQPONMLKJ%,&ZYXWVUTS/@#0
                987654321<.=IHGFEDCBA*$-RQPONMLKJ%,&ZYXWVUTS/23098765432
                1<.=IHGFEDCBA*$-RQPONMLKJ%,&ZYXWVUTS/@#0987654321<.=IHGF
                EDCBA*$-RQPONMLKJ%,&ZYXWVUTS/@#098765432
STD2 FCB        LPI=6,                                                    C
                LNCH=((4,1),(10,2),(16,3),(22,4),(28,5),(34,6),(40,7),    C
                (46,8),(52,10),(58,11),(64,12),(66,9)),                   C
                FORMEND=66
      END
```

The control statements are discussed below:

- DFN specifies the channel and unit number of the 3211 and specifies that lowercase letters are not to be printed as uppercase letters when the lowercase print train is not available.
- UCS specifies the characters to be loaded into the UCS buffer.
- FCB specifies the values to be loaded into the forms control buffer.

ICAPRTBL Example 2

In this example, a 3211 UCS image (P11) and an IBM standard FCB image are loaded into the UCS and FCB buffers by specifying images via the UCS and FCB parameter of the DFN statement.

```
JOB LOAD 3211 P11 IMAGE
DFN UCS=P11,ADDR=004,FCB=STD1
END
```

The DFN control statement is discussed below:

- By omitting the DEVT parameter, the default device type is 3211.
- The UCS parameter specifies the UCS *image-id* to be loaded into the UCS buffer from standard image tables provided by the utility.

- The ADDR parameter specifies the channel and unit number of the 3211.
- By omitting the FOLD parameter, the default FOLD value N is selected, specifying that lowercase letters are not to be printed as uppercase letters when the lowercase print train is not available.
- The FCB parameter specifies the standard FCB *image-id* (STD1) to be loaded into the FCB buffer from standard image tables provided by the utility.

ICAPRTBL Example 3

In this example, a 3203-4 UCS image (AN by default) and a standard FCB image (STD2 by default) are loaded into the UCS and FCB buffers.

```
JOB
DFN DEVT=3203-4,ADDR=002
END
```

The DFN statement is discussed below:

- The DEVT parameter specifies the device type as 3203-4.
- The ADDR parameter specifies the channel and unit number of the 3203.
- By omitting the FOLD parameter, the default FOLD value N is selected specifying that lowercase letters are not to be printed as uppercase letters when the lowercase print train is not available.
- By omitting both a UCS statement and the UCS parameter, the default 3203 UCS image (AN) is loaded into the UCB buffer from standard image tables provided by the utility.
- By omitting both an FCB statement and the FCB parameter, the default FCB image (STD2) is loaded into the FCB buffer from standard image tables provided by the utility.

ICAPRTBL Example 4

In this example, a 3203-4 UCS image (AN by default) and a provided FCB image are loaded, respectively, into the UCS and FCB buffers.

```
                                                            72
      JOB        3203-4 AN USER FCB
USER  FCB        FORMEND=88,LPI=8,LNCH=((4,1),(12,2),        C
                 (20,3),(28,4),(36,5),(44,6),(52,7),         C
                 (60,8),(68,10),(76,11),(84,12),(88,9))
      DFN FOLD=Y,                                            C
          FCB=STD1,                                          C
          ADDR=003,                                          C
          DEVT=3203-4
      END
```

The control statements are discussed below:

- The FCB statement specifies the values to be loaded into the forms control buffer.
- The specification of the FCB parameter on the DFN statement is overridden by the FCB statement specification.
- The DEVT parameter of the DFN statement specifies the device type as 3203.
- The ADDR parameter specifies the channel and unit number of the 3203.
- The FOLD parameter specifies that lowercase letters are to be printed as uppercase letters when the lowercase print train is not available.

- By omitting both a UCS statement and the UCS parameter of the DFN statement, the default 3203-4 UCS image (AN) is loaded from standard image tables provided by the utility.

IEBCOMPR PROGRAM

IEBCOMPR is a data set utility used to compare two sequential or two partitioned data sets at the logical record level to verify a backup copy. Fixed, variable, or undefined records from blocked or unblocked data sets or members can also be compared.

Two sequential data sets are considered *equal,* that is, are considered to be identical, if:

- The data sets contain the same number of records, and,
- Corresponding records and keys are identical.

If these conditions are not met, an unequal comparison results. If records are unequal, the record and block numbers, the names of the DD statements that define the data sets, and the unequal records are listed in a message data set. Ten successive unequal comparisons terminate the job step unless a user routine is provided to handle error conditions.

Two partitioned data sets are considered equal if:

- Corresponding members contain the same number of records.
- Note lists are in the same position within corresponding members.
- Corresponding records and keys are identical.

If these conditions are not met, an unequal comparison results. If records are unequal, the record and block numbers, the names of the DD statements that define the data sets, and the unequal records are listed in a message data set. After ten successive unequal comparisons, processing continues with the next member unless a user routine is provided to handle error conditions.

Partitioned data sets can be compared only if all the names in one or both of the directories have counterpart entries in the other directory. The comparison is made on members identified by these entries and corresponding user data.

Figure 5-1 shows the directories of two partitioned data sets. Directory 2 contains corresponding entries for all the names in Directory 1; therefore, the data sets can be compared.

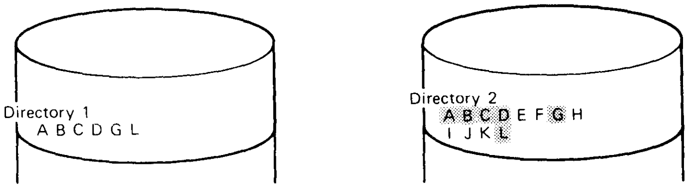


Figure 5-1. Partitioned Directories Whose Data Sets Can Be Compared Using IEBCOMPR

Figure 5-2 shows the directories of two partitioned data sets. Each directory contains a name that has no corresponding entry in the other directory; therefore, the data sets cannot be compared, and the job step is terminated.

User exits are provided for optional user routines to process user labels, handle error conditions, and modify source records. See "Appendix A: Exit Routine Linkage" for a discussion of the linkage conventions to be followed when user routines are used.

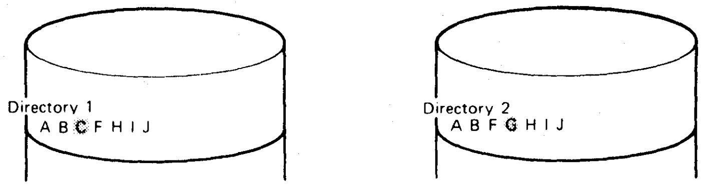


Figure 5-2. Partitioned Directories Whose Data Sets Cannot Be Compared Using IEBCOMPR

At the completion or termination of IEBCOMPR, the highest return code encountered within the program is passed to the calling program.

Input and Output

IEBCOMPR uses the following input:

- Two sequential or two partitioned data sets to be compared.
- A control data set that contains utility control statements. This data set is required if the input data sets are partitioned or if user routines are used.

IEBCOMPR produces as output a message data set that contains informational messages (for example, the contents of utility control statements), the results of comparisons, and error messages.

IEBCOMPR provides a return code to indicate the results of program execution. The return codes and their meanings are:

- 00, which indicates successful completion.
- 08, which indicates an unequal comparison. Processing continues.
- 12, which indicates an unrecoverable error. The job step is terminated.
- 16, which indicates that a user routine passed a return code of 16 to IEBCOMPR. The job step is terminated.

Control

IEBCOMPR is controlled by job control statements and utility control statements. The job control statements are required to execute or invoke IEBCOMPR and to define the data sets that are used and produced by IEBCOMPR. The utility control statements are used to indicate the input data set organization (that is, sequential or partitioned), to identify any user routines that may be provided, and to indicate whether user labels are to be treated as data.

Job Control Statements

Figure 5-3 shows the job control statements necessary for using IEBCOMPR.

One or both of the input data sets can be passed from a preceding job step.

Input data sets residing on different device types can be compared. Input data sets with a sequential organization written at different densities can also be compared.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEBCOMPR) or, if the job control statements reside in a procedure library, the procedure name.
SYSPRINT DD	Defines a sequential message data set, which can be written to a system output device, a tape volume, or a direct access volume.
SYSUT1 DD	Defines an input data set to be compared.
SYSUT2 DD	Defines an input data set to be compared.
SYSIN DD	Defines the control data set or specifies DUMMY if the input data sets are sequential and no user routines are provided. The control data set normally resides in the input stream; however, it can be defined as a member within a library of partitioned members.

Figure 5-3. IEBCOMPR Job Control Statements

Utility Control Statements

The utility control statements used to control IEBCOMPR are:

Statement	Use
COMPARE	Indicates the organization of a data set.
EXITS	Identifies user exit routines to be used.
LABELS	Indicates whether user labels are to be treated as data by IEBCOMPR.

Figure 5-4. IEBCOMPR Utility Control Statements

COMPARE Statement

The COMPARE statement is used to indicate the organization of data sets to be compared.

The COMPARE statement, if included, must be the first utility control statement. COMPARE is required if the EXITS or LABELS statement is used or if the input data sets are partitioned data sets.

The format of the COMPARE statement is:

[*label*] **COMPARE TYPORG={<u>PS</u> | PO}**

EXITS Statement

The EXITS statement is used to identify any user exit routines to be used. The EXITS statement is required if a user exit routine is to be used. If more than one valid EXITS statement is included, all but the last EXITS statement are ignored. For a discussion of the processing of user labels as data set descriptors, see "Appendix D: Processing User Labels."

The format of the EXITS statement is:

[*label*] **EXITS** [**INHDR=***routinename*]
[**,INTLR=***routinename*]
[**,ERROR=***routinename*]
[**,PRECOMP=***routinename*]

LABELS Statement

The LABELS statement specifies whether user labels are to be treated as data by IEBCOMPR. For a discussion of this option, refer to "Processing User Labels as Data" in "Appendix D: Processing User Labels."

The format of the LABELS statement is:

[*label*] **LABELS [DATA={YES | NO | ALL | ONLY}]**

Note: LABELS DATA=NO must be specified to make standard user label (SUL) exits inactive when input/output data sets with nonstandard labels (NSL) are to be processed.

If more than one valid LABELS statement is included, all but the last LABELS statement are ignored.

Operands	Applicable Control Statements	Description of Operands/Parameters
DATA	LABELS	**DATA={YES | NO | ALL | ONLY}** specifies whether user labels are to be treated as data. The values that can be coded are: **YES** specifies that any user labels that are not rejected by a user's label processing routine are to be treated as data. Processing of labels as data stops in compliance with standard return codes. **NO** specifies that user labels are not to be treated as data. **ALL** specifies that user labels are to be treated as data regardless of any return code. A return code of 16 causes IEBCOMPR to complete processing of the remainder of the group of user labels and to terminate the job step. **ONLY** specifies that only user header labels are to be treated as data. User header labels are processed as data regardless of any return code. The job terminates upon return from the OPEN routine.
ERROR	EXITS	**ERROR=***routinename* specifies the symbolic name of a routine that is to receive control after each unequal comparison for error handling. If this parameter is omitted and ten consecutive unequal comparisons occur while IEBCOMPR is comparing sequential data sets, processing is terminated; if the input data sets are partitioned, processing continues with the next member.
INHDR	EXITS	**INHDR=***routinename* specifies the symbolic name of a routine that processes user input header labels.
INTLR	EXITS	**INTLR=***routinename* specifies the symbolic name of a routine that processes user input trailer labels.
PRECOMP	EXITS	**PRECOMP=***routinename* specifies the symbolic name of a routine that processes logical records (physical blocks in the case of VS or VBS records longer than 32K bytes) from either or both of the input data sets before they are compared.
TYPORG	COMPARE	**TYPORG={PS | PO}** specifies the organization of the input data sets. The values that can be coded are: **PO** specifies that the input data sets are partitioned data sets. **PS** specifies that the input data sets are sequential data sets.

Restrictions

- The SYSPRINT DD statement must be present for each use of IEBCOMPR.
- The SYSIN DD statement is required.
- The logical record lengths of the input data sets must be identical; otherwise, unequal comparisons result. The block sizes of the input data sets can differ; however, block sizes must be multiples of the logical record length.
- The block size specified in the SYSPRINT DD statement must be a multiple of 121. The block size specified in the SYSIN DD statement must be a multiple of 80.

IEBCOMPR Examples

The examples that follow illustrate some of the uses of IEBCOMPR. Figure 5-5 can be used as a quick reference guide to IEBCOMPR examples. The numbers in the "Example" column point to examples that follow. **Note:** Examples which use *disk* or *tape* in place of actual device-ids, must be changed before use. See the Device Support section, in the Introduction to this manual, for valid device-id notation.

Operation	Data Set Organization	Devices	Comments	Example
COMPARE	Sequential	9-track Tape	No user routines. Blocked input.	1
COMPARE	Sequential	7-track Tape	No user routines. Blocked input.	2
COMPARE	Sequential	7-track and 9-track Tape	User routines. Blocked input. Different density tapes.	3
COMPARE	Sequential	Card Reader, 9-track Tape	No user routines. Blocked input.	4
COMPARE	Partitioned	Disk	No user routines. Blocked input.	5
COPY (using IEBCOPY) and COMPARE	Sequential	9-track Tape	No user routines. Blocked input. Two job steps; data sets are passed to second job step.	6
COPY (using IEBCOPY) and COMPARE	Partitioned	Disk	User routine. Blocked input. Two job steps; data sets are passed to second job step.	7

Figure 5-5.IEBCOMPR Example Directory

IEBCOMPR Example 1

In this example, two sequential data sets that reside on 9-track tape volumes are to be compared.

```
//TAPETAPE JOB   09#660,SMITH
//         EXEC  PGM=IEBCOMPR
//SYSPRINT DD    SYSOUT=A
//SYSUT1   DD    UNIT=tape,LABEL=( ,NL),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000),
// DISP=(OLD,KEEP),VOLUME=SER=001234
//SYSUT2   DD    UNIT=tape,LABEL=( ,NL),DISP=(OLD,KEEP),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=1040),
// VOLUME=SER=001235
//SYSIN     DD   DUMMY
/*
```

Because no user routines are to be used and the input data sets have a sequential organization, utility control statements are not used.

The control statements are discussed below:

- SYSUT1 DD defines an input data set, which resides on an unlabeled, 9-track tape volume. The blocked data set was originally written at a density of 800 bits per inch.
- SYSUT2 DD defines an input data set, which resides on an unlabeled, 9-track tape volume. The blocked data set was originally written at a density of 800 bits per inch.
- SYSIN DD defines a dummy data set.

IEBCOMPR Example 2

In this example, two sequential data sets that reside on 7-track tape volumes are to be compared.

```
//TAPETAPE JOB  09#660,SMITH
//          EXEC PGM=IEBCOMPR
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=SET1,LABEL=(2,SUL),DISP=(OLD,KEEP),
// VOL=SER=001234,DCB=(DEN=2,RECFM=FB,LRECL=80,
// BLKSIZE=2000,TRTCH=C),UNIT=2400
//SYSUT2   DD   DSNAME=SET1,LABEL=(,SUL),DISP=(OLD,KEEP),
// VOL=SER=001235,DCB=(DEN=2,RECFM=FB,LRECL=80,
// BLKSIZE=2000,TRTCH=C),UNIT=2400
//SYSIN    DD   *
      COMPARE   TYPORG=PS
       LABELS   DATA=ONLY
/*
```

The control statements are discussed below:

- SYSUT1 DD defines an input data set, which resides on a labeled, 7-track tape volume. The blocked data set was originally written at a density of 800 bits per inch with the data converter on.
- SYSUT2 DD defines an input data set, which is the first or only data set on a labeled, 7-track tape volume. The blocked data set was originally written at a density of 800 bits per inch with the data converter on.
- SYSIN DD defines the control data set, which follows in the input stream.
- COMPARE specifies that the input data sets are sequentially organized.
- LABELS specifies that only user header labels are to be compared.

IEBCOMPR Example 3

In this example, two sequential data sets written at different densities on different device types are to be compared.

```
//TAPETAPE JOB   09#660,SMITH
//          EXEC PGM=IEBCOMPR
//SYSPRINT DD    SYSOUT=A
//SYSUT1   DD    DSNAME=SET1,LABEL=(,SUL),DISP=(OLD,KEEP),
// VOL=SER=001234,DCB=(DEN=1,RECFM=FB,LRECL=80,
// BLKSIZE=320,TRTCH=C),UNIT=2400
//SYSUT2   DD    DSNAME=SET2,LABEL=(,SUL),DISP=(OLD,KEEP),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=640),UNIT=tape
// VOLUME=SER=001235
//SYSIN    DD    *
      COMPARE    TYPORG=PS
        EXITS    INHDR=HDRS,INTLR=TLRS
       LABELS    DATA=NO
/*
```

The control statements are discussed below:

- SYSUT1 DD defines an input data set, which is the first or only data set on a labeled, 7-track tape volume. The blocked data set was originally written at a density of 556 bits per inch with the data converter on.
- SYSUT2 DD defines an input data set, which is the first or only blocked data set on a labeled tape volume.
- SYSIN DD defines the control data set, which follows in the input stream.
- COMPARE specifies that the input data sets are sequentially organized.
- EXITS identifies the names of routines to be used to process user input header labels and trailer labels.
- LABELS specifies that the user input header and trailer labels are not to be compared.

IEBCOMPR Example 4

In this example, two sequential data sets (card input and tape input) are to be compared.

```
//CARDTAPE JOB   09#660,SMITH
//          EXEC PGM=IEBCOMPR
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    DUMMY
//SYSUT2   DD    UNIT=tape,VOLUME=SER=001234,LABEL=(,NL),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000),DISP=(OLD,KEEP)
//SYSUT1   DD    DATA
```

(input card data set)

```
/*
```

The control statements are discussed below:

- SYSIN DD defines a dummy control data set. Because no user routines are provided and the input data sets are sequential, utility control statements are not used.
- SYSUT2 DD defines an input data set, which resides on an unlabeled, 9-track tape volume. The blocked data set was originally written at a density of 800 bits per inch.
- SYSUT1 DD defines an input data set (card input).

IEBCOMPR Example 5

In this example, two partitioned data sets are to be compared.

```
//DISKDISK JOB  09#660,SMITH
//          EXEC PGM=IEBCOMPR
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=PDSSET,UNIT=disk,DISP=SHR,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000),
// VOLUME=SER=111112
//SYSUT2   DD   DSNAME=PDSSET,UNIT=disk,DISP=SHR,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000),
// VOLUME=SER=111113
//SYSIN    DD   *
      COMPARE   TYPORG=PO
/*
```

The control statements are discussed below:

- SYSUT1 DD defines an input partitioned data set. The blocked data set resides on a disk volume.
- SYSUT2 DD defines an input partitioned data set. The blocked data set resides on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set consists of one utility control statement.

IEBCOMPR Example 6

In this example, a sequential data set is to be copied and compared in two job steps.

```
//TAPETAPE JOB 09#660,SMITH
//STEPA  EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD DSN=COPYSET,UNIT=tape,DISP=(OLD,PASS),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=640),LABEL=(,SL),
// VOLUME=SER=001234
//SYSUT2   DD DSNAME=COPYSET,DISP=(,PASS),LABEL=(,SL),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=640),UNIT=tape
// VOLUME=SER=001235
//SYSIN    DD DUMMY
/*
//STEPB    EXEC PGM=IEBCOMPR
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD DSNAME=*.STEPA.SYSUT1,DISP=(OLD,KEEP)
//SYSUT2   DD DSNAME=*.STEPA.SYSUT2,DISP=(OLD,KEEP)
//SYSIN    DD DUMMY
/*
```

The first job step copies the data set and passes the original and copied data sets to the second job step. The second job step compares the two data sets.

The control statements for the IEBCOMPR job step are discussed below:

- SYSUT1 DD defines an input data set passed from the preceding job step. The data set resides on a labeled, 9-track tape volume. The blocked data set was originally written at a density of 800 bits per inch.
- SYSUT2 DD defines an input data set passed from the preceding job step. The data set, which was created in the preceding job step, resides on a labeled, 9-track tape volume. The blocked data set was originally written at a density of 800 bits per inch.
- SYSIN DD defines a dummy control data set. Because the input is sequential and no user exits are provided, no utility control statements are required.

IEBCOMPR Example 7

In this example, a partitioned data set is to be copied and compared in two job steps.

The example follows:

```
//DISKDISK JOB 09#660,SMITH
//STEPA    EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD DSNAME=OLDSET,UNIT=disk,DISP=SHR,
// VOLUME=SER=111112,DCB=( RECFM=FB,LRECL=80,
// BLKSIZE=640 )
//SYSUT2   DD DSNAME=NEWMEMS,UNIT=disk,DISP=( ,PASS ),
// VOLUME=SER=111113,SPACE=( TRK,( 5,5,5 ) ),
// DCB=( RECFM=FB,LRECL=80,BLKSIZE=640 )
//SYSUT3   DD UNIT=SYSDA,SPACE=( TRK,( 1 ) )
//SYSUT4   DD UNIT=SYSDA,SPACE=( TRK,( 1 ) )
//SYSIN    DD *
         COPY OUTDD=SYSUT2,INDD=SYSUT1
       SELECT MEMBER=( A,B,D,E,F )
/*
//STEPB    EXEC PGM=IEBCOMPR
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD DSNAME=OLDSET,DISP=( OLD,KEEP )
//SYSUT2   DD DSNAME=NEWMEMS,DISP=( OLD,KEEP )
//SYSIN    DD *
      COMPARE TYPORG=PO
        EXITS ERROR=SEEERROR
/*
```

The first job step copies the data set and passes the original and copied data sets to the second job step. The second job step compares the two data sets.

The control statements for the IEBCOMPR job step are discussed below:

- SYSUT1 DD defines a blocked input data set that is passed from the preceding job step. The data set resides on a disk volume.
- SYSUT2 DD defines a blocked input data set that is passed from the preceding job step. The data set resides on a disk volume.
- SYSUT3 and SYSUT4 define temporary system data sets to be used for work files if needed.
- SYSIN DD defines the control data set, which contains a COMPARE statement and an EXITS statement.
- COMPARE specifies partitioned organization.
- EXITS specifies that a user routine, SEEERROR, is to be used.

Because the input data set names are not identical, the data sets can be retrieved by their data set names.

IEBCOPY PROGRAM

IEBCOPY is a data set utility used to copy one or more partitioned data sets or to merge partitioned data sets. A partitioned data set which is copied to a sequential data set is said to be 'unloaded'. The sequential data set created by an unload operation can be copied to any direct access device. When one or more data sets created by an unload operation are used to re-create a partitioned data set, this is called a 'load' operation. Specific members of a partitioned or unloaded data set can be selected for, or excluded from, a copy, unload, or load process.

IEBCOPY can be used to:

- Create a backup copy of a partitioned data set.
- Copy one or more data sets per copy operation.
- Copy one partitioned data set to a sequential data set (unload).
- Copy one or more data sets created by an unload operation to any direct access device (load).
- Select members from a data set to be copied, unloaded, or loaded.
- Replace identically named members on data sets (except when unloading).
- Replace selected data set members.
- Rename selected members.
- Exclude members from a data set to be copied, unloaded, or loaded.
- Compress partitioned data sets in place (except when the data set is an unloaded data set).
- Merge data sets (except when unloading).
- Re-create a data set that has exhausted its primary, secondary, or directory space allocation.

In addition, IEBCOPY automatically lists the number of unused directory blocks and the number of unused tracks available for member records in the output partitioned data set. If LIST=NO is coded (see "COPY Statement"), the names of copied, unloaded, or loaded members listed by the input data set are suppressed.

Copying Members That Have Aliases

When copying members that have aliases, the following should be noted:

- When the main member and its aliases are copied, they exist on the output partitioned data set in the same relationship they had on the input partitioned data set.
- When members with alias names are copied using the SELECT or EXCLUDE member option, those aliases that are to be selected or excluded must be explicity named.

The rules for replacing or renaming members apply to both aliases and members; no distinction is made between them. Note, however, that the replace option (on the SELECT statement) does not apply to an unload operation.

At the completion or termination of the program, the highest return code encountered within the program is passed to the calling program.

Creating a Backup Copy

IEBCOPY can be used to create a backup copy of a data set by copying (unloading) it to a sequential data set. A partitioned data set can be totally or partially unloaded to any tape volume or direct access device supported by BSAM. A data set is unloaded when physical sequential organization space allocation is specified for the output data set on a direct access device or when the output data set is a tape volume. To unload more than one partitioned data set to the same volume in one execution of IEBCOPY, multiple copy operations must be used and multiple sequential data sets must be allocated on the same volume.

A data set with a physical sequential organization resulting from an unload operation can, in turn, be copied.

Copying Data Sets

IEBCOPY can be used to copy a partitioned data set, totally or in part, from one direct access volume to another. In addition, a data set can be copied to its own volume, provided its data set name is changed. If the data set name is not changed, the data set is compressed in place.

Note that copied members are not reordered. Members are copied in the order in which they exist on the original data set. If the members are to be reordered, IEHMOVE can be used for the copy operation (see "IEHMOVE Program").

Copying or Loading Unloaded Data Sets

Data sets can be copied or loaded, totally or in part, from one or more direct access volumes or tape volumes to a single direct access volume. To copy or load more than one input partitioned data set, specify more than one input data set with the COPY statement. The input data sets are copied or loaded in the order in which they are specified.

Selecting Members to be Copied, Unloaded, or Loaded

Members can be selected from one or more input data sets. Selected members can be copied, unloaded, or loaded from the input data sets specified on the INDD statement preceding a SELECT statement.

Selected members are searched for in a low-to-high (a-to-z) collating sequence, regardless of the order in which they are specified; however, they are copied in the same physical sequence in which they appear on the input partitioned data set.

Once a member of a dataset has been found, no search is made for it on any subsequent input data set. Similarly, when all of the selected members are found, the copy or load step is terminated even though all of the input data sets may not have been searched. For example, if members A and B are specified and A is found on the first of three input data sets, it is not searched for again; if B is found on the second input data set, the copy or load operation is successfully terminated after the second input data set has been processed, although both A and B may also exist on the third input data set.

However, if the first membername is not found on the first input dataset, the search for that member stops and the first dataset is searched for the second member. This process continues until the first input dataset has been searched for all specified members. All the members that were found on the input data set are then processed for copying, unloading, or loading to the output data set. This process is repeated for the second input data set (except that the members that were found on the first input data set are not searched for again).

Note: Only one data set can be processed if an unload operation is to be performed. Multiple unload operations are allowed per job step; multiple INDD statements are *not* allowed per unload operation.

Replacing Identically Named Members

In many copy and load operations, the output partitioned data set may contain members that have names identical to the names of the input partitioned data set members to be copied or loaded. When this occurs, the user may specify that the identically named members are to be copied from the input partitioned data set to replace existing members.

The replace option allows an input member to override an existing member on the output partitioned data set with the same name. The pointer in the output partitioned data set directory is changed to point to the copied or loaded member.

If the replace option is not specified, input members are not copied when they have the same name as a member on the output partitioned data set.

The replace option can be specified on the data set or member level. The level is specified on a utility control statement.

When replace is specified on the data set level with a COPY or INDD statement, the input data is processed as follows:

- In a full copy or load process, all members on an input partitioned data set are copied to an output partitioned data set; members whose names already exist on the output partitioned data set are replaced by the members copied or loaded from the input partitioned data set.
- In a selective copy or load process, all selected input members will be copied to the output dataset, replacing any identically named output dataset members.
- In an exclusive copy process, all nonexcluded members on input partitioned data sets are copied or loaded to an output partitioned data set replacing those duplicate named members on the output partitioned data set.

When replace is specified on the member level (specified on a SELECT statement), only selected members for which replace is specified are copied or loaded, and identically named members on the output partitioned data set are replaced.

There are differences between full, selective, and exclusive copy or load processing. These differences should be remembered when specifying the replace option and all of the output data sets contain member names common to some or all of the input partitioned data sets being copied or loaded. These differences are:

- When a full copy or load is performed, the output partitioned data set contains the replacing members that were on the last input partitioned data set copied.
- When a selective copy or load is performed, the output partitioned data set contains the selected replacing members which were *found* on the earliest input partitioned data set searched. Once a selected member is found, it is not searched for again; therefore, once found, a selected member is copied or loaded. If the same member exists on another input partitioned data set it is not searched for, and hence, not copied or loaded.
- When an exclusive copy or load is performed, the output partitioned data set contains all members, except those specified for exclusion, that were on the last input partitioned data set copied or loaded.

Replacing Selected Members

The user may specify the replace option on either the data set or the member level when members are being selected for copying or loading.

If the replace option is specified on the data set level, all selected members found on the designated input data sets replace identically named members on the output partitioned data set. This is limited by the fact that once a selected member is found it is not searched for again.

If the replace option is specified on the member level, the specified members on the input data set replace identically named members on the output partitioned data set. Once a member is found it is not searched for again. (See "Replacing Identically Named Members" earlier in this chapter.)

Renaming Selected Members

Selected members on input data sets can be copied and renamed on the output data set, the input and output datasets must not be the same. However, in the case of a copy or load operation, if the new name is identical to a member name on the output data set, the input member is not copied or loaded unless the replace option is also specified. See "SELECT Statement" below for information on renaming selected members.

Note: Renaming is not physically done to the input data set directory entry. The output data set directory, however, will contain the new name.

Excluding Members from a Copy Operation

Members from one or more input data sets can be excluded from a copy, unload, or load operation. The excluded member is searched for on every input data set in the copy, unload, or load operation and is always omitted. Members are excluded from the input data sets named on an INDD statement that precedes the EXCLUDE statement. (See "COPY Statement" and "EXCLUDE Statement" in this chapter.)

The replace option can be specified on the data set level in an exclusive copy or load, in which case, nonexcluded members on the input data set replace identically named members on the output data set. See "Replacing Identically Named Members" earlier in this chapter for more information on the replace option.

Compressing a Data Set

A compressed data set is one that does not contain embedded, unused space. After copying or loading one or more input partitioned data sets to a *new* output partitioned data set (by means of a selective, exclusive, or full copy or load that does not involve replacing members), the output partitioned data set contains no embedded, unused space.

To make unused space available, either the entire data set must be scratched or it must be compressed in place. A compressed version can be created by specifying the same data set for both the input and the output parameters in a full copy step. A backup copy of the partitioned data set to be compressed in place should be kept until successful completion of an in-place compression is indicated (by an end-of-job message and a return code of 00).

An in-place compression does not release extents assigned to the data set. Inclusion, exclusion, or renaming of selected members cannot be done during the compression of a partitioned data set.

Note: When the same ddname is specified for the INDD and OUTDD keywords (see "COPY Statement" below) and the DD statement specifies a block size different from the block size specified in the DSCB, the DSCB block size is overridden; however, no physical reblocking or deblocking is performed by IEBCOPY.

Merging Data Sets

A merged data set is one to which an additional member is copied or loaded. It is created by copying or loading the additional members to an existing output partitioned data set; the merge operation—the ordering of the output partitioned data set's directory—is automatically performed by IEBCOPY.

Note: If there is a question about whether or not enough directory blocks are allocated to the output partitioned data set to which an input data set is being merged, the output partitioned data set should be *re-created* with additional directory space prior to the merge operation.

Re-creating a Data Set

A data set can be re-created by copying or loading it and allocating a larger amount of space than was allocated for the original data set. This application of IEBCOPY is especially useful if insufficient directory space was allocated to a data set. Space cannot be allocated in this manner for an existing partitioned data set into which members are being merged.

Input and Output

IEBCOPY uses the following input:

- An input data set, which contains the members to be copied, loaded, merged or unloaded to a sequential data set.
- A control data set, which contains utility control statements. The control data set is required if members are to be selected for or excluded from a copy, unload, load, or merge operation.

If no utility control statements are supplied, a full copy of the input partitioned data set is attempted. In this case, SYSUT1 and SYSUT2 are required ddnames for the input partitioned data set and output partitioned data set, respectively, as described under "Job Control Statements" below.

IEBCOPY produces the following output:

- An output data set, which contains the copied, merged, unloaded, or loaded data. The output data set is either a new data set (from a copy, load, or unload) or an old data set (from a merge, compress-in-place, copy, or load).
- A message data set, which contains informational messages (for example, the names of copied, unloaded, or loaded members) and error messages, if applicable.
- Spill data sets, which are temporary data sets used to provide space when not enough virtual storage is available for the input and/or output partitioned data set directories. These data sets are opened only when needed.

IEBCOPY produces a return code to indicate the results of program execution. The return codes and their meanings are:

- 00, which indicates successful completion.

- 04, which indicates a condition from which recovery may be possible.
- 08, which indicates an unrecoverable error. The job step is terminated.

Control

IEBCOPY is controlled by job control statements and utility control statements.

Job Control Statements

Figure 6-1 shows the job control statements necessary for using IEBCOPY.

Statement	**Use**
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEBCOPY) or, if the job control statements reside in the procedure library, the procedure name. This statement can include optional PARM information to define the size of the buffer to be used; see "PARM Information on the EXEC Statement" below.
SYSPRINT DD	Defines the sequential message data set used for listing statements and messages. This data set can be written to a system output device, a tape volume, or a direct access volume.
anyname1 DD	Defines an input partitioned data set. These DD statements can describe partitioned data sets on direct access devices or sequential data sets, created as a result of unload operations, on tape or direct access devices. The data set can be defined by a data set name, as a cataloged data set, or as a data set passed from a previous job step.
anyname2 DD	Defines an output partitioned data set. These DD statements can describe partitioned data sets on direct access devices or sequential data sets, created as a result of unload operations, on tape or direct access devices.
SYSUT3 DD	Defines a spill data set on a direct access device. SYSUT3 is used when there is no space in virtual storage for some or all of the *current* input partitioned data set's directory entries. SYSUT3 may also be used when not enough space is available in virtual storage for retaining information during table sorting.
SYSUT4 DD	Defines a spill data set on a direct access device. SYSUT4 is used when there is no space in virtual storage for the current output partitioned data set's merged directory and the output partitioned data set is not *new*.
SYSIN DD	Defines the control data set. The control data set normally resides in the input stream; however, it can reside on a system input device, a tape volume, or a direct access volume.

Figure 6-1. IEBCOPY Job Control Statements

Fixed or variable records can be reblocked. Reblocking or deblocking is done if the block size of the input partitioned data set is not equal to the block size of the output partitioned data set.

An unloaded partitioned data set will have a variable spanned record format. When an unloaded data set is subsequently loaded, the output data set will have the same characteristics it had before the unload operation, unless specified differently by the user.

IEBCOPY Unloaded Data Set Block Size

The block size for unloaded data sets is determined as follows:

1. The minimum block size for the unloaded data set is calculated as being equal to the larger of:
   - 284 bytes, or
   - 16 bytes + the block size and key length of the input data set.
2. If a user-supplied block size was specified, and it is larger than the minimum size calculated in step 1 (above), it will be passed to step 3 (below). Otherwise, the minimum size is passed.
3. The block size value passed from step 2 (above) is then compared with the largest block size acceptable to the output device. If the output device capacity is less than the block size passed in step 2, the unloaded data set block size is set to the maximum allowed for the output device.
4. The logical record length (LRECL) is then set to the block size minus four (4) bytes.

Note: Reference source: IEBCOPY module IEBLDUL.

For unload and load operations, requests are handled in the same way as for a copy operation.

Figure 6-2 shows how input record formats can be changed. In addition, any record format can be changed to the undefined format (in terms of its description in the DSCB).

Input	**Output**
Fixed	Fixed Blocked
Fixed Blocked	Fixed
Variable	Variable Blocked
Variable Blocked	Variable

Figure 6-2. Changing Input Record Format Using IEBCOPY

System data sets should not be compressed in place unless the subject partitioned data set is made *non-sharable*. The libraries in which IEBCOPY resides (SYS1.LINKLIB and SYS1.SVCLIB) must not be compressed by IEBCOPY unless IEBCOPY is first transferred to a JOBLIB.

Refer to *OS/VS2 Data Management Services Guide* for information on estimating space allocations.

Refer to *OS/VS2 Storage Estimates,* or *OS/VS2 System Programming Library: Storage Estimates,* to determine when spill data sets are required; see "Space Allocation" below for a description of how to determine the amount of space to allocate.

IEBCOPY issues a conditional storage request (GETMAIN) of one megabyte for work areas.

PARM Information on the EXEC Statement

The EXEC statement for IEBCOPY can contain PARM information that is used to define the number of bytes used as a buffer. The PARM parameter can be coded:

```
PARM='SIZE=nnnnnnnn[K]'
```

The nnnnnnnn can be replaced by one to eight digits. The K causes the nnnnnnnn to be multiplied by 1024.

If PARM is not specified, or is invalidly specified, or a value below the minimum buffer size is specified, IEBCOPY defaults to the minimum buffer size, which is twice the maximum of the input or output block sizes or four times the input or output track capacities.

The maximum buffer size that can be specified is equal to the storage remaining in the storage area gotten when IEBCOPY issues a conditional one-megabyte storage request (GETMAIN) for work areas and buffers. If the value specified in PARM exceeds this maximum, IEBCOPY defaults to the maximum.

Note: A request for too much buffer storage may result in increased system paging because of a lack of available system page frames. This will degrade overall system performance.

Space Allocation

Sometimes it is necessary to allocate space on spill data sets (SYSUT3 and SYSUT4). The space to be allocated for SYSUT3 depends on the number of members to be copied or loaded. The space to be allocated for SYSUT4 depends on the number of directory blocks to be written to the output data set.

To conserve space on the direct access volume, an initial quantity and a secondary quantity for space allocation may be used, as shown in the following SPACE parameter:

```
SPACE=(c,(x,y))
```

The c value should be a block length of 80 for SYSUT3 and of 256 for SYSUT4. The x value is the number of blocks in the primary allocation, and the y value is the number of blocks in a secondary allocation.

For SYSUT3, $x + 15y$ must be equal to or greater than the number of entries in the largest input partitioned data set in the copy operation, multiplied by 1.05.

For SYSUT4, $x + 15y$ must be equal to or greater than the number of blocks allocated to the largest output partitioned data set directory in the IEBCOPY job step.

For example, if there are 700 members on the largest input partitioned data set, space could be allocated for SYSUT3 as follows:

```
SPACE=(80,(25,45))
```

However, the total amount of space required for SYSUT3 in the worst case is used only if needed. If space is allocated in this manner for SYSUT4, the user must specify in his SYSUT4 DD statement:

```
DCB=(KEYLEN=8)
```

Note that IEBCOPY ignores all other DCB information specified for SYSUT3 and/or SYSUT4. Multivolume SYSUT3 and SYSUT4 data sets are not supported.

The temporary spill data sets may or may not be opened, depending on the amount of of virtual storage available; therefore, the SYSUT3 and SYSUT4 DD statements should always appear in the job stream.

Utility Control Statements

IEBCOPY is controlled by the following utility control statements:

Statement	Use
COPY	Indicates the beginning of a COPY operation.
SELECT	Specifies which members in the input data set are to be copied.
EXCLUDE	Specifies members in the input data set to be excluded from the copy step.

Figure 6-3. IEBCOPY Utility Control Statements

In addition, when INDD, a COPY statement parameter, appears on a card other than the COPY statement, it is referred to as an INDD statement; it can function as a control statement in this context.

Utility control statements may be continued on subsequent cards provided that all the data is contained in columns 2 through 71. Control statement operation and keyword parameters can be abbreviated to their initial letters; COPY, INDD, OUTDD, and LIST can be abbreviated to C, I, O, and L.

COPY Statement

The COPY Statement is required to initiate one or more IEBCOPY copy, unload, or load operations. Any number of operations can follow a single COPY statement; any number of COPY statements can appear within a single job step.

IEBCOPY copy, unload, and load operations are specified by a combination of job control language and utility control statements. The OUTDD and INDD keyword parameters on COPY statements name DD statements that define data sets to be copied, unloaded, or loaded. For example:

```
//COPY      JOB    accountnb,'name',MSGLEVEL=(1,1)
//JOBSTEP   EXEC   PGM=IEBCOPY
//SYSPRINT  DD     SYSOUT=A
//IN        DD     DSN=xxxxx,UNIT=yyyy,VOL=SER=yyyyyy,DISP=OLD
//OUT       DD     DSN=xxxxx,UNIT=yyyy,VOL=SER=yyyyyy,
// DISP=NEW,SPACE=xxxx
//SYSUT3    DD     DSN=TEMP1,UNIT=SYSDA,DISP=(NEW,DELETE),
// SPACE=(CYL,(2,2))
//SYSUT4    DD     DSNAME=TEMP2,UNIT=DA,DISP=(NEW,DELETE),
// SPACE=(CYL,(2,2))
//SYSIN     DD     *
            COPY   OUTDD=OUT,INDD=IN
/*
```

The INDD parameter names the DD statement that identifies the input data set.

The OUTDD parameter names the DD statement that identifies the output data set.

The characteristics of the input and output data sets depend on the operation to be performed, as follows:

- If a data set is to be copied, the input and output data sets must both be partitioned data sets.
- If a data set is to be loaded, the input data set may be either partitioned or sequential; the output data set must be partitioned.
- If a data set is to be unloaded, the input data set must be either a partitioned data set or a sequential data set that was created as a result of a previous unload operation. The output data set may reside on either a direct access or tape volume. If the output data set is to reside on a direct access volume, the organization of the data set must be specified as sequential. To specify sequential organization for a direct access data set, specify the SPACE parameter, omitting the directory or index value.

A COPY statement must precede a SELECT or EXCLUDE statement when members are selected for or excluded from a copy, unload, or load step. In addition, if an input ddname is specified on a separate INDD statement, it must follow the COPY statement and precede the SELECT or EXCLUDE statement to which it applies. If one or more INDD statements are immediately followed by the /* card or another COPY statement, a full copy, unload, or load is invoked onto the most recent output partitioned data set previously specified

IEBCOPY uses a copy operation/copy step concept.[1] A copy operation starts with a COPY statement and continues until either another COPY statement or the end of the control data set is found. Within each copy operation, one or more copy steps are present. Any INDD statement directly following a SELECT or EXCLUDE statement marks the beginning of the next copy step and the end of the preceding copy step within the copy operation. If such an INDD statement cannot be found in the copy operation, then the copy operation consists of only one copy step.

Figure 6-4 shows the copy operation/copy step concept. Two copy operations are shown in the figure: the first begins with the statement containing the name COPOPER1, and the second begins with the statement containing the name COPOPER2.

---

[1] The same applies to an unload or load operation or step.

	Job Control Statements		
1st	COPOPER1	COPY	OUTDD=AA,INDD=ZZ
Copy			INDD=BB,CC
Operation			INDD=DD
			INDD=EE
		SELECT	MEMBER=MEMA,MEMB
		SELECT	MEMBER=MEMC
STEP 1			
			INDD=GG
			INDD=HH
		EXCLUDE	MEMBER=MEMD,MEMH
STEP 2			
2nd	COPOPER2	COPY	OUTDD=YY,I=(MM,PP),LIST=NO
Copy		SELECT	MEMBER=MEMB
Operation			
STEP 1			
			INDD=KK
			INDD=LL,NN
STEP 2			

Figure 6-4. Multiple Copy Operations Within a Job Step

There are two copy steps within the first copy operation shown in Figure 6-4: the first begins with the COPY statement and continues through the two SELECT statements; the second begins with the first INDD statement following the two SELECT statements and continues through the EXCLUDE statement preceding the second COPY statement. There are two copy steps within the second copy operation: the first begins with the COPY statement and continues through the SELECT statement; the second begins with the INDD statement immediately following the SELECT statement and ends with the same /* (delimiter) statement that ended the copy operation.

The format of the COPY statement is:

[*label*] **COPY** **OUTDD**=*ddname*
[,INDD= {*ddname1* [, *ddname2*]... |
ddname1 [, *ddname2*][,(*ddname2* ,**R**)]... |
((*ddname1* ,**R**)[, *ddname2*]...)}]
[,LIST=NO]

Note: The control statement operation and keyword parameters can be abbreviated to their initial letters; for example, COPY can be abbreviated to C and OUTDD can be abbreviated to O. Only one INDD and one OUTDD keyword may be placed on a single card. OUTDD must appear on the COPY statement. When INDD appears on a separate card, no other operands may be specified on that card. If INDD appears on a separate card, it is not preceded by a comma.

If there are no keywords on the COPY card, compatibility with the previous version is implied. In this case, comments may not be placed on this card.

If more than one ddname is specified, the input partitioned data sets are processed in the same sequence as that in which the ddnames are specified.

A full copy, unload, or load is invoked only by specifying different input and output ddnames; that is, by omitting the SELECT or EXCLUDE statement from the copy step.

The compress-in-place function is valid for partitioned data sets. Compress-in-place is normally invoked by specifying the same ddname for both the OUTDD and INDD parameters of a COPY statement. If multiple entries are made on the INDD statement, a compress-in-place will occur if one of the input ddnames is the same as the ddname specified by the OUTDD parameter of the COPY statement, provided that SELECT or EXCLUDE is not specified.

When a compression is invoked by specifying the same ddname for the INDD and OUTDD parameters, and the DD statement specifies a block size that differs from the block size specified in the DSCB, the DSCB block size is overridden; however, no physical reblocking or deblocking is done by IEBCOPY.

SELECT Statement

The SELECT statement specifies members to be selected from input data sets to be copied, loaded, or unloaded to an output data set. This statement is also used to rename and/or replace selected members on the output data set. More than one SELECT statement may be used in succession, in which case the second and subsequent statements are treated as a continuation of the first.

The SELECT statement must follow either a COPY statement that includes an INDD parameter or one or more INDD statements. A SELECT statement cannot appear with an EXCLUDE statement in the same copy, unload, or load step, and it cannot be used with a compress-in-place function.

When a selected member is found on an input data set, it is not searched for again, regardless of whether it the member is copied, unloaded, or loaded. A selected member will not replace an identically named member on the output partitioned data set unless the replace option is specified on either the data set or member level. (For a description of replacing identically named members see "Replacing Identically Named Data Set Members," and "Replacing Selected Members" in this chapter.) In addition, a renamed member will not replace a member on the output partitioned data set that has the same new name as the renamed member, unless the replace option is specified.

The format of the SELECT statement is:

```
[label]SELECT MEMBER=     {[(] name1[, name2][,...][)] |
                            ({(name1,newname[,R])[,...] |
                            (name1, newname )[,... ] |
                            (name1,, R)[,... ]})}
```

where:

MEMBER=
specifies the members to be selected from the input data set. The values that can be coded are:

name
specifies the name of a member that is to be selected in a copy step. Each member name specified within one copy step must be unique; that is, duplicate names cannot be specified as either old names, or new names, or both, under any circumstances.

newname
specifies a new name for a selected member. The member is copied, unloaded, or loaded to the output partitioned data set using its new name. If the name already appears on the output partitioned data set, the member is not copied unless replacement (R) is also specified.

R
specifies that the input member is to replace any identically named member that exists on the output partitioned data set. The replace option is not valid for an unload operation.

The control statement operation and keyword parameter can be abbreviated to their initial letters; SELECT can be abbreviated to S and MEMBER can be abbreviated to M.

To rename a member, the old member name is specified in the SELECT statement, followed by the new name and, optionally, the R parameter. When this option is specified, the *old* member name and *new* member name must be enclosed in a set of parentheses. When any option within parentheses is specified anywhere in the MEMBER field, the entire field, exclusive of the MEMBER keyword, must be enclosed in a second set of parentheses.

EXCLUDE Statement

The EXCLUDE statement specifies members to be excluded from the copy, unload, or load step. Unlike the selective copy, unload, or load, an exclusive copy, unload, or load causes all members specified on each EXCLUDE statement to be omitted from the operation.

More than one EXCLUDE statement may be used in succession, in which case the second and subsequent statements are treated as a continuation of the first. The EXCLUDE statement must follow either a COPY statement that includes an 1480 parameter or one or more INDD statements. An EXCLUDE statement cannot appear with a SELECT statement in the same copy, unload, or load step; however, both may be used following a COPY statement for a copy or load operation. The EXCLUDE statement cannot be used with a compress-in-place function.

The format of the EXCLUDE statement is:

[*label*] **EXCLUDE MEMBER=**[(] *membername1* [, *membername2*]...[)]

The control statement operation and keyword parameter can be abbreviated to their initial letters; EXCLUDE can be abbreviated to E and MEMBER can be abbreviated to M.

Operands	Applicable Control Statements	Description of Operands/Parameters
INDD	**COPY**	**INDD=[(]** *ddname1*[, *ddname2*][,(*ddname3*,**R**)][,...][)] specifies the names of the input partitioned data sets. INDD may, optionally, be placed on a separate card following a COPY statement containing the OUTDD parameter, another INDD statement, a SELECT statement, or an EXCLUDE statement. These values can be coded: *ddname* specifies the ddname, which is specified on a DD statement, of an input data set. For an unload operation, only one ddname may be specified per COPY statement. If more than one ddname is specified in the case of a copy or load operation, the input data sets are processed in the same sequence as the ddnames are specified. **R** specifies that all members to be copied or loaded from this input data set are to replace any identically named members on the output partitioned data set. (In addition, members whose names are not on the output partitioned data set are copied or loaded as usual.) When this option is specified with the INDD parameter, it does not have to appear with the MEMBER parameter (discussed in "SELECT Statement" in this chapter) in a selective copy operation. When this option is specified, the ddname and the R parameter must be enclosed in a set of parentheses; if it is specified with the first ddname in INDD, the entire field, exclusive of the INDD parameter, must be enclosed in a second set of parentheses.
LIST	**COPY**	**LIST=NO** specifies that the names of copied members are not to be listed on on SYSPRINT at the end of each input data set. **Default:** The names of copied members are listed.

<table>
<tr><th>Operands</th><th>Applicable Control Statements</th><th>Description of Operands/Parameters</th></tr>
<tr><td>MEMBER</td><td>SELECT</td><td>**MEMBER**={[(] *name1*[, *name2*][,...][)] |
({(*name1,newname*[,**R**])[,...] |
(*name1,newname*)[,...] |
(*name1* ,,**R**)[,...]})}

specifies the members to be selected from the input data set. The values that can be coded for SELECT are:

name
specifies the name of a member that is to be selected in a copy step. Each member name specified within one copy step must be unique; that is, duplicate names cannot be specified as either old names, or new names, or both, under any circumstances.

newname
specifies a new name for a selected member. The member is copied, unloaded, or loaded to the output partitioned data set using its new name. If the name already appears on the output partitioned data set, the member is not copied unless replacement (R) is also specified.

R
specifies that the input member is to replace any identically named member that exists on the output partitioned data set. The replace option is not valid for an unload operation.</td></tr>
<tr><td></td><td>EXCLUDE</td><td>**MEMBER**=[(] *membername1*[, *membername2*]...[)]
specifies members on the input data set that are not to be copied, unloaded, or loaded to the output data set. The members are not deleted from the input data set unless the entire data set is deleted. (This can be done by specifying DISP=DELETE in the operand field of the input DD job control statement.) Each member name specified within one copy step must be unique.</td></tr>
<tr><td>OUTDD</td><td>COPY</td><td>**OUTDD=** *ddname*

specifies the name of the output partitioned data set. One ddname is required for each copy, unload, or load operation; the ddname used must be specified on a DD statement.</td></tr>
</table>

Restrictions

- IEBCOPY must run from an authorized library because of special storage key requirements for IEBCOPY I/O appendages.
- SYSPRINT and SYSIN are mandatory DD statements. The block size for the SYSPRINT data set must be a multiple of 121. The block size for the SYSIN data set must be a multiple of 80. Any blocking factor may be specified for these data sets, with a maximum allowable block size of 32,767 bytes.
- The SYSPRINT DD statement must define a data set with fixed blocked or fixed records.
- INPUT DD and OUTPUT DD statements are required. There must be one INPUT DD statement for each unique data set used for input and one OUTPUT DD statement for each unique data set used for output in the job step.
- Input data sets cannot be concatenated.
- The SYSIN DD statement must define a data set with fixed block or fixed records.
- Variable spanned and variable block spanned format data sets are not supported.
- The maximum block size for input data sets to be unloaded is 32,767 (input key length + 20).
- VIO is not supported by IEBCOPY for SYSUT4 and partitioned input or output data sets.
- When merging into or compressing system libraries, do not specify DISP=SHR. The results of a merge into or compress of the current SYS1.LINKLIB or SYS1.SVCLIB would be unpredictable.
- IEBCOPY does its own buffering; therefore, coding the BUFNO parameter will cause a JCL error.
- Reblocking or deblocking cannot be done if either the input or the output data set has undefined format records, keyed records, track overflow records, note lists, or user TTRNs, or if compress-in-place is specified.

The compress-in-place function cannot be performed for the following:

- An unloaded data set.
- A data set with track overflow records.
- A data set with keyed records.
- A data set for which reblocking is specified in the DCB parameter.
- An *unmovable* data set.

IEBCOPY Examples

The following examples illustrate some of the uses of IEBCOPY. Figure 6-5 can be used as a quick reference guide to IEBCOPY examples. The numbers in the "Example" column point to examples that follow.

Note: Examples which use *disk* or *tape*, in place of actual device-ids, must be changed before use. See the Device Support section, in the Introduction, to this manual for valid device-id notation.

Operation	Device	Comments	Example
COPY	Disk	Full Copy. The input and output data sets are partitioned.	1
COPY	Disk	Multiple input partitioned data sets. Fixed blocked and fixed record formats.	2
COPY	Disk	All members are to be copied. Identically named members on the output data set are to be replaced. The input and output data sets are partitioned.	3
COPY	Disk	Selected members are to be copied. Variable blocked data set is to be created. Record formats are variable blocked and variable. The input and output data sets are partitioned.	4
COPY	Disk	Selected members are to be copied. One member is to replace an identically named member on the output data set. The input and output data sets are partitioned.	5
COPY	Disk, and 2305 Fixed Head Storage	Selected members are to be copied. Members found on the first input data set replace identically named members on the output data set. The input and output data sets are partitioned.	6
COPY	Disk	Selected members are to be copied. Two members are to be renamed. One renamed member is to replace an identically named member on the output data set. The input and output data sets are partitioned.	7
COPY	Disk	Exclusive Copy. Fixed blocked and fixed record formats. The input and output data sets are partitioned.	8
Unload and Compress-in-place	Disk and Tape	Copy a partitioned data set to tape (unload) and compress-in-place if the first step is successful.	9
COPY and Compress-in-place	Disk	Full copy to be followed by a compress-in-place of the output data set. Replace specified for one input data set. The input and output data sets are partitioned.	10
COPY	Disks	Multiple copy operations. The input and output data sets are partitioned.	11
COPY	Disks	Multiple copy operations.	12
Unload	Disk, and Tape	A partitioned data set is to be unloaded to tape.	13
Load	Tape, and Disk	An unloaded data set is to be loaded to disk.	14
Unload, Load, and COPY	Disk, and Tape	Selected members are to be unloaded, loaded, and copied. The input data set is partitioned; the output data set is sequential.	15

Figure 6-5. IEBCOPY Example Directory

IEBCOPY Example 1

In this example, a partitioned data set (DATASET5) is to be copied from one disk volume to another.
Figure 6-6 shows the input and output data sets before and after processing.

```
//COPY      JOB    06#990,MCEWAN
//JOBSTEP   EXEC   PGM=IEBCOPY
//SYSPRINT  DD     SYSOUT=A
//INOUT4    DD     DSNAME=DATASET4,UNIT=3350,VOL=SER=111112,
// DISP=(NEW,KEEP),SPACE=(TRK,(5,1,2))
//INOUT5    DD     DSNAME=DATASET5,UNIT=3350,VOL=SER=111113,
// DISP=SHR
//SYSUT3    DD     UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4    DD     UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN     DD     *
COPYOPER    COPY   OUTDD=INOUT4,INDD=INOUT5
/*
```

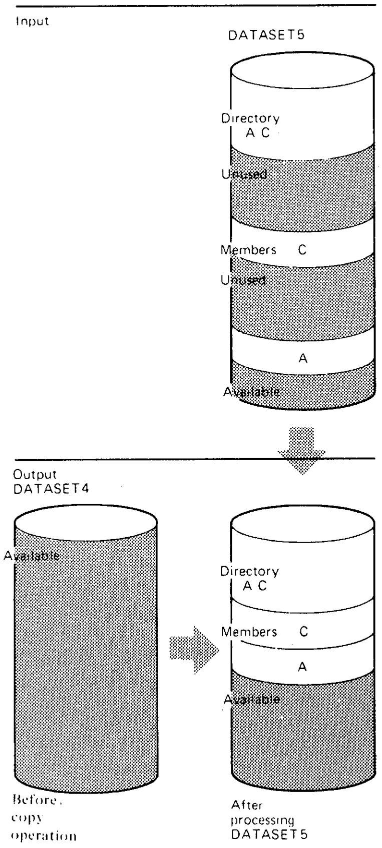


Figure 6-6. Copying a Partitioned Data Set—Full Copy

The control statements are discussed below:

- INOUT4 DD defines a new partitioned data set (DATASET4) that is to be kept after the copy operation. Five tracks are allocated for the data set on a 3350 volume. Two blocks are allocated for directory entries.
- INOUT5 DD defines a partitioned data set (DATASET5), that resides on a 3350 volume and contains two members (A and C).

- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement.
- COPY indicates the start of the copy operation. The absence of a SELECT or EXCLUDE statement causes a default to a full copy. The OUTDD parameter specifies INOUT4 as the DD statement for the output data set (DATASET4); the INDD parameter specifies INOUT5 as the DD statement for the input data set. After the copy operation is finished, the output data set (DATASET4) will contain the same members that are on the input data set (DATASET5); however, there will be no embedded, unused space on DATASET4.

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, it is suggested that the SYSUT3 and SYSUT4 DD statements always appear in the job stream.

IEBCOPY Example 2

In this example, members are to be copied from three input partitioned data sets (DATASET1, DATASET5, and DATASET6) to an existing output partitioned data set (DATASET2). The sequence in which the control statements occur controls the manner and sequence in which partitioned data sets are processed. Figure 6-7 shows the input and output data sets before and after processing.

```
//COPY      JOB    06#990,MCEWAN
//JOBSTEP   EXEC   PGM=IEBCOPY
//SYSPRINT  DD     SYSOUT=A
//INOUT1    DD    DSNAME=DATASET1,UNIT=3330,VOL=SER=111112,
// DISP=SHR
//INOUT5    DD     DSNAME=DATASET5,UNIT=3350,VOL=SER=111114,
// DISP=OLD
//INOUT2    DD     DSNAME=DATASET2,UNIT=3350,VOL=SER=111115,
// DISP=(OLD,KEEP)
//INOUT6    DD     DSNAME=DATASET6,UNIT=3350,VOL=SER=111117,
// DISP=(OLD,DELETE)
//SYSUT3    DD     UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4    DD     UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN     DD     *
COPYOPER    COPY   OUTDD=INOUT2
                   INDD=INOUT1
                   INDD=INOUT6
                   INDD=INOUT5
/*
```

The control statements are discussed below:

- INOUT1 DD defines a partitioned data set (DATASET1). This data set, which resides on a 3330 volume, contains three members (A, B, and F) in fixed format with a logical record length of 80 bytes and a block size of 80 bytes.
- INOUT5 DD defines a partitioned data set (DATASET5), which resides on a 3350 volume. This data set contains two members (A and C) in fixed blocked format with a logical record length of 80 bytes and a block size of 160 bytes.
- INOUT2 DD defines a partitioned data set (DATASET2), which resides on a 3350 volume. This data set contains two members (C and E) in fixed blocked format. The members have a logical record length of 80 bytes and a block size of 240 bytes.

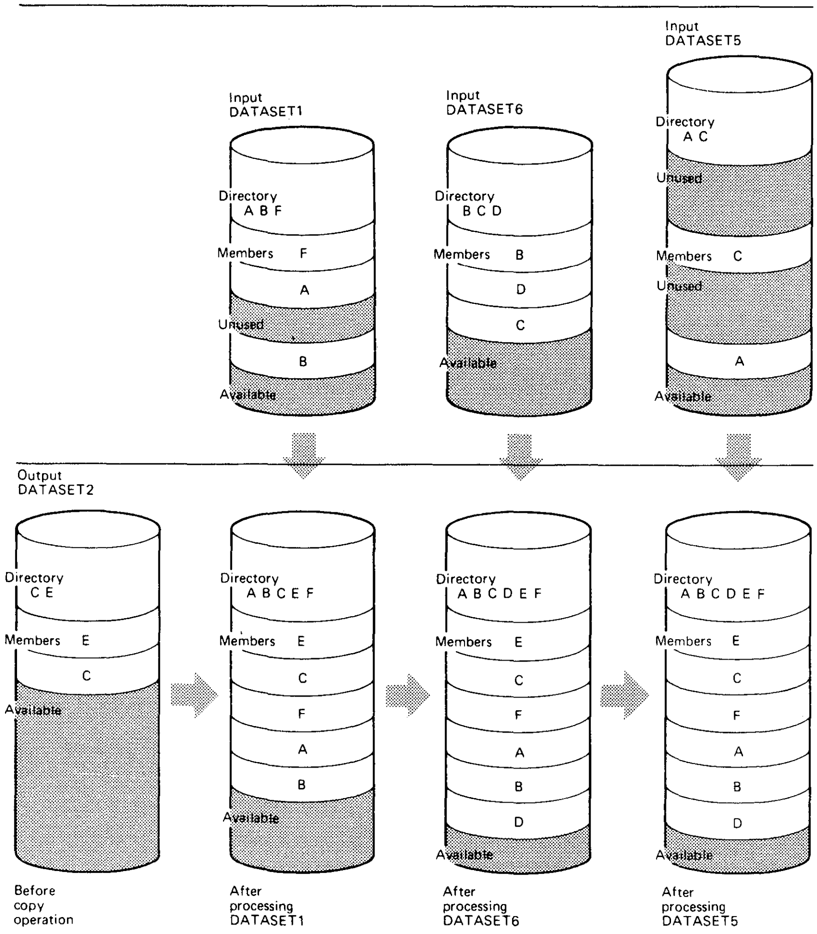


Figure 6-7. Copying from Three Input Partitioned Data Sets

- INOUT6 DD defines a partitioned data set (DATASET6), which resides on a 3350 volume. This data set contains three members (B, C, and D) in fixed blocked format with a logical record length of 80 bytes and a block size of 400 bytes. This data set is to be deleted when processing is completed.
- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement and three INDD statements.
- COPY indicates the start of the copy operation. The absence of a SELECT or EXCLUDE statement causes a default to a full copy. The OUTDD parameter specifies INOUT2 as the DD statement for the output data set (DATASET2).
- The first INDD statement specifies INOUT1 as the DD statement for the first input data set (DATASET1) to be processed. All members (A, B, and F) are copied to the output data set (DATASET2).
- The second INDD statement specifies INOUT6 as the DD statement for the second input data set (DATASET6) to be processed. Processing occurs, as follows: (1) members B and C, which already exist on DATASET2, are not

copied to the output data set (DATASET2), (2) member D is copied to the output data set (DATASET2), and (3) all members on DATASET6 are lost when the data set is deleted.

- The third INDD statement specifies INOUT5 as the DD statement for the third input data set (DATASET5) to be processed. No members are copied to the output data set (DATASET2) because all of them exist on DATASET2.

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, it is suggested that the SYSUT3 and SYSUT4 DD statements always appear in the job stream.

IEBCOPY Example 3

In this example, members are to be copied from an input partitioned data set (DATASET6) to an existing output partitioned data set (DATASET2). In addition, all copied members are to replace identically named members on the output partitioned data set.

Figure 6-8 shows the input and output data sets before and after processing.

The example follows:

```
//COPY      JOB    06#990,MCEWAN
//JOBSTEP   EXEC   PGM=IEBCOPY
//SYSPRINT  DD     SYSOUT=A
//INOUT2    DD     DSNAME=DATASET2,UNIT=3330-1,VOL=SER=111113,
// DISP=OLD
//INOUT6    DD     DSNAME=DATASET6,UNIT=3350,VOL=SER=111117,
// DISP=(OLD,KEEP)
//SYSUT3    DD     UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4    DD     UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN     DD     *
COPYOPER    COPY   OUTDD=INOUT2
                   INDD=((INOUT6,R))
/*
```

The control statements are discussed below:

- INOUT2 DD defines a partitioned data set (DATASET2), which resides on a 3330-1 volume. This data set contains two members (C and E).
- INOUT6 DD defines a partitioned data set (DATASET6), which resides on a 3350 volume. This data set contains three members (B, C, and D).
- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement and an INDD statement.
- COPY indicates the start of the copy operation. The absence of a SELECT or EXCLUDE statement causes a default to a full copy. The OUTDD parameter specifies INOUT2 as the DD statement for the output data set (DATASET2).
- INDD specifies INOUT6 as the DD statement for the input data set (DATASET6). Members B, C, and D are copied to the output data set (DATASET2). The pointer in the output data set directory is changed to point to the new (copied) member C; thus, the space occupied by the old member C is embedded unused space. Member C is copied even though the output data set already contains a member named "C" because the replace option is specified

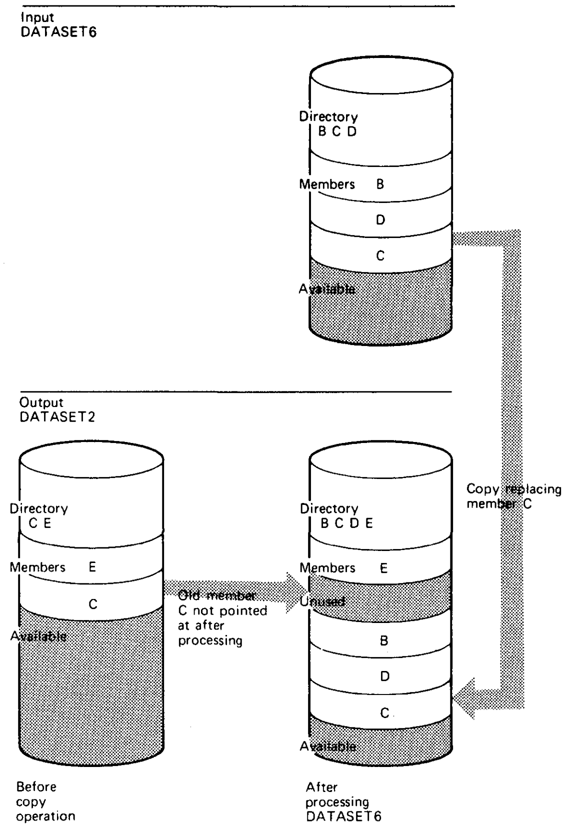


Figure 6-8. Copy Operation with "Replace" Specified on the Data Set Level

for all identically named members on the input data set; that is, the replace option is specified on the data set level.

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, it is suggested that the SYSUT3 and SYSUT4 DD statements always appear in the job stream.

IEBCOPY Example 4

In this example, five members (A, C, D, E, and G) are to be selected from two input partitioned data sets (DATASET6 and DATASET2) to be copied to a new output partitioned data set (DATASET4). Figure 6-9 shows the input and output data sets before and after processing.

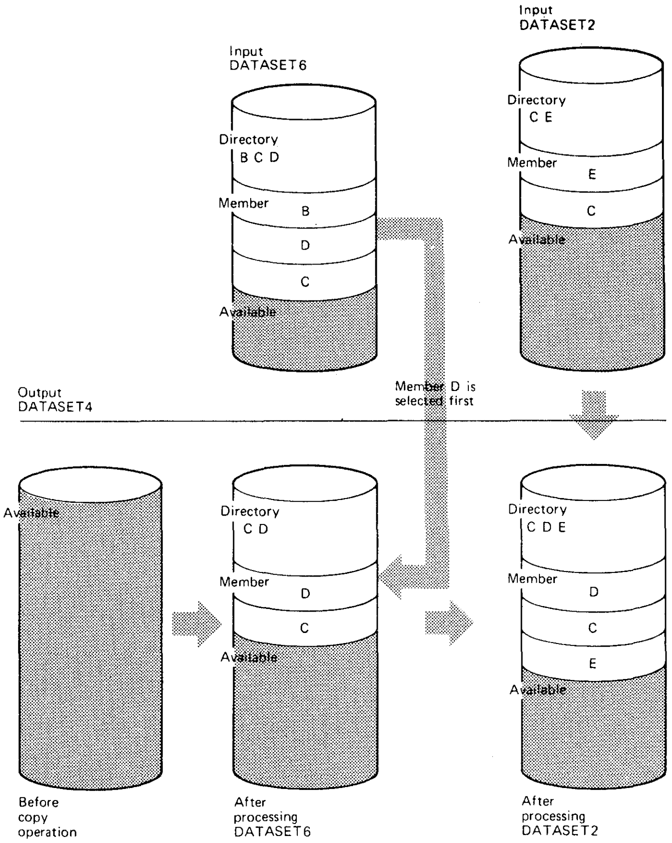


Figure 6-9. Copying Selected Members with Reblocking and Deblocking

```
//COPY      JOB     06#990,MCEWAN
//JOBSTEP   EXEC    PGM=IEBCOPY
//SYSPRINT  DD      SYSOUT=A
//INOUT2    DD      DSNAME=DATASET2,UNIT=3330,VOL=SER=111114,
// DISP=( OLD,DELETE )
//INOUT6    DD      DSNAME=DATASET6,UNIT=3350,VOL=SER=111117,
// DISP=( OLD,KEEP )
//INOUT4    DD      DSNAME=DATASET4,UNIT=3350,VOL=SER=111116,
// DISP=( NEW,KEEP ),SPACE=( TRK,( 5,,2 )),
// DCB=( RECFM=VB,LRECL=96,BLKSIZE=300 )
//SYSUT3    DD      UNIT=SYSDA,SPACE=( TRK,( 1 ))
//SYSUT4    DD      UNIT=SYSDA,SPACE=( TRK,( 1 ))
//SYSIN     DD      *
COPYOPER    COPY    OUTDD=INOUT4
                    INDD=INOUT6
                    INDD=INOUT2
            SELECT  MEMBER=C,D,E,A,G
/*
```

The control statements are discussed below:

- INOUT2 DD defines a partitioned data set (DATASET2), which resides on a 3330 volume. This data set contains two members (C and E) in variable blocked format with a logical record length of 96 bytes and a block size of 500 bytes. This data set is to be deleted when processing is completed.
- INOUT6 DD defines a partitioned data set (DATASET6), which resides on a 3350 volume. This data set contains three members (B, C, and D) in variable format with a logical record length of 96 bytes and a block size of 100 bytes.

- INOUT4 DD defines a partitioned data set (DATASET4). This data set is new and is to be kept after the copy operation. Five tracks are allocated for the data set on a 3350 volume. Two blocks are allocated for directory entries. In addition, records are to be copied to this data set in variable blocked format with a logical record length of 96 bytes and a block size of 300 bytes.
- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement, two INDD statements, and a SELECT statement.
- COPY indicates the start of the copy operation. The presence of a SELECT statement causes a selective copy. The OUTDD parameter specifies INOUT4 as the DD statement for the output data set (DATASET4).
- The first INDD statement specifies INOUT6 as the DD statement for the first input data set (DATASET6) to be processed. The members specified on the SELECT statement are searched for. The found members (C and D) are copied to the output data set (DATASET4) in the order in which they reside on the input data set, that is, in TTR order. In this case, member D is copied first, and then member C is copied.
- The second INDD statement specifies INOUT2 as the DD statement for the second input data set (DATASET2) to be processed. The members specified on the SELECT statement and not found on the first input data set are searched for. The found member (E) is copied onto the output data set (DATASET4). All members on DATASET2 are lost when the data set is deleted.
- SELECT specifies the members to be selected from the input data sets (DATASET6 and DATASET2) to be copied to the output data set (DATASET4).

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, it is suggested that the SYSUT3 and SYSUT4 DD statements always appear in the job stream.

IEBCOPY Example 5

In this example, two members (A and B) are to be selected from two input partitioned data sets (DATASET5 and DATASET6) to be copied to an existing output partitioned data set (DATASET1). Member B is to replace an identically named member that already exists on the output data set. Figure 6-10 shows the input and output data sets before and after processing.

```
//COPY       JOB      06#990,MCEWAN
//JOBSTEP    EXEC     PGM=IEBCOPY
//SYSPRINT   DD       SYSOUT=A
//INOUT1     DD       DSNAME=DATASET1,UNIT=3330,VOL=SER=111112,
// DISP=(OLD,KEEP)
//INOUT6     DD       DSNAME=DATASET6,UNIT=3350,VOL=SER=111115,
// DISP=OLD
//INOUT5     DD       DSNAME=DATASET5,UNIT=3330,VOL=SER=111116,
// DISP=(OLD,KEEP)
//SYSUT3     DD       UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4     DD       UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN      DD       *
COPYOPER     COPY     OUTDD=INOUT1
                      INDD=INOUT5,INOUT6
             SELECT   MEMBER=((B,,R),A)
/*
```

The control statements are discussed below:

- INOUT1 DD defines a partitioned data set (DATASET1). This data set resides on a 3330 volume and contains three members (A, B, and F).
- INOUT6 DD defines a partitioned data set (DATASET6). This data set resides on a 3350 volume and contains three members (B, C, and D).
- INOUT5 DD defines a partitioned data set (DATASET5). This data set resides on a 3330 volume and contains two members (A and C).
- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement, an INDD statement, and a SELECT statement.
- COPY indicates the start of the copy operation. The presence of a SELECT statement causes a selective copy. The OUTDD parameter specifies INOUT1 as the DD statement for the output data set (DATASET1).
- INDD specifies INOUT5 as the DD statement for the first input data set (DATASET5) to be processed and INOUT6 as the DD statement for the second input data set (DATASET6) to be processed. Processing occurs, as follows: (1) selected members are searched for on DATASET5, (2) member A is found, but is not copied to the output data set because it already exists on DATASET2 and the replace option is not specified, (3) selected members not found on DATASET5 are searched for on DATASET6, and (4) member B is found and copied to the output data set (DATASET1), even though a member named B already exists on the output data set, because the replace option is specified for member B on the member level. The pointer in the output data set directory is changed to point to the new (copied) member B; thus, the space occupied by the old member B is unused.

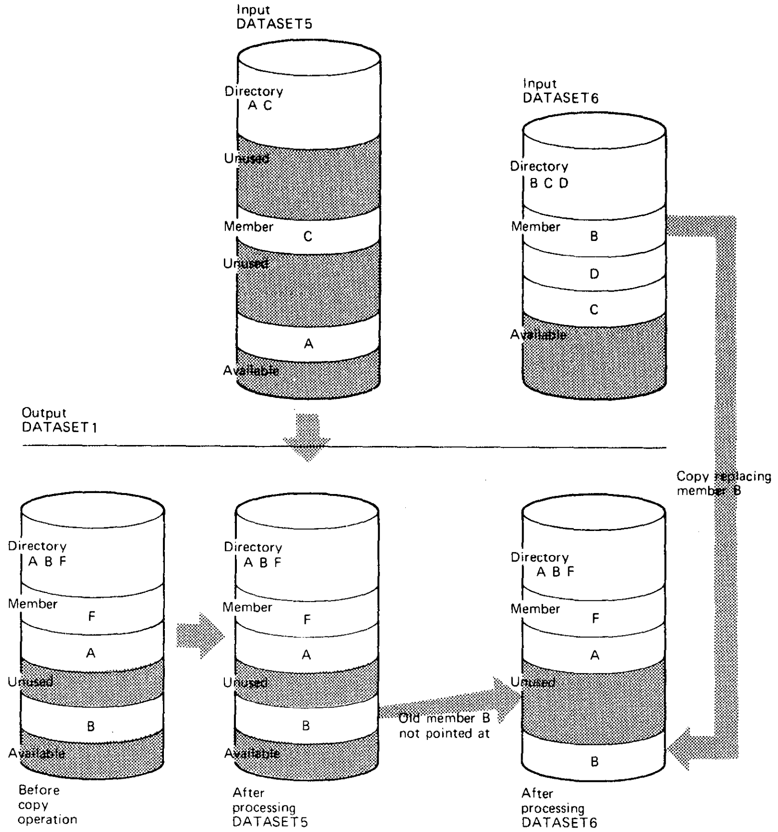


Figure 6-10. Selective Copy with "Replace" Specified on the Member Level

- SELECT specifies the members to be selected from the input data sets (DATASET5 and DATASET6) to be copied to the output data set (DATASET1).

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, it is suggested that the SYSUT3 and SYSUT4 DD statements always appear in the job stream.

IEBCOPY Example 6

In this example, two members (A and B) are to be selected from two input partitioned data sets (DATASET5 and DATASET6) to be copied to an existing output partitioned data set (DATASET1). All members found on DATASET5 are to replace identically named members on DATASET1. Figure 6-11 shows the input and output data sets before and after processing.

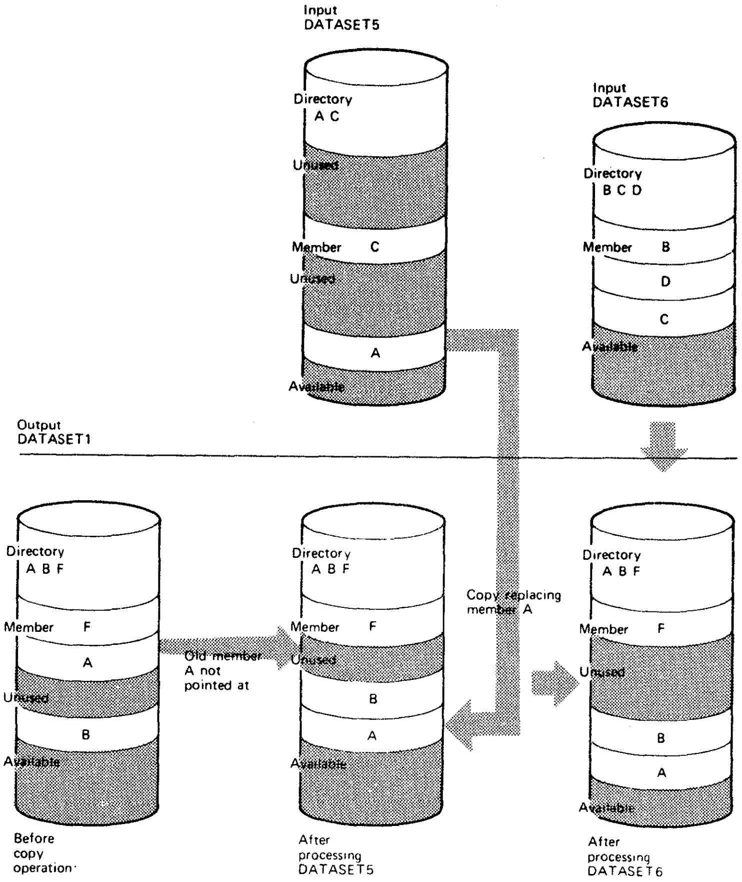


Figure 6-11. Selective Copy with "Replace" Specified on the Data Set Level

```
//COPY      JOB      06#990,MCEWAN
//JOBSTEP   EXEC     PGM=IEBCOPY
//SYSPRINT  DD       SYSOUT=A
//INOUT1    DD       DSNAME=DATASET1,UNIT=3350,VOL=SER=111112,
// DISP=(OLD,KEEP)
//INOUT5    DD       DSNAME=DATASET5,UNIT=3330,VOL=SER=111114,
// DISP=(OLD,DELETE)
//INOUT6    DD       DSNAME=DATASET6,UNIT=2305,VOL=SER=111115,
// DISP=(OLD,KEEP)
//SYSUT3    DD       UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4    DD       UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN     DD       *
COPYOPER    COPY     OUTDD=INOUT1
                     INDD=((INOUT5,R),INOUT6)
            SELECT   MEMBER=(A,B)
/*
```

The control statements are discussed below:

- INOUT1 DD defines a partitioned data set (DATASET1). This data set resides on a 3350 volume and contains three members (A, B, and F).
- INOUT5 DD defines a partitioned data set (DATASET5). This data set contains two members (A and C) and resides on a 3330 volume. This data set is to be deleted when processing is completed.
- INOUT6 DD defines a partitioned data set (DATASET6). This data set contains three members (B, C, and D) and resides on a 2305 volume.

- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement, an INDD statement, and a SELECT statement.
- COPY indicates the start of the copy operation. The presence of a SELECT statement causes a selective copy. The OUTDD operand specifies INOUT1 as the DD statement for the output data set (DATASET1).
- INDD specifies INOUT5 as the DD statement for the first input data set (DATASET5) to be processed and INOUT6 as the statement for the second input data set (DATASET6) to be processed. Processing occurs, as follows: (1) selected members are searched for on DATASET5, (2) member A is found and copied to the output data set (DATASET1) because the replace option was specified on the data set level for DATASET5, (3) member B, which was not found on DATASET5 is searched for and found on DATASET6, (4) member B is not copied because DATASET1 already contains a member called member B and the replace option is not specified for DATASET6. The pointer in the output data set directory is changed to point to the new (copied) member A; thus, the space occupied by the old member A is unused.
- SELECT specifies the members to be selected from the input data sets (DATASET5 and DATASET6) to be copied to the output data set (DATASET1).

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, it is suggested that the SYSUT3 and SYSUT4 DD statements always appear in the job stream.

IEBCOPY Example 7

In this example, four members (A, B, C, and D) are to be selected from an input partitioned data set (DATASET6) to be copied to an existing output partitioned data set (DATASET3). Member B is to be renamed H; member C is to be renamed J; and member D is to be renamed K. In addition, member C (renamed J) is to replace the identically named member (J) on the output partitioned data set. Figure 6-12 shows the input and output data sets before and after processing.

```
//COPY      JOB      #990,MCEWAN
//JOBSTEP   EXEC     PGM=IEBCOPY
//SYSPRINT DD        SYSOUT=A
//INOUT3    DD       DSNAME=DATASET3,UNIT=disk,VOL=SER=111114,
// DISP=(OLD,KEEP)
//INOUT6    DD       DSNAME=DATASET6,UNIT=disk,VOL=SER=111117,
// DISP=(OLD,DELETE)
//SYSUT3    DD       UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4    DD       UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN     DD       *
COPYOPER    COPY     OUTDD=INOUT3,INDD=INOUT6
            SELECT MEMBER=((B,H),(C,J,R),A,(D,K))
/*
```

The control statements are discussed below:

- INOUT3 DD defines a partitioned data set (DATASET3). This data set contains four members (D, G, H, and J) and resides on a disk volume.

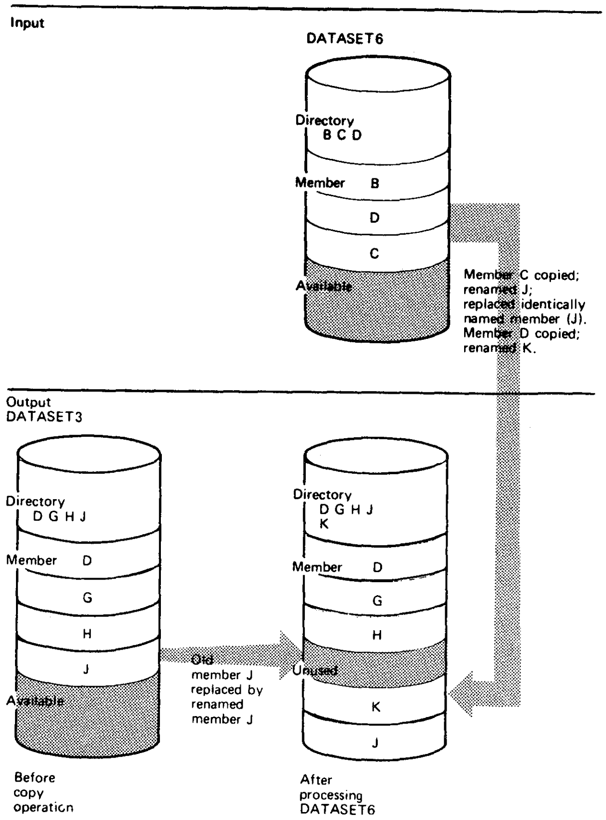


Figure 6-12. Renaming Selected Members Using IEBCOPY

- INOUT6 DD defines a partitioned data set (DATASET6). This data set contains three members (B, C, and D) and resides on a disk volume. DATASET6 is to be deleted when processing is completed; thus, all members on this data set are lost.

- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.

- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.

- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement, an INDD statement, and a SELECT statement.

- COPY indicates the start of the copy operation. The presence of a SELECT statement causes a selective copy. The OUTDD parameter specifies INOUT3 as the DD statement for the output data set (DATASET3).

- INDD specifies INOUT6 as the DD statement for the input data set (DATASET6). Processing occurs, as follows: (1) selected members are searched for on DATASET6, (2) member B is found, but is not copied to DATASET3 because its intended new name (H) is identical to the name of a member (H), which already exists on the output data set, and replace is not specified, (3) member C is found and copied to the output data set (DATASET3), although its new name (J) is identical to the name of a member (J), which already exists on the output data set, because the replace option is specified for the renamed member, and (4) member D is copied onto the output data set (DATASET3) because its new name (K) does not already exist there.

- SELECT specifies the members to be selected from the input data set (DATASET6) to be copied to the output data set (DATASET3).

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, it is suggested that the SYSUT3 and SYSUT4 DD statements always appear in the job stream.

IEBCOPY Example 8

In this example, five members (A, B, C, J, and L) are to be excluded from the copy operation when each of the input partitioned data sets (DATASET1, DATASET3, and DATASET6) is processed. In addition, replace is specified for the last input partitioned data set (DATASET6) to be processed; thus, with the exception of the members specified on the EXCLUDE statement, all members on DATASET6 will replace any identically named members on the output partitioned data set (DATASET4). Figure 6-13 shows the input and output data sets before and after processing.

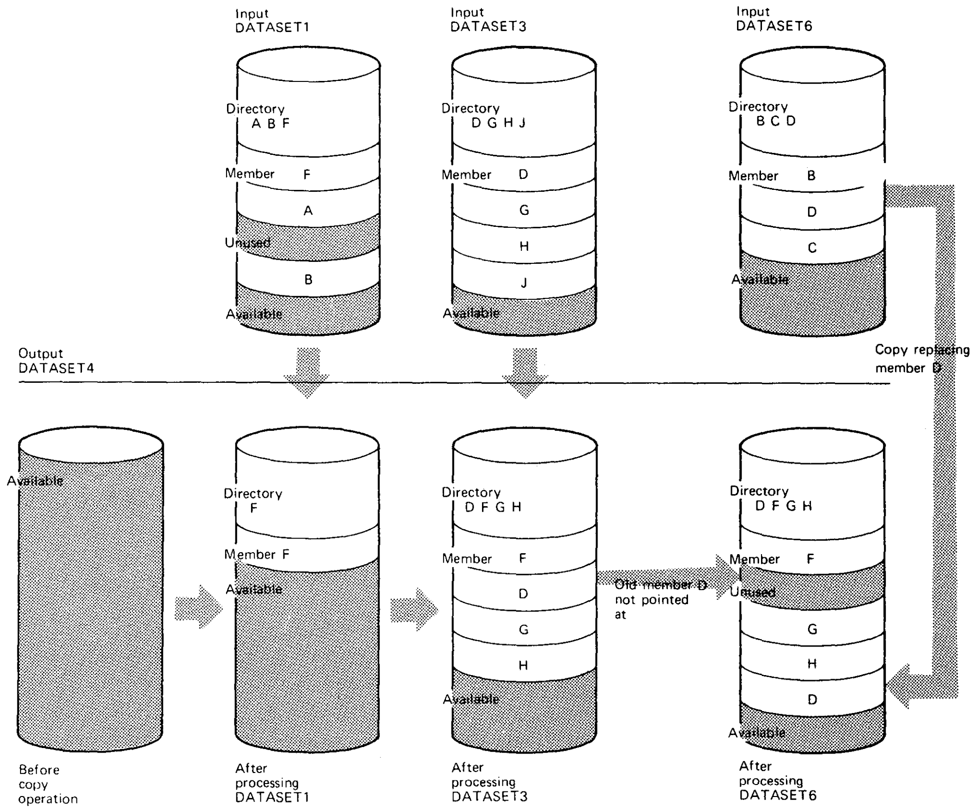


Figure 6-13. Exclusive Copy with "Replace" Specified for One Input Partitioned Data Set

```
//COPY      JOB      06#990,MCEWAN
//JOBSTEP   EXEC     PGM=IEBCOPY
//SYSPRINT  DD       SYSOUT=A
//INOUT1    DD       DSNAME=DATASET1,UNIT=disk,VOL=SER=111112,
// DISP=(OLD,KEEP)
//INOUT3    DD       DSNAME=DATASET3,UNIT=disk,VOL=SER=111114,
// DISP=OLD
//INOUT4    DD       DSNAME=DATASET4,UNIT=disk,VOL=SER=111115,
// DISP=(NEW,KEEP),SPACE=(TRK,(3,1,2)),DCB=(LRECL=100,
// RECFM=FB,BLKSIZE=400)
//INOUT6    DD       DSNAME=DATASET6,UNIT=disk,VOL=SER=111116,
// DISP=OLD
//SYSUT3    DD       UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4    DD       UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN     DD       *
COPYOPER COPY      OUTDD=INOUT4,INDD=INOUT1,INOUT3,(INOUT6,R)
         EXCLUDE   MEMBER=A,J,B,L,C
/*
```

The control statements are discussed below:

- INOUT1 DD defines a partitioned data set (DATASET1). This data set contains three members (A, B, and F) and resides on a disk volume. The record format is fixed blocked with a logical record length of 100 bytes and a block size of 400 bytes.
- INOUT3 DD defines a partitioned data set (DATASET3), which resides on a disk volume. This data set contains four members (D, G, H, and J) in fixed blocked format with a logical record length of 100 bytes and a block size of 600 bytes.
- INOUT4 DD defines a new partitioned data set (DATASET4). Five tracks are allocated for the copied members on a disk volume. Two blocks are allocated for directory entries. In addition records are to be copied to this data set in fixed blocked format with a logical record length of 100 bytes and a block size of 400 bytes.
- INOUT6 DD defines a partitioned data set (DATASET6). This data set contains three members (B, C, and D) in fixed format. The records have a logical record length of 100 bytes and a block size of 100 bytes. This data set resides on a disk volume.
- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement and an EXCLUDE statement.
- COPY indicates the start of the copy operation. The presence of an EXCLUDE statement causes an exclusive copy. The OUTDD parameter specifies INOUT4 as the DD statement for the output data set (DATASET4). The INDD parameter specifies INOUT1 as the DD statement for the first input data set (DATASET1) to be processed, INOUT3 as the DD statement for the second input data set (DATASET3) to be processed, and INOUT6 as the DD statement for the last input data set (DATASET6) to be processed. Processing occurs, as follows: (1) member F, which is not named on the EXCLUDE statement, is copied from DATASET1, (2) members D, G, and H, which are not named on the EXCLUDE statement, are copied from DATASET3, and (3) member D is copied from DATASET6 because the replace option is specified for nonexcluded members. The pointer in the output data set directory is changed to point at the

new (copied) member D; thus, the space occupied by the *old* member D (copied from DATASET3) is unused.

- EXCLUDE specifies the members to be excluded from the copy operation. The named members are excluded from all of the input partitioned data sets specified in the copy operation.

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, it is suggested that the SYSUT3 and SYSUT4 DD statements always appear in the job stream.

IEBCOPY Example 9

In this example, a partitioned data set is to be unloaded to a tape volume to create a backup copy of the data set. If this step is successful, the partitioned data set is to be compressed in place.

```
//SAVE       JOB  123456,'name',MSGLEVEL=(1,1)
//STEP1      EXEC PGM=IEBCOPY
//SYSPRINT   DD   SYSOUT=A
//INPDS      DD   DSNAME=PARTPDS,UNIT=disk,VOL=SER=PCP001,
// DISP=OLD
//BACKUP     DD   DSNAME=SAVDATA,UNIT=tape,VOL=SER=TAPE03,
// DISP=(NEW,KEEP),LABEL=(,SL)
//SYSUT3     DD   DSNAME=TEMP1,UNIT=disk,VOL=SER=111111,
// DISP=(NEW,DELETE),SPACE=(80,(60,45))
//SYSIN      DD   *
             COPY OUTDD=BACKUP,INDD=INPDS
/*
//STEP2      EXEC PGM=IEBCOPY,COND=(0,NE),
// PARM='SIZE=99999999K'
//SYSPRINT   DD   SYSOUT=A
//COMPDS     DD   DSNAME=PARTPDS,UNIT=disk,DISP=OLD,
// VOL=SER=PCP001
//SYSUT3     DD   DSNAME=TEMPA,UNIT=disk,VOL=SER=111111,
// DISP=(NEW,DELETE),SPACE=(80,(60,45))
//SYSUT4     DD   DSNAME=TEMPB,UNIT=disk,VOL=SER=111111,
// SPACE=(256,(15,1)),DCB=KEYLEN=8
//SYSIN      DD   *
             COPY OUTDD=COMPDS,INDD=COMPDS
/*
```

The control statements are discussed below:

- INPDS DD defines a partitioned data set (PARTPDS) that resides on a disk volume and has 700 members. The number of members is used to calculate the space allocation on SYSUT3.
- BACKUP DD defines a sequential data set to hold PARTPDS in unloaded form. Block size information can optionally be added; this data set must be new.
- SYSUT3 DD defines the temporary spill data set.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement.
- COPY marks the beginning of the unload operation; the absence of an EXCLUDE or SELECT statement causes the entire partitioned data set (INDD=INPDS) to be unloaded to a sequential data set (OUTDD=BACKUP).
- The second EXEC statement marks the beginning of the compress-in-place operation. The SIZE parameter indicates that the buffers are to be as large as possible. The COND parameter indicates that the compress-in-place is to be performed only if the unload operation was successful.

- COMPDS DD defines a partitioned data set (PARTPDS) that contains 700 members and resides on a disk volume.
- SYSUT3 DD defines the temporary spill data set to be used if there is not enough space in main storage for the input data set's directory entries.
- SYSUT4 DD defines the temporary spill data set to be used if there is not enough space in main storage for the output partitioned data set's directory blocks.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement.
- COPY marks the beginning of the copy operation. The absence of a SELECT or EXCLUDE statement causes a default to a full copy. Because the same DD statement is specified for both the INDD and OUTDD operands, the data set is compressed in place.

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, it is suggested that the SYSUT3 and SYSUT4 DD statements always appear in the job stream. Note, however, that the SYSUT4 data set is never used for an unload operation.

Note: For an unload operation, only one INDD data set may be specified for one OUTDD data set.

IEBCOPY Example 10

In this example, two input partitioned data sets (DATASET5 and DATASET6) are to be copied to an existing output partitioned data set (DATASET1). In addition, all members on DATASET6 are to be copied; members on the output data set that have the same names as the copied members are replaced. After DATASET6 is processed, the output data set (DATASET1) is to be compressed in place. Figure 6-14 shows the input and output data sets before and after processing.

```
//COPY     JOB   06#990,MCEWAN
//JOBSTEP  EXEC  PGM=IEBCOPY
//SYSPRINT DD    SYSOUT=A
//INOUT1   DD    DSNAME=DATASET1,UNIT=3330,VOL=SER=111112,
// DISP=(OLD,KEEP)
//INOUT5   DD    DSNAME=DATASET5,UNIT=3350,VOL=SER=111114,
// DISP=OLD
//INOUT6   DD    DSNAME=DATASET6,UNIT=3350,VOL=SER=111115,
// DISP=(OLD,KEEP)
//SYSUT3   DD    UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4   DD    UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN    DD    *
COPYOPER   COPY  OUTDD=INOUT1
                 INDD=INOUT5,(INOUT6,R),INOUT1
/*
```

The control statements are discussed below:

- INOUT1 DD defines a partitioned data set (DATASET1). This data set contains three members (A, B, and F) and resides on a 3330 volume.
- INOUT5 DD defines a partitioned data set (DATASET5). This data set contains two members (A and C) and resides on a 3350 volume.
- INOUT6 DD defines a partitioned data set (DATASET6). This data set contains three members (B, C, and D) and resides on a 3350 volume.
- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.

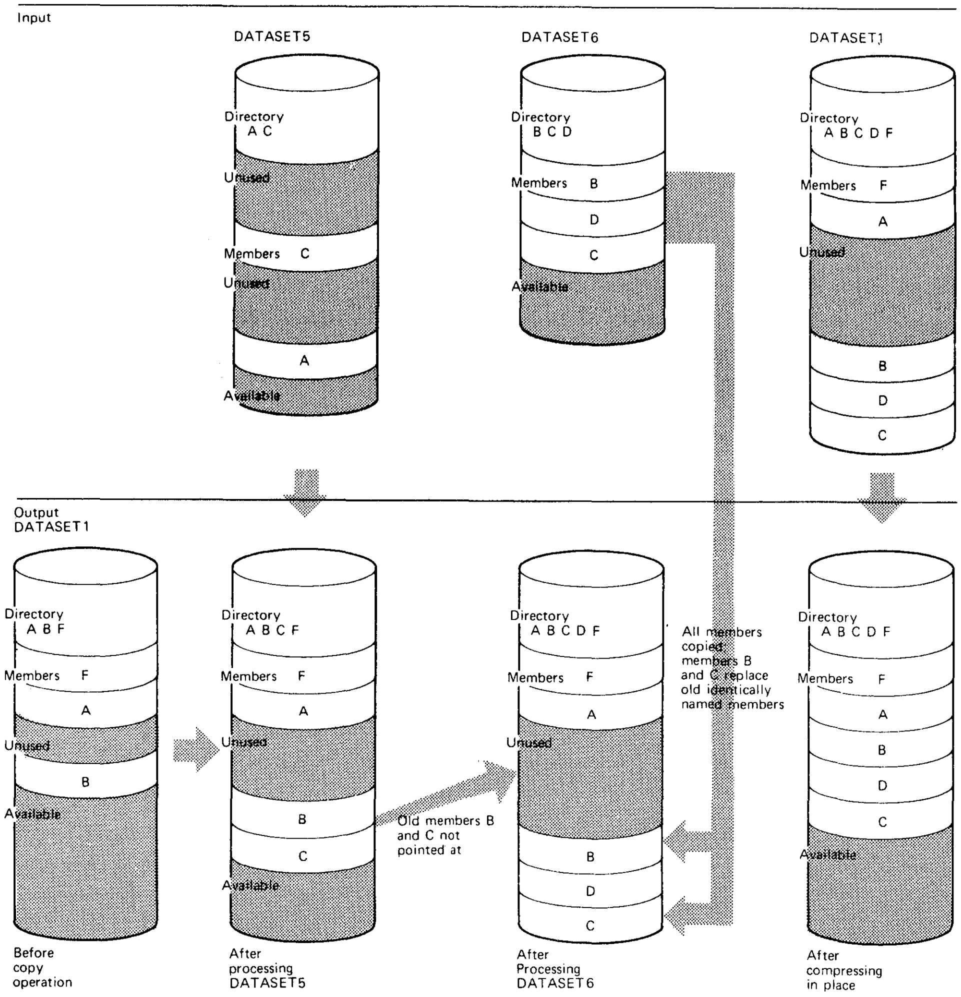


Figure 6-14. Compress-in-Place Following Full Copy with "Replace" Specified

- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement and an INDD statement.
- COPY indicates the start of the copy operation. The OUTDD operand specifies INOUT1 as the DD statement for the output data set (DATASET1). The absence of a SELECT or EXCLUDE statement causes a default to a full copy.
- INDD specifies INOUT5 as the DD statement for the first input data set (DATASET5) to be processed. It then specifies INOUT6 as the DD statement for the second input data set (DATASET6) to be processed; in addition, the replace option is specified for all members copied from DATASET6. Finally, it specifies INOUT1 as the DD statement for the last input data set (DATASET1) to be processed; this causes a compress-in-place of DATASET1 because it is also specified as the output data set. Processing occurs, as follows: (1) member A is not copied from DATASET5 onto the output data set (DATASET1) because it already exists on DATASET1 and the replace option was not specified for DATASET5, (2) member C is copied from DATASET5 to the output data set (DATASET1), occupying the first available space, and (3) all

members are copied from DATASET6 to the output data set (DATASET1), immediately following the last member. Members B and C are copied even though the output data set already contains members with the same names because the replace option is specified on the data set level. The pointers in the output data set directory are changed to point to the new members B and C; thus, the space occupied by the old members B and C is unused. The members currently on DATASET1 are compressed in place, thereby eliminating embedded unused space.

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, it is suggested that the SYSUT3 and SYSUT4 DD statements always appear in the job stream.

IEBCOPY Example 11

In this example, members are to be selected, excluded, and copied from input partitioned data sets onto an output partitioned data set. This example is designed to illustrate multiple copy operations. Figure 6-15 shows the input and output data sets before and after processing.

```
//COPY      JOB       06#990,MCEWAN
//JOBSTEP  EXEC       PGM=IEBCOPY
//SYSPRINT DD         SYSOUT=A
//INOUTA    DD        DSNAME=DATASETA,UNIT=disk,VOL=SER=111113,
// DISP=OLD
//INOUTB    DD        DSNAME=DATASETB,UNIT=disk,VOL=SER=111115,
// DISP=(OLD,KEEP)
//INOUTC    DD        DSNAME=DATASETC,UNIT=disk,VOL=SER=111114,
// DISP=(OLD,KEEP)
//INOUTD    DD        DSNAME=DATASETD,UNIT=disk,VOL=SER=111116,
// DISP=OLD
//INOUTE    DD        DSNAME=DATASETE,UNIT=disk,VOL=SER=111117,
// DISP=OLD
//INOUTX    DD        DSNAME=DATASETX,UNIT=disk,VOL=SER=111112,
// DISP=(NEW,KEEP),SPACE=(TRK,(3,1,2))
//SYSUT3    DD        UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4    DD        UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN     DD        *
COPERST1    COPY      O=INOUTX,I=INOUTA
            COPY      OUTDD=INOUTA,INDD=INOUTA
                      INDD=INOUTB
            COPY      O=INOUTA
                      INDD=INOUTD
            EXCLUDE   MEMBER=MM
                      INDD=INOUTC
            SELECT    MEMBER=((ML,MD,R))
                      INDD=INOUTE
/*
```

The control statements are discussed below:

- INOUTA DD defines a partitioned data (DATASETA). This data set contains eight members (MA, MB, MC, MD, ME, MF, and MG) and resides on a disk volume.
- INOUTB DD defines a partitioned data set (DATASETB). This data set resides on a disk volume and contains two members (MA and MJ).
- INOUTC DD defines a partitioned data set (DATASETC), which resides on a disk volume. The data set contains four members (MF, ML, MM, and MN).

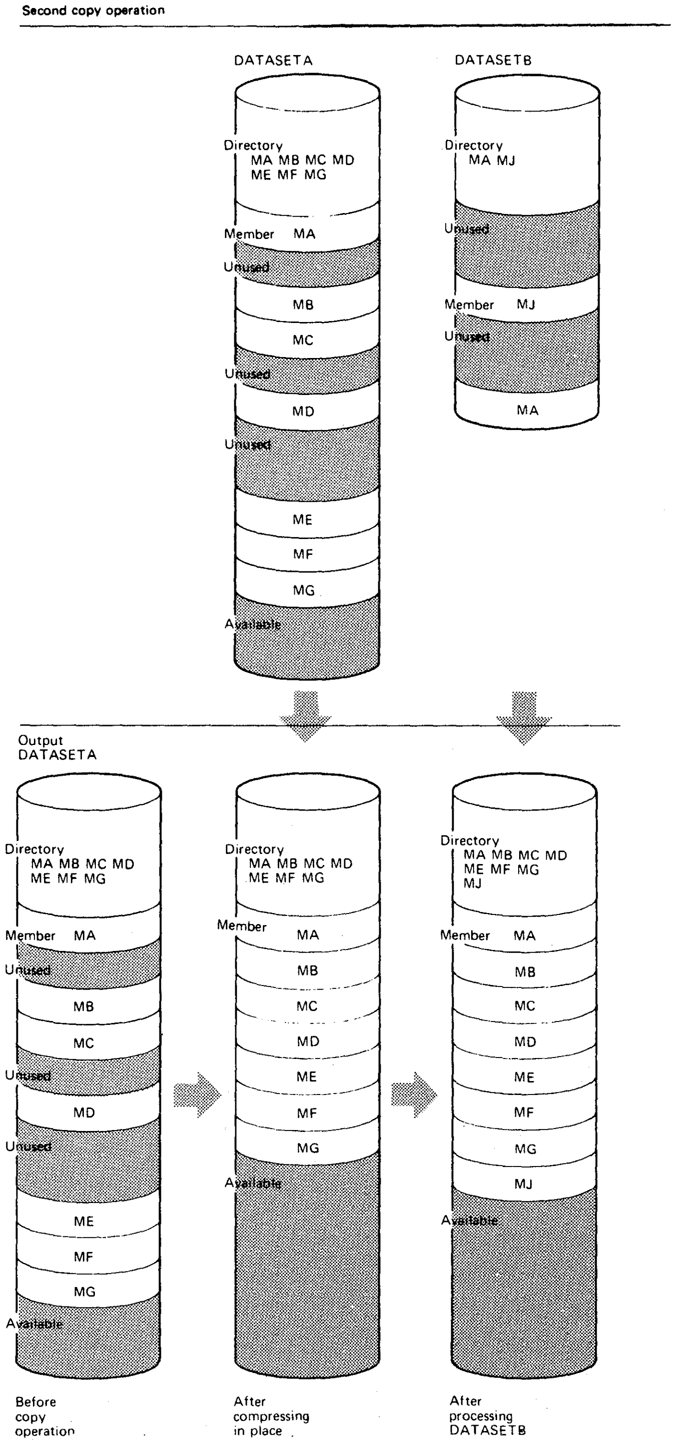


Figure 6-15 (Part 1 of 2). Multiple Copy Operations/Copy Steps

Third copy operation

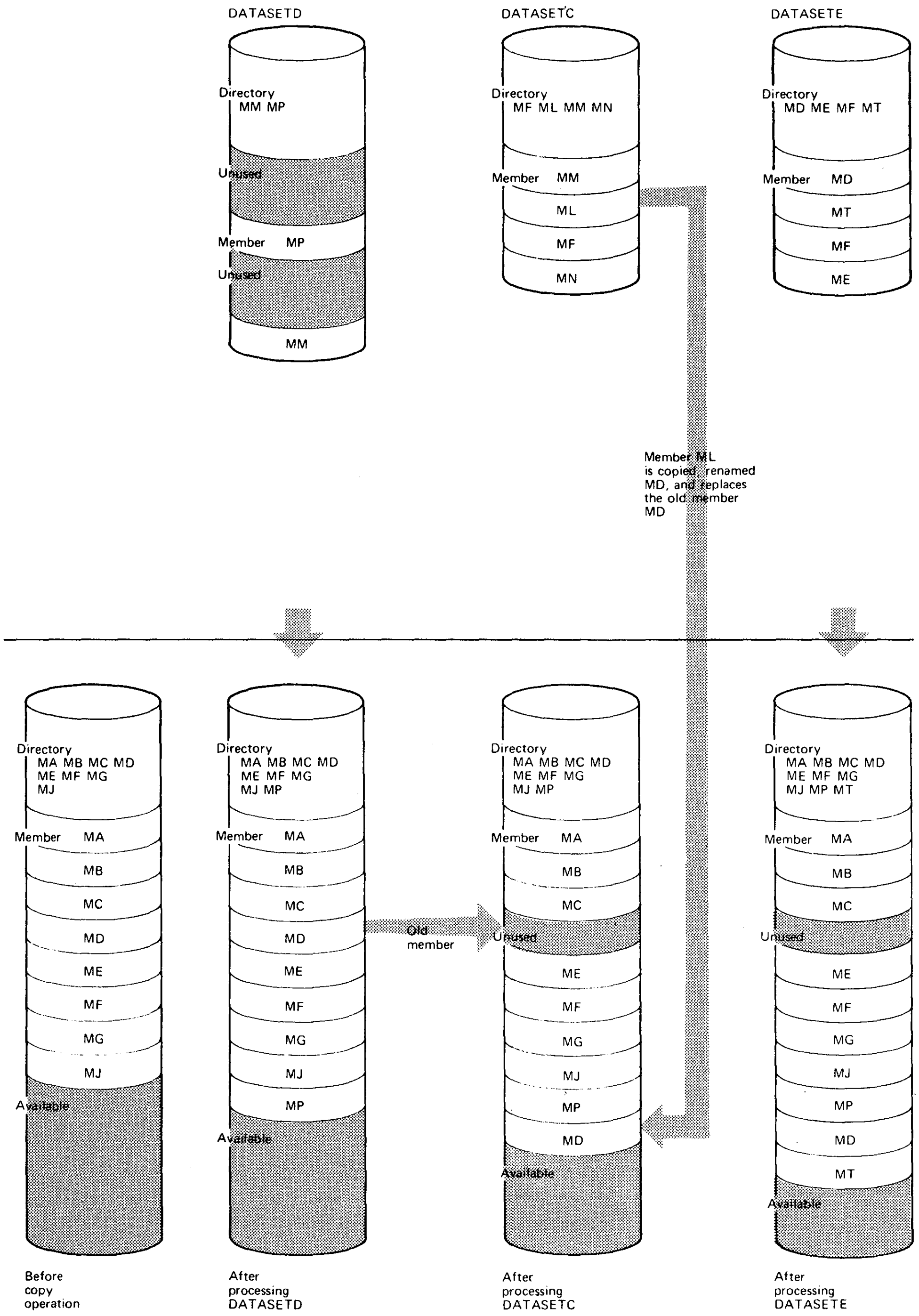


Figure 6-15 (Part 2 of 2). Multiple Copy Operations/Copy Steps

- INOUTD DD defines a partitioned data set (DATASETD). This data set resides on a disk volume and contains two members (MM and MP).
- INOUTE DD defines a partitioned data set (DATASETE). This data set contains four members (MD, ME, MF, and MT) and resides on a disk volume.
- INOUTX DD defines a partitioned data set (DATASETX). This data set is new and is to be kept after the copy operation. Five tracks are allocated for the data set on a disk volume. Two blocks are allocated for directory entries.
- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains two COPY statements, several INDD statements, a SELECT statement, and an EXCLUDE statement.
- The first COPY statement indicates the start of the first copy operation. This copy operation is done to create a backup copy of DATASETA, which is subsequently compressed in place.
- The second COPY statement indicates the start of another copy operation. The absence of a SELECT or EXCLUDE statement causes a default to a full copy; however, the same DD statement, INOUTA, is specified for both the INDD and OUTDD parameters, causing a compress-in-place of the specified data set.
- INDD specifies INOUTB as the DD statement for the input data set (DATASETB) to be copied. Only member MJ is copied because member MA already exists on the output data set.
- The third COPY statement indicates the start of the third copy operation. The OUTDD parameter specifies INOUTA as the DD statement for the output data set (DATASETA). This copy operation contains more than one copy step.
- The first INDD statement specifies INOUTD as the DD statement for the first input data set (DATASETD) to be processed. Only member MP is copied to the output data set (DATASETA) because member MM is specified on the EXCLUDE statement.
- EXCLUDE specifies the member to be excluded from the first copy step within this copy operation.
- The second INDD statement marks the beginning of the second copy step for this copy operation and specifies INOUTC as the DD statement for the second input data set (DATASETC) to be processed. Member ML is searched for, found, and copied to the output data set (DATASETA). Member ML is copied even though its new name (MD) is identical to the name of a member (MD) that already exists on the output data set, because the replace option is specified for the renamed member.
- SELECT specifies the member to be selected from the input data set (DATASETC) to be copied to the output partitioned data set.
- The third INDD statement marks the beginning of the third copy step for this copy operation and specifies INOUTE as the DD statement for the last data set (DATASETE) to be copied. Only member MT is copied because the other members already exist on the output data set. Because the INDD statement is not followed by an EXCLUDE or SELECT statement, a full copy is performed.

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, it is suggested that the SYSUT3 and SYSUT4 DD statements always appear in the job stream.

The output data set is compressed in place first to save space because it is known that it contains embedded, unused space.

IEBCOPY Example 12

In this example, members are to be selected, excluded, and copied from input partitioned data sets to an output partitioned data set. This example is designed to illustrate multiple copy operations. Figure 6-16 shows the input and output data sets before and after processing.

First copy operation

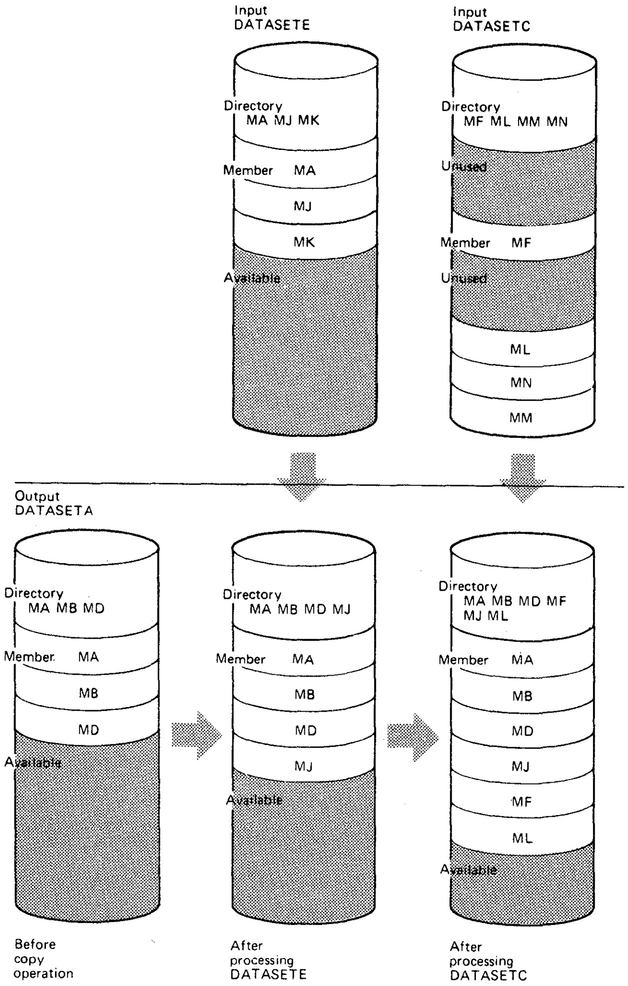


Figure 6-16 (Part 1 of 3). Multiple Copy Operations/Copy Steps Within a Job Step

Second copy operation

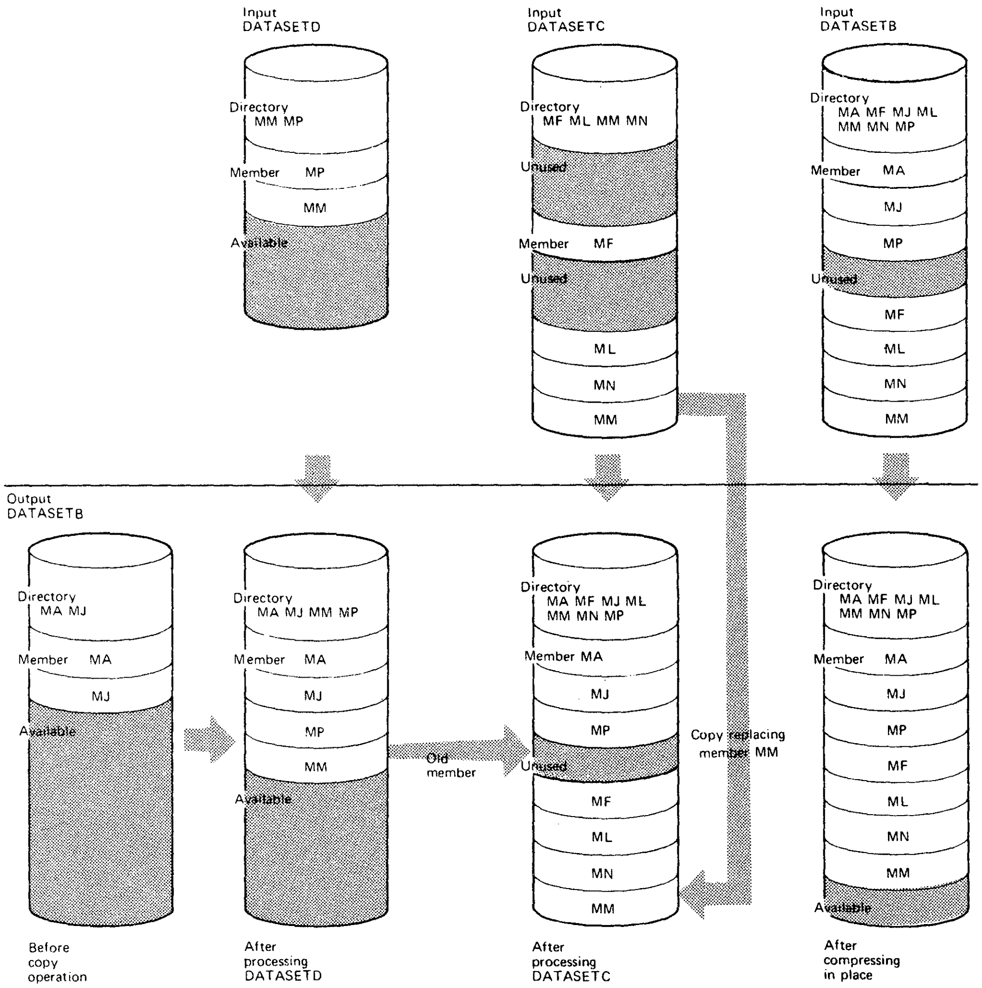


Figure 6-16 (Part 2 of 3). Multiple Copy Operations/Copy Steps Within a Job Step

Third copy operation

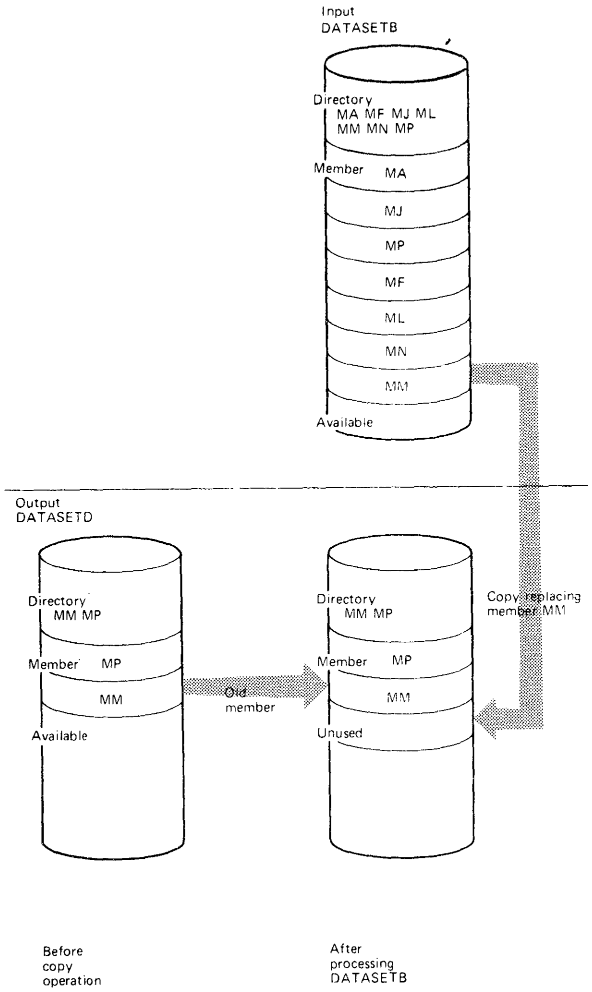


Figure 6-16 (Part 3 of 3). Multiple Copy Operations/Copy Steps Within a Job Step

```
//COPY       JOB       06#990,MCEWAN
//JOBSTEP    EXEC      PGM=IEBCOPY
//SYSPRINT   DD        SYSOUT=A
//INOUTA     DD        DSNAME=DATASETA,UNIT=disk,VOL=SER=111113,
// DISP=OLD
//INOUTB     DD        DSNAME=DATASETB,VOL=SER=111115,UNIT=disk,
// DISP=(OLD,KEEP)
//INOUTC     DD        DSNAME=DATASETC,VOL=SER=111114,UNIT=disk,
// DISP=(OLD,KEEP)
//INOUTD     DD        DSNAME=DATASETD,VOL=SER=111116,DISP=OLD,
// UNIT=disk
//INOUTE     DD        DSNAME=DATASETE,VOL=SER=111117,DISP=OLD,
//  UNIT=disk
//SYSUT3     DD        UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4     DD        UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN      DD        *
             COPY      OUTDD=INOUTA
                       INDD=INOUTE
             SELECT    MEMBER=MA,MJ
                       INDD=INOUTC
             EXCLUDE   MEMBER=MM,MN
             COPY      O=INOUTB,INDD=INOUTD
                       I=((INOUTC,R),INOUTB)
             COPY      O=INOUTD,I=((INOUTB,R))
             SELECT    MEMBER=MM
/*
```

The control statements are discussed below:

- INOUTA DD defines a partitioned data set (DATASETA). This data set contains three members (MA, MB, and MD) and resides on a disk volume.
- INOUTB DD defines a partitioned data set (DATASETB). This data set resides on a disk volume and contains two members (MA and MJ).
- INOUTC DD defines a partitioned data set (DATASETC), which resides on a disk volume. This data set contains four members (MF, ML, MM, and MN).
- INOUTD DD defines a partitioned data set (DATASETD). This data set resides on a disk volume and contains two members (MM and MP).
- INOUTE DD defines a partitioned data set (DATASETE), which resides on a disk volume. This data set contains three members (MA, MJ and MK).
- SYSUT3 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSUT4 DD defines a temporary spill data set. One track is allocated on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains three COPY statements, SELECT and EXCLUDE statements, and several INDD statements.
- The first COPY statement indicates the start of a copy operation. The OUTDD operand specifies INOUTA as the DD statement for the output data set (DATASETA).
- The first INDD statement specifies INOUTE as the DD statement for the first input data set (DATASETE) to be processed. Processing occurs, as follows: (1) member MA is searched for and found, but is not copied because the replace option is not specified, and (2) member MJ is searched for, found, and copied to the output data set. Members are not searched for again after they are found.
- SELECT specifies the members (MA and MJ) to be selected from the input data set (DATASETE) to be copied.

- The second INDD statement marks the end of the first copy step and the beginning of the second copy step within the first copy operation. It specifies INOUTC as the DD statement for the second input data set (DATASETC) to be processed. Members MF and ML, which are not named on the EXCLUDE statement, are copied because neither exists on the output data set.
- EXCLUDE specifies the members (MM and MN) to be excluded from the second copy operation.
- The second COPY statement indicates the start of another copy operation. The absence of a SELECT or EXCLUDE statement causes a default to a full copy. The O (OUTDD) parameter specifies INOUTB as the output data set (DATASETB). The INDD parameter specifies INOUTD as the first input data set (DATASETD) to be processed. Members MP and MM are copied to the output data set.
- INDD(I) specifies INOUTC as the DD statement for the second input data set (DATASETC) and INOUTB as the DD statement for the third input data set (DATASETB) to be processed. Members MF, ML, MM, and MN are copied from DATASETC. Member MM is copied, although it already exists on the output partitioned data sets, because the replace option is specified. Because DATASETB is also the data set specified in the OUTDD parameter, a compress-in-place takes place. (The pointer in the output data set directory is changed to point to the new (copied) member MM; thus the space occupied by the replaced member MM is embedded, unused space.)
- The third COPY statement indicates the start of another copy operation. The O (OUTDD) parameter specifies INOUTD as the DD statement for the output data set (DATASETD). The I (INDD) parameter specifies INOUTB as the DD statement for the input data set (DATASETB).
- SELECT specifies the member (MM) to be selected from the input partitioned data set (DATASETB) to be copied. The replace option is specified on the data set level.

The temporary spill data sets may or may not be opened, depending on the amount of virtual storage available; therefore, it is suggested that the SYSUT3 and SYSUT4 DD statements always appear in the job stream.

Data sets used as input data sets in one copy operation can be used as output data sets in another copy operation, and vice versa.

IEBCOPY Example 13

In this example, a partitioned data set (SYS1.LINKLIB) is to be unloaded to a tape volume.

```
//UNLOAD      JOB    246803,'name',MSGLEVEL=(1,1)
//STEP1       EXEC   PGM=IEBCOPY,PARM='SIZE=100K'
//SYSPRINT    DD     SYSOUT=A
//INPDS       DD     DSNAME=SYS1.LINKLIB,UNIT=disk,DISP=SHR,
// VOL=SER=666666
//OUTTAPE     DD     DSNAME=LINKLIB,UNIT=tape,VOL=SER=TAPE00,
// LABEL=(,SL),DISP=(NEW,KEEP)
//SYSUT3      DD     DSN=TEMP1,UNIT=disk,VOL=SER=111111,
// DISP=(NEW,DELETE),SPACE=(80,(60,45))
//SYSIN       DD     *
              COPY   OUTDD=OUTTAPE
                     INDD=INPDS
/*
```

The control statements are discussed below:

- EXEC specifies the execution of IEBCOPY. The PARM parameter specifies the size of the input/output buffer to be used.
- INPDS DD defines a partitioned data set (SYS1.LINKLIB), which resides on a disk volume. This data set has 700 members; the number of members is used to calculate the space allocation for SYSUT3.
- OUTTAPE DD defines a sequential data set to which SYS1.LINKLIB is to be unloaded. The unloaded data set is named LINKLIB. If a tape volume is used, it can be standard labeled or unlabeled.
- SYSUT3 DD defines a temporary spill data set on a disk volume. This data set is used if there is not enough space in virtual storage for the input partitioned data set's directory entries. This data set may or may not be opened depending on the amount of virtual storage available; therefore, it is suggested that the statement always appear in the job stream.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY and INDD statement.
- COPY indicates the start of an unload operation because the OUTDD parameter refers to OUTTAPE DD, which specifies a sequential output data set. Because of the absence of an EXCLUDE or SELECT statement, the entire data set is unloaded.
- INDD refers to INPDS DD, which defines the input partitioned data set to be unloaded. Note that for an unload operation, only one INDD data set may be specified for each OUTDD data set.

The SYSUT4 data set is never used for an unload operation. The SYSUT3 data set for an unload operation is used under the same conditions as it is used for a copy operation.

Note: If too much space is allocated, the paging process slows down because the buffer areas are fixed.

IEBCOPY Example 14

In this example, a sequential data set created by an IEBCOPY unload operation is to be loaded.

```
//LOAD      JOB  246803,'name',MSGLEVEL=(1,1)
//STEPA     EXEC PGM=IEBCOPY,PARM='SIZE=14588'
//SYSPRINT  DD   SYSOUT=A
//SEQIN     DD   DSNAME=UNLOADSET,UNIT=tape,LABEL=(,SL),
// VOL=SER=TAPE01,DISP=OLD
//INOUT4    DD   DSNAME=DATASET4,UNIT=disk,VOL=SER=2222222,
// DISP=(NEW,KEEP),SPACE=(CYL,(10,5,10))
//SYSUT3    DD   DSN=TEMP1,UNIT=disk,VOL=SER=111111,
// DISP=(NEW,DELETE),SPACE=(80,(15,1))
//SYSIN     DD   *
            COPY OUTDD=INOUT4,INDD=SEQIN
/*
```

The control statements are discussed below:

- EXEC specifies the execution of IEBCOPY. The PARM parameter allocates two tracks on a disk volume. If less space is specified, two tracks are allocated because two tracks are the minimum required by IEBCOPY when the unloaded data set's block size does not exceed the track capacity.

- SEQIN DD defines a sequential data set that was previously unloaded by IEBCOPY. The data set contains 28 members in sequential organization.
- INOUT4 DD defines a partitioned data set on a disk volume. This data set is to be kept after the load operation. Ten cylinders are allocated for the data set; ten blocks are allocated for directory entries.
- SYSUT3 DD defines a temporary spill data set on a disk volume. This data set is used if there is not enough space in main storage for the input data set's directory entries. This data set may or may not be opened, depending on the amount of main storage available; therefore, it is suggested that the statement always appear in the job stream. Note that the space allocated for this data set is based on the number of members in the input data set.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement.
- COPY indicates the start of a load operation because the INDD parameter refers to SEQIN DD, which defines a sequential data set, and OUTDD refers to INOUT4 DD, which defines a direct access volume.

Because the output data set in this example is new, the SYSUT4 data set is not needed. SYSUT4 should be specified, however, when the output data set is old.

Note: Reblocking may be specified for the output partitioned data set.

IEBCOPY Example 15

In this example, members are to be selected, excluded, unloaded, loaded, and copied. Processing will occur, as follows: (1) unload, excluding members, (2) unload, selecting members, and (3) load and copy to merge members.

```
//COPY      JOB  06#990,'name',MSGLEVEL=(1,1)
//STEP      EXEC PGM=IEBCOPY
//SYSPRINT  DD   SYSOUT=A
//PDS1      DD   DSNAME=ACCOUNTA,UNIT=3350,VOL=SER=333333,
// DISP=OLD
//PDS2      DD   DSNAME=ACCOUNTB,UNIT=3350,VOL=SER=333333,
// DISP=OLD
//SEQ1      DD   DSNAME=SAVAC,UNIT=3350,VOL=SER=333333,
// DISP=(NEW,KEEP),SPACE=(CYL,(5,2))
//SEQ2      DD   DSNAME=SAVACB,UNIT=tape,VOL=SER=T01911,
// DISP=(NEW,KEEP),LABEL=(,SL)
//NEWUP     DD   DSNAME=NEWACC,UNIT=tape,VOL=SER=T01219,
// DISP=OLD,LABEL=(,SL)
//MERGE     DD   DSNAME=ACCUPDAT,UNIT=3330-1,VOL=SER=22222222,
// DISP=OLD
//SYSUT3    DD   DSNAME=TEMP1,VOL=SER=666666,UNIT=3330-1,
// DISP=(NEW,DELETE),SPACE=(80,(1,1))
//SYSUT4    DD   DSNAME=TEMP2,VOL=SER=666666,UNIT=3330-1,
// DISP=(NEW,DELETE),SPACE=(256,(1,1)),DCB=(KEYLEN=8)
//SYSIN     DD   *
          COPY OUTDD=SEQ1,INDD=PDS1
          EXCLUDE MEMBER=(D,C)
          COPY OUTDD=SEQ2,INDD=PDS2
          SELECT MEMBER=(A,K)
          COPY OUTDD=MERGE,INDD=((NEWUP,R),PDS1,PDS2)
          EXCLUDE MEMBER=A
/*
```

The control statements are discussed below:

- PDS1 DD defines a partitioned data set that contains six members (A, B, C, D, E, and F) and resides on a 3350 volume.
- PDS2 DD defines a partitioned data set that contains three members (A, K, and L) and resides on a 3350 volume.
- SEQ1 DD defines a new sequential data set on a 3350 volume.
- SEQ2 DD defines a new sequential data set on a tape volume.
- NEWUP DD defines an old sequential data set that is the unloaded form of a partitioned data set that contains eight members (A, B, C, D, M, N, O, and P). It resides on a tape volume.
- MERGE DD defines a partitioned data set that contains six members (A, B, C, D, Q, and R) and resides on a 3330-1 volume.
- The first COPY statement indicates the start of the first unload operation. (The input data set is partitioned; the output data set is sequential.)
- The first EXCLUDE statement specifies that members D and C are to be excluded from the unload operation specified by the preceding COPY statement.
- The second COPY statement indicates the start of the second unload operation. (The input data set is partitioned; the output data set is sequential.)
- The SELECT statement specifies that members A and K are to be included in the unload operation specified by the preceding COPY statement.
- The third COPY statement indicates the start of the copy and load operations. The replace option is specified for the NEWUP data set; therefore, members in this data set replace identically named members on the output data set. The first INDD data set is an unloaded data set that is to be loaded. The second and third INDD data sets are partitioned data sets that are to be copied. (The input data sets are sequential and partitioned; the output data set is partitioned.)

IEBDG PROGRAM

IEBDG is a data set utility used to provide a *pattern* of test data to be used as a programming debugging aid.

An output data set, containing records of any format, can be created through the use of utility control statements, with or without input data. An optional user exit is provided to pass control to a user routine to monitor each output record before it is written. Sequential, indexed sequential, and partitioned data sets can be used for input or output.

The user codes utility control statements to generate a pattern of data that he can analyze quickly for predictable results.

When the user defines the contents of a field, he decides:

- What type of pattern—IBM-supplied or user-supplied—he wishes to place initially in the defined field.
- What action, if any, is to be performed to alter the contents of the field after it is selected for each output record.

IBM-Supplied Patterns

IBM supplies seven patterns: alphameric, alphabetic, zoned decimal, packed decimal, binary number, collating sequence, and random number. The user may choose one of them when he defines the contents of a field. All patterns except the binary and random number patterns repeat in a given field, provided that the defined field length is sufficient to permit repetition. For example, the alphabetic pattern is:

ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFG. . .

Figure 7-1 shows the IBM-supplied patterns.

Type	**Expressed in Hexadecimal**	**Expressed in Printable Characters**
Alphameric	C1 C2 . . .E9 F0. . .F9	AB . . . Z 0 . . . 9
Alphabetic	C1 C2 . . . E9	AB . . . Z
Zoned Decimal	F0F0 . . . F0F1	00 . . . 01
Packed Decimal	0000 . . . 001C (Positive pattern) 0000 . . . 001D (Negative pattern)	Not applicable
Binary Number	00 . . . 01 (Positive pattern) FF . . . FF (Negative pattern)	Not applicable
Collating Sequence	40 . . . F9	ƀ¢.<(+ | &!$*);¬-/,%_>?:#@'=" A . . . Z 0 . . . 9
Random Number	Random hexadecimal digits	Not applicable

Figure 7-1. IBM-Supplied Patterns

Note: A packed decimal or binary number is right aligned in the defined field.

The user can specify a starting character when defining an alphameric, alphabetic, or collating-sequence field. For example, a ten-byte alphabetic field for which "H" is specified as the starting character would appear as:

HIJKLMNOPQ

The same ten-byte alphabetic field with no specified starting character would appear as:

ABCDEFGHIJ

The user can specify a mathematical sign when defining a packed decimal or binary field. If no sign is specified, the field is assumed to be positive.

User-Specified Pictures

Instead of selecting an IBM-supplied pattern, the user can specify a picture to be placed in the defined field. The user can provide:

- An EBCDIC character string.
- A decimal number to be converted to packed decimal by IEBDG.
- A decimal number to be converted to binary by IEBDG.

When the user supplies a picture, he must specify a picture length that is equal to or less than the specified field length. An EBCDIC picture is left aligned in a defined field; a decimal number that is converted to packed decimal or to binary is right aligned in a defined field.

The user can initially load (fill) a defined field with either an EBCDIC character or a hexadecimal digit. For example, the 10-byte picture "BADCFEHGJI" is to be placed in a 15-byte field. An EBCDIC "2" is to be used to pad the field. The result is BADCFEHGJI22222. (If no fill character is provided, the remaining bytes contain binary zeros.) Remember that the fill character, if specified, is written in each byte of the defined field prior to the inclusion of an IBM-supplied pattern or user-supplied picture.

Modification of Selected Fields

IEBDG can be used to change the contents of a field in a specified manner. One of eight actions can be selected to change a field after its inclusion in each applicable output record. These actions are ripple, shift left, shift right, truncate left, truncate right, fixed, roll, and wave.

Figure 7-2 shows the effects of each of the actions on a six-byte alphabetic field. Note that the roll and wave actions are applicable only when a user pattern is supplied. In addition, the result of a ripple action depends on which type of pattern—IBM-supplied or user-supplied—is present.

If no action is selected, or if the specified action is not compatible with the format, the *fixed* action is assumed by IEBDG.

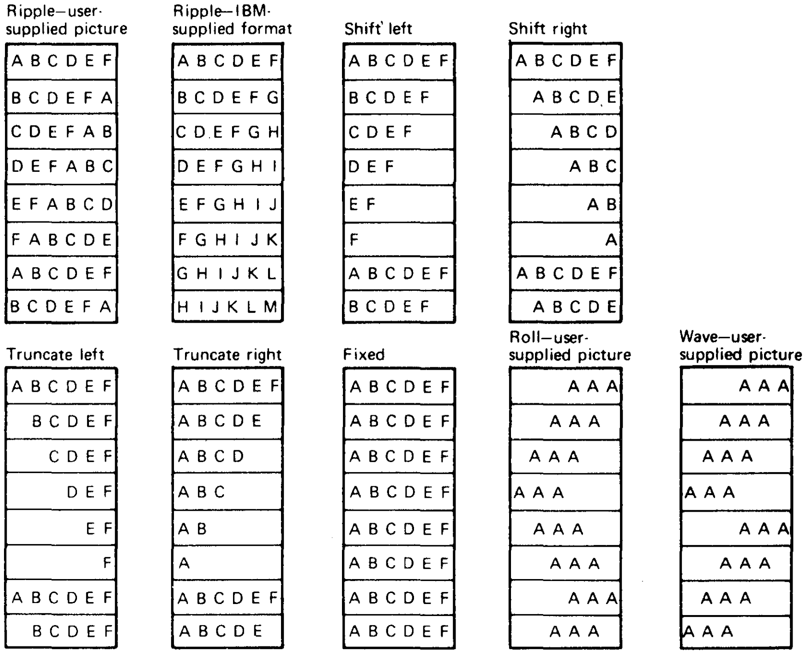


Figure 7-2. IEBDG Actions

Input and Output

IEBDG uses the following input:

- An input data set, which contains records that are to be used in the construction of an output data set or partitioned data set member. The input data sets are optional; that is, output records can be created entirely from utility control statements.
- A control data set, which contains any number of sets of utility control statements.

IEBDG produces the following output:

- An output data set, which is the result of the IEBDG operation. One output data set is created by each set of utility control statements included in the job step.
- A message data set, which contains informational messages, the contents of applicable utility control statements, and any error messages.

Note that input and output data sets may be sequential, indexed sequential, or partitioned data set members.

BDAM is not supported.

IEBDG produces a return code to indicate the results of program execution. The return codes and their meanings are:

- 00, which indicates successful completion.
- 04, which indicates that a user routine returned a code of 16 to IEBDG. The job step is terminated at the user's request.

- 08, which indicates that an error occurred while processing a set of utility control statements. No data is generated following the error. Processing continues normally with the next set of utility control statements, if any.
- 12, which indicates that an error occurred while processing an input or output data set. The job step is terminated.
- 16, which indicates that an error occurred from which recovery is not possible. The job step is terminated.

Control

IEBDG is controlled by job control statements and utility control statements. The job control statements are used to execute or invoke IEBDG and define the data sets used and produced by IEBDG. Utility control statements are used to control the functions of the program and to define the contents of the output records.

Job Control Statements

Figure 7-3 shows the job control statements necessary for using IEBDG.

Both input and output data sets can contain fixed, variable, or undefined records.

Statement	**Use**
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEBDG) or, if the job control statements reside in a procedure library, the procedure name. Additional information can be specified in the EXEC statement; see "PARM Information on the EXEC Statement" below.
SYSPRINT DD	Defines a sequential message data set. The data set can be written on a system output device, a tape volume, or a direct access volume.
SYSIN DD	Defines the control data set, which contains the utility control statements and, optionally, input records. The data set normally resides in the input stream; however, it can be defined as a sequential data set or as a member of a partitioned data set.
seqinset DD	Defines an optional sequential or indexed sequential data set used as input to IEBDG. The data set can reside on a tape volume or on a direct access volume. Any number of these statements (each having a ddname different from all other ddnames in the job step) can be included in the job step. Each DD statement is subsequently referred to by a DSD utility control statement.
parinset DD	Defines an optional input partitioned data set member residing on a direct access volume. Any number of these statements (each having a ddname different from all other ddnames in the job step) can be included in the job step. The "parinset" DD statement is referred to by a DSD utility control statement.
seqout DD	Defines an output (test) sequential or indexed sequential data set. Any number of "seqout" DD statements can be included per job step; however, only one "seqout" statement is applicable per set of utility control statements.
parout DD	Defines an optional output partitioned data set member to be created and placed on a direct access volume. Any number of "parout" DD statements (each DD statement referring to the same or to a different data set) can be included per job step; however, only one "parout" statement is applicable per set of utility control statements.

Figure 7-3. IEBDG Job Control Statements

Refer to *OS/VS2 MVS Data Management Services Guide* for information on estimating space allocations.

The "seqinset" DD statement can be entered:

```
//seqinset DD  DSNAME=setname,UNIT=xxxx,DISP=(OLD,KEEP),
//             VOLUME=SER=xxxxxx,LABEL=(...,...),
//             DCB=(applicable subparameters)
```

The LABEL parameter is included only for a magnetic tape volume. If the input data set has an indexed sequential organization, DSORG=IS should be coded in the DCB parameter.

The "parinset" DD statement can be entered:

```
//parinset DD  DSNAME=setname(membername),UNIT=xxxx,DISP=(OLD,
//             KEEP),VOLUME=SER=xxxxxx,
//             DCB=(applicable subparameters)
```

The "seqout" DD statement can be entered:

```
//seqout DD  DSNAME=setname, UNIT=xxxx,
//           DISP=(,KEEP),VOLUME=SER=xxxxxx,
//           DCB=(applicable subparameters)
```

The LABEL parameter is included for magnetic tape; the SPACE parameter is included for direct access.

The "parout" DD statement can be entered:

```
//parout DD  DSNAME=setname(membername),UNIT=xxxx,
//           DISP=(,KEEP),VOLUME=SER=xxxxxx,DCB=(applicable
//           subparameters),DISP=(,KEEP),
//           SPACE=(applicable subparameter)
```

The SPACE parameter is included on the parout DD statement when creating the first member to be placed in a partitioned data set.

PARM Information on the EXEC Statement

The EXEC statement can include an optional PARM parameter to specify the number of lines to be printed between headings in the message data set, coded as follows:

```
PARM=LINECT=nnnn
```

The nnnn is a four-digit decimal number that specifies the number of lines (0000 to 9999) to be printed per page of output listing.

If PARM is omitted, 58 lines are printed between headings (unless a channel 12 punch is encountered in the carriage control tape, in which case a skip to channel 1 is performed and a heading is printed).

Note: If IEBDG is invoked, the line-count option can be passed in a parameter list that is referred to by a subparameter of the LINK or ATTACH macro instruction. In addition, a page count can be passed in a six-byte parameter list that is referred to by a subparameter of the LINK or ATTACH macro instruction. For a discussion of linkage conventions, refer to "Appendix B: Invoking Utility Programs from a Problem Program."

Utility Control Statements

IEBDG is controlled by the following utility control statements:

Statement	Use
DSD	Specifies the ddnames of the input and output data sets. One DSD statement must be included for each set of utility control statements.
FD	Defines the contents and lengths of fields to be used in creating output records.
CREATE	Defines the contents of output records.
REPEAT	Specifies the number of times a CREATE statement or a group of CREATE statements are to be used in generating output records.
END	Marks the end of a set of IEBDG utility control statements.

Figure 7-4. IEBDG Utility Control Statements

Any number of sets of control statements can appear in a single job step. Each set defines one data set.

Note: Continuation of PICTURE parameter statements requires a nonblank character in column 72 and must begin in column 4 on the next statement.

DSD Statement

The DSD statement marks the beginning of a set of utility control statements and specifies the data sets that IEBDG is to use as input. The DSD statement can be used to specify one output data set and any number of input data sets for each application of IEBDG.

The format of the DSD statement is:

[*label*] **DSD** **OUTPUT=(***ddname***)**
[,INPUT=(*ddname***,...)]**

Note: The ddname SYSIN must not be coded in the INPUT parameter. Each parameter should appear no more than once on any DSD statement.

FD Statement

The FD statement defines the contents and length of a field that will be used subsequently by a CREATE statement (or statements) to form output records. A defined field within the input logical record may be selected for use in the output records if it is referred to, by name, by a subsequent CREATE statement.

Figure 7-5 shows how fields defined in FD statements are placed in buffer areas so that subsequent CREATE statements can assign selected fields to specific output records.

Figure 7-6 shows how the FD statement is used to specify a field in an input record to be used in output records. The left-hand side of the figure shows that a field in the input record beginning at byte 50 is selected for use in the output record. The right-hand side of the figure shows that the field is to be placed at byte 20 in the output record.

Note: When retrieving data sets with RECFM=F and RKP>0, the record consists of the key plus the data with embedded key. To copy the entire record, the output DCB=LRECL has to be input LRECL + KEYLEN. If only the data is to be copied, the FROMLOC must point to start of the data, that is, FROMLOC=keylength.

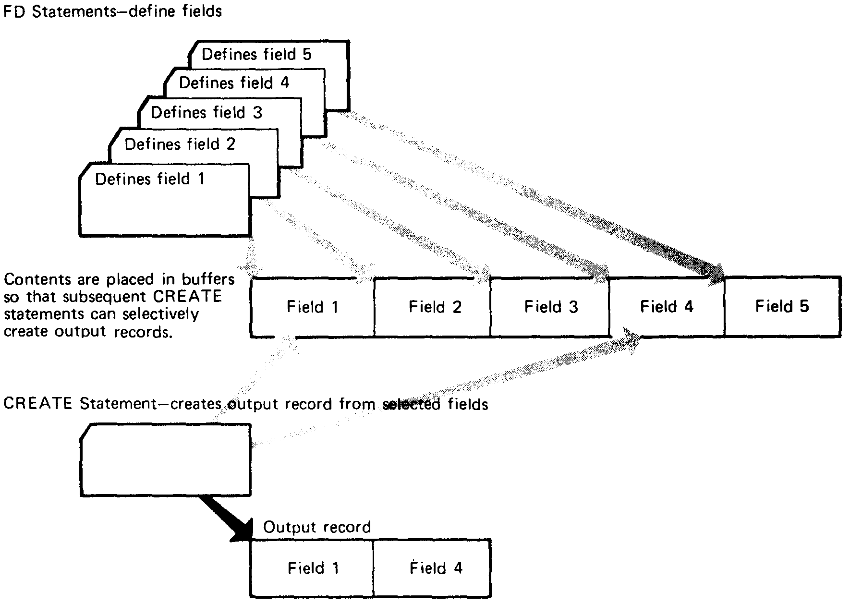


Figure 7-5. Defining and Selecting Fields for Output Records Using IEBDG

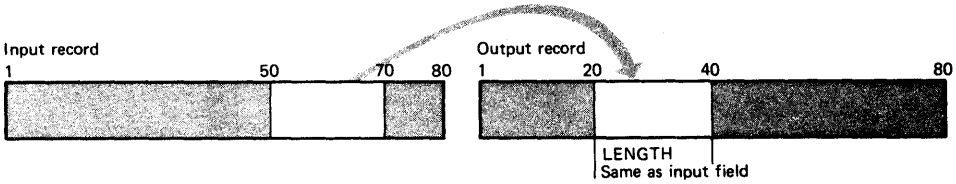


Figure 7-6. Field Selected from the Input Record for Use in the Output Record

The format of the FD statement is:

```
[label] FD  NAME=name
            ,LENGTH=length-in-bytes
            [,STARTLOC=starting-byte-location]
            [,FILL={'character' |
                    X'2-hexadecimal-digits'}]
            {[,FORMAT=pattern [,CHARACTER=character] |
            ,PICTURE=length,   {'character-string' |
                               P'decimal-number' |
                               B'decimal-number'}]}
            [,SIGN=sign]
            [,ACTION=action]
            [,INDEX=number [,CYCLE=number][,RANGE=number]]
            [,INPUT=ddname]
            [,FROMLOC=number]
```

Some of the FD keywords do not apply when certain patterns or pictures are selected by the user; for example, the INDEX, CYCLE, RANGE, and SIGN parameters are used only with numeric fields. Figure 7-7 shows which IEBDG keywords can be used with the applicable pattern or picture chosen by the user. Each keyword should appear no more than once on any FD statement.

FORMAT/PICTURE	**Compatible Operations**
Format	**Action**
AL	SL
AN	SR
CO	TL
	TR
	FX
	RP
Format	
ZD	Index
PD	Cycle
BI	Range
	Sign*
Picture	
PD	Index
BI	Cycle
	Range
	Sign
Picture	**Action**
EBCDIC	SL
	SR
	TL
	TR
	FX
	RP
	WV
	RO

***Zoned decimal numbers (ZD) do not include a sign.**

Figure 7-7. Compatible IEBDG Operations

CREATE Statement

The CREATE statement defines the contents of a record (or records) to be made available to a user routine or to be written directly as an output record (or records).

The format of the CREATE statement is:

[*label*] **CREATE** **[QUANTITY=***number* **]**
[,FILL= **{'** *character* **' | X'** *2-hexadecimal-digits* **'}]**
[,INPUT= **{***ddname* **| SYSIN[(***cccc* **)]}]**
[,PICTURE=*length* **,** *startloc* **{,'** *character-string* **' |**
,P' *decimal-number* **' |**
,B' *decimal-number* **'}]**
[,NAME= **{***name* **| (***name1* **,** *namen ...* **) |**
(*name* **(COPY=***number* **,** *name1* **,** *namen ...* **)...)}]**
[,EXIT=*routinename* **]**

After processing each potential output record, the user routine provides a return code to instruct IEBDG how to handle the output record. The user codes are:

- 00, which specifies that the record is to be written.

- 04, which specifies that the record is not to be written. The skipped record is not to be counted as a generated output record; processing is to continue as though a record were written. If skips are requested through user exits and input records are supplied, each skip causes an additional input record to be processed in the generation of output records. For example, if a CREATE statement specifies that ten output records are to be generated and a user exit indicates that two records are to be skipped, 12 input records are processed.
- 12, which specifies that the processing of the remainder of this set of utility control statements is to be bypassed. Processing is to continue with the next DSD statement.
- 16, which specifies that all processing is to halt.

Note: When an exit routine is loaded and when the user returns control to IEBDG, register one contains the address of the first byte of the output record. Each keyword should appear no more than once on any CREATE statement.

Figure 7-8 shows the addition of field X to two different records. In record 1, field X is the first field referred to by the CREATE statement; therefore, field X begins in the first byte of the output record. In record 2, two fields, field A and field B, have already been referred to by a CREATE statement; field X, the next field referred to, begins immediately after field B. Field X does not have a special starting location in this example.

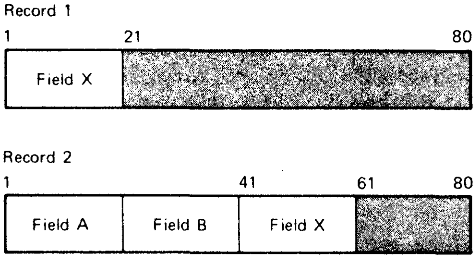


Figure 7-8. Default Placement of Fields Within an Output Record Using IEBDG

The user can also indicate that a numeric field is to be modified after it has been referred to *n* times by a CREATE statement or statements, that is, after *n* cycles, a modification is to be made. A modification will add a user-specified number to a field.

The CREATE statement constructs an output record by referring to previously defined fields by name and/or by providing a picture to be placed in the record. The user can generate multiple records with a single CREATE statement.

When defining a picture in a CREATE statement, the user must specify its length and starting location in the output record. The specified length must be equal to the number of specified EBCDIC or numeric characters. (When a specified decimal number is converted to packed decimal or binary, it is automatically right aligned.)

Figure 7-9 shows three ways in which output records can be created from utility control statements.

As an alternative to creating output records from utility control statements alone, the user can provide input records, which can be modified and written as output records. Input records can be provided directly in the input stream, or in a data set. Only one input data set can be read for each CREATE statement.

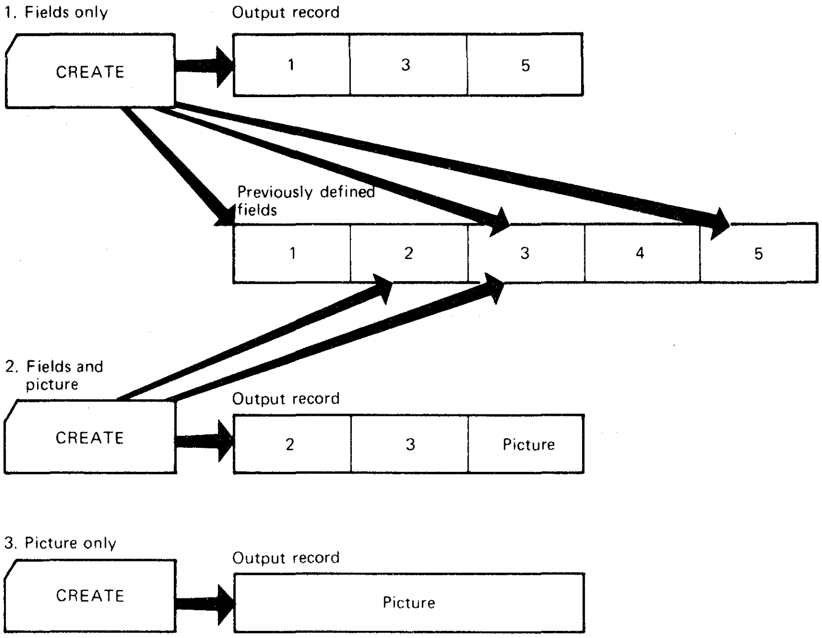


Figure 7-9. Creating Output Records with Utility Control

As previously mentioned, the CREATE statement is responsible for the construction of an output record. An output record is constructed in the following order:

1. A fill character, specified or default (binary zero), is initially loaded into each byte of the output record.
2. If the INPUT operand is specified on the CREATE statement, and not on an FD statement, the input records are left aligned in the corresponding output record.
3. If the INPUT operand specifies a *ddname* in any FD statement, only the fields described by the FD statement(s) are placed in the output record.
4. FD fields, if any, are placed in the output record in the order of the appearance of their names in the CREATE statement.
5. A CREATE statement picture, if any, is placed in the output record.

IEBDG provides a user exit so that the user can provide his own routine to analyze or further modify a newly constructed record before it is placed in the output data set.

A set of utility control statements contains one DSD statement, any number of FD, CREATE, and REPEAT statements, and one END statement when the INPUT parameter is omitted from the FD card.

When selecting fields from an input record (FD INPUT=ddname), the field must be defined by an FD statement within each set of utility control statements. In that case, defined fields for field selection are not usable across sets of utility control statements; such an FD card may be duplicated and used in more than one set of utility control statements within the job step.

REPEAT Statement

The REPEAT statement specifies the number of times a CREATE statement or group of CREATE statements is to be used repetitively in the generation of output records. The REPEAT statement precedes the CREATE statements to which it applies.

Figure 7-10 shows a group of five CREATE statements repeated *n* times.

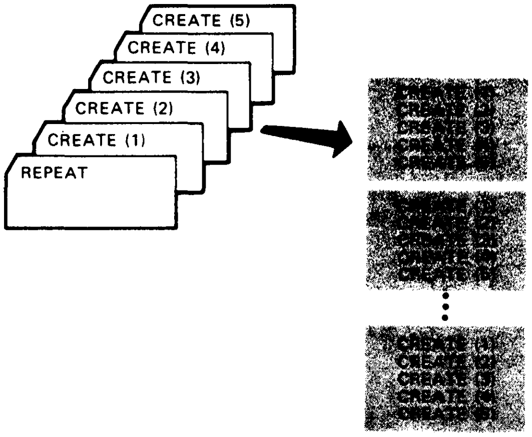


Figure 7-10. Repetition Caused by the REPEAT Statement Using IEBDG

The format of the REPEAT statement is:

[*label*] **REPEAT QUANTITY**=*number* [**,CREATE**=*number*]

END Statement

The END statement is used to mark the end of a set of utility control statements. Each set of control statements can pertain to any number of input data sets and a single output data set.

The format of the END statement is:

[*label*] **END**

Operands	Applicable Control Statements	Description of Operands/Parameters
ACTION	FD	ACTION=*action* specifies that the contents of a defined field are to be altered after the field's inclusion in an output record. These values can be coded: **SL** specifies that the contents of a defined field are to be shifted left after the field's inclusion in an output record. **SR** specifies that the contents of a defined field are to be shifted right after the field's inclusion in an output record. **TL** specifies that the contents of a defined field are to be truncated left after the field's inclusion in an output record. **TR** specifies that the contents of a defined field are to be truncated right after the field's inclusion in an output record. **RO** specifies that the contents of a defined field are to be rolled after the field inclusion in an output record. RO can be used only for a user-defined field. **WV** specifies that the contents of a defined field are to be waved after the field's inclusion in an output record. WV can be used only for a user-defined field. **FX** specifies that the contents of a defined field are to be fixed after the field's inclusion in an output record. **RP** specifies that the contents of a defined field are to be rippled after the field's inclusion in an output record. **Default:** FX
CREATE	REPEAT	CREATE=*number* specifies the number of following CREATE statements to be included in the group. **Default:** One CREATE statement is repeated.
EXIT	CREATE	EXIT=*routinename* specifies the name of a user routine that is to receive control from IEBDG before writing each output record.

Operands	Applicable Control Statements	Description of Operands/Parameters
FILL	CREATE	**FILL=**{‘*character*’ | **X‘***2-hexadecimal-digits*’} specifies a value that is to be placed in each byte of the output record before any other operation in the construction of record. These values can be coded: ‘*character*’ specifies an EBCDIC character that is to be placed in each byte of the output record. **X‘***2-hexadecimal-digits*’ specifies two hexadecimal digits (for example, FILL=X‘40’, or FILL=X‘FF’) to be placed in each byte of the output record. **Default:** Binary zeros are placed in the output record.
FORMAT	FD	**FORMAT=***pattern* [**,CHARACTER=***character*] specifies an IBM-supplied pattern that is to be placed in the defined field. FORMAT must not be used when PICTURE is used. The values that can be coded are: *pattern* specifies the IBM=supplied patterns, as follows: **AN** specifies an alphameric pattern. **ZD** specifies a zoned decimal pattern. **PD** specifies a packed decimal pattern. **CO** specifies a collating sequence pattern. **BI** specifies a binary pattern. **AL** specifies an alphabetic pattern. **RA** specifies a random binary pattern. **CHARACTER=***character* specifies the starting character of a field.
FROMLOC	FD	**FROMLOC=***number* specifies the location of the selected field within the input logical record. The number represents the position in the input record. If, for example, FROMLOC=10 is coded, the specified field begins at the tenth byte; if FROMLOC=1 is coded, the specified field begins at the first byte. (For variable records, significant data begins on the first byte after the four-byte length descriptor.) **Default:** The start of the input record.

Operands	Applicable Control Statements	Description of Operands/Parameters
INDEX	FD	**INDEX=** *number*[**,CYCLE=** *number*][**,RANGE=** *number*] specifies a number to be added to this field whenever a specified number of records have been written. These additional values can be coded: CYCLE=*number* specifies a number of output records (to be written as output or made available to an exit routine) that are treated as a group by the INDEX keyword. Whenever this field has been used in the construction of the specified number of records, it is modified as specified in the INDEX parameter. For example, if CYCLE=3 is coded, output records might appear as 111 222 333 444 etc. This parameter can be coded only when INDEX is coded. RANGE=*number* specifies an absolute value which the contents of this field can never exceed. If an index operation attempts to exceed the specified absolute value, the contents of the field as of the previous index operation are used. **Default:** No indexing is performed. If CYCLE is omitted and INDEX is coded, a CYCLE value of 1 is assumed; that is, the field is indexed after each inclusion in a potential output record.
INPUT	DSD	**INPUT=**(*ddname*,...) specifies the ddname of a DD statement defining a data set used as input to the program. Any number of data sets can be included as input—that is, any number of ddnames referring to corresponding DD statements can be coded. Whenever ddnames are included on a continuation card, they must begin in column four. **Note:** The ddname SYSIN must not be coded as the INPUT parameter on the DSD and FD control statements. Each ddname should not appear more than once on any control statement.
	FD	INPUT=*ddname* specifies the ddname of a DD-statement defining a data set used as input for field selection. Only a portion of the record described by the FD statement will be placed in the output record. If the record format of the output data set indicates variable-length records, the position within the output record will depend upon where the last insert into the output record was made. A corresponding ddname must also be specified in the associated CREATE statement in order to have the input record(s) read.

Operands	Applicable Control Statements	Description of Operands/Parameters
INPUT (continued)	CREATE	**INPUT={** *ddname* **| SYSIN[(** *cccc* **)]}** defines an input data set whose records are to be used in the construction of output records. If INPUT is coded, QUANTITY should also be coded, unless the remainder of the input records are all to be processed by this CREATE statement. If INPUT is specified in an FD statement referenced by this CREATE statement, there must be a corresponding ddname specified in the CREATE statement in order to get the input record(s) read. These values can be coded: *ddname* specifies the ddname of a DD statement defining an input data set. **SYSIN**[*cccc*] specifies that the SYSIN data set (input stream) contains records (other than utility control statements) to be used in the construction of output records. If **SYSIN** is coded, the input records follow this CREATE statement (unless the CREATE statement is in a REPEAT group, in which case the input records follow the last CREATE statement of the group). When INPUT=SYSIN with no *cccc* value is coded, the input records are delimited from any additional utility control statements by a record containing $$$E in columns 1 through 4. If this value is coded, the input records are delimited by a record containing EBCDIC characters beginning in column 1; the *cccc* can be any combination of from one to four EBCDIC characters.
LENGTH	FD	LENGTH=*length-in-bytes* specifies the length in bytes of the defined field. For variable records, four bytes of length descriptor are added.
NAME	FD	NAME=*name* specifies the name of the field defined by this FD statement.

Operands	Applicable Control Statements	Description of Operands/Parameters
NAME (continued)	CREATE	**NAME=**{ *name* | (*name1,namen* ...) | (*name,* **(COPY=** *number,name1,namen* ...)...)} specifies the name or names of previously defined fields to be included in the applicable output records. If both NAME and PICTURE are omitted, the fill character specified in the CREATE statement appears in each byte of the applicable output record. These values can be coded: (*name1* ,...) specifies the name or names of a field or fields to be included in the applicable output record(s). Each field is included in an output record in the order in which its name is encountered in the CREATE statement. **COPY=** *number* indicates that all fields named in the inner parentheses (maximum of twenty) are to be treated as a group and included the specified number of times in each output record produced by this CREATE statement. Any number of sets of inner parentheses can be included with NAME; however sets of parentheses cannot be embedded. Within each set of inner parentheses, COPY must appear before the name of any field.
OUTPUT	DSD	**OUTPUT=***(ddname)* specifies the ddname of the DD statement defining the output data set.

Operands	Applicable Control Statements	Description of Operands/Parameters
PICTURE	FD CREATE	**PICTURE=***length,startloc* {,' *character-string'* | **,P** *'decimal-number'* | **,B'** *decimal-number* '} specifies the length, starting byte (CREATE only), and the contents of a user-supplied picture. For FD, PICTURE must not be used when FORMAT is used. If both PICTURE and NAME are omitted, the fill character specified in the CREATE statement appears in each byte of applicable output records. These values can be coded: *length* specifies the number of bytes that the picture will occupy. *startloc* (CREATE only) specifies a starting byte (within any applicable output record) in which the picture is to begin. *'character-string'* specifies an EBCDIC character string that is to be placed in the applicable record(s). The character string is left aligned at the defined starting byte. A character string may be broken in column 71, a nonblank character in column 72 is required, and it must be continued in column 4 of the next statement. **P** *'decimal-number'* specifies a decimal number that is to be converted to packed decimal and right aligned (within the boundaries of the defined length and starting byte) in the output records or defined field. **B** *'decimal-number'* specifies a decimal number that is to be converted to binary and right aligned (within the boundaries of the defined length and starting byte) in the output records or defined field. In all cases for FD, the number of characters within the quotation marks must equal the number specified in the *length* subparameter.
QUANTITY	CREATE	**QUANTITY=** *number* specifies the number of records that this CREATE statement is to generate; each record is specified by the other parameters. If both QUANTITY and INPUT are coded, and the quantity specified is greater than the number of records in the input data set, the number of records created is equal to the number of input records to be processed plus the generated data up to the specified number. **Default:** If QUANTITY is omitted and INPUT is not specified, only one output record is created. If QUANTITY is omitted and INPUT is specified, the number of records created is equal to the number of records in the input data set.

Operands	Applicable Control Statements	Description of Operands/Parameters
QUANTITY (continued)	REPEAT	specifies the number of times the defined group of CREATE statements is to be used repetitively. This number cannot exceed 65,535.
SIGN	FD	**SIGN=** *sign* specifies a mathematical sign (+ or -), which is used when defining a packed decimal or binary field. **Default:** Positive (+).
STARTLOC	FD	**STARTLOC=***starting-byte-location* specifies a starting location (within all output records using this field) in which a field is to begin. For example, if the first byte of an output record is chosen as the starting location, the keyword is coded STARTLOC=1; if the tenth byte is chosen, STARTLOC =10 is coded, etc. **Default:** The field will begin in the first available byte of the output record (determined by the order of specified field names in the applicable CREATE statement). For variable records the starting location is the first byte after the length descriptor.

Restrictions

- The DSORG subparameter must be included in the DCB subparameters if the input or output data set has an indexed sequential organization (DSORG=IS). If members of a partitioned data set are used, DSORG=PO or DSORG=PS may be coded. If the DSORG subparameter is not coded, DSORG=PS is assumed.
- If the SYSPRINT DD statement is omitted, no messages are written.
- For an indexed sequential data set, the key length must be specified in the DCB.
- The block size for the SYSPRINT data set must be a multiple of 121. The block size for the SYSIN data set must be a multiple of 80. Any blocking factor can be specified for these block sizes.

IEBDG Examples

The following examples illustrate some of the uses of IEBDG. Figure 7-11 can be used as a quick reference guide to IEBDG examples. The numbers in the "Example" column point to examples that follow.

Operation	Data Set Organization	Device	Comments	Example
Place binary zeros in selected fields.	Sequential	9-track Tape	Blocked input and output.	1
Ripple alphabetic pattern	Sequential	9-track Tape, 2314 Disk	Blocked input and output.	2
Create output records from utility control statements	Sequential	2314 Disk	Blocked output.	3
Modify records from partitioned members and input stream	Partitioned, Sequential	2314 Disk	Reblocking is performed. Each block of output records contains ten modified partitioned input records and two input stream records.	4
Create partitioned members for utility control statements	Partitioned	2314 Disk	Blocked output. One set of utility control statements per member.	5
Roll and wave user-supplied patterns	Sequential	2314 Disk	Output records are created from utility control statements.	6
Create indexed sequential data set using field selection and data generation	Sequential, Indexed sequential	2314 Disk	Output records are created by augmenting selected input fields with generated data.	7

Figure 7-11. IEBDG Example Directory

Note: Examples which use *disk* or *tape*, in place of actual device-ids, must be changed before use. See the Device Support section, in the Introduction to this manual, for valid device-id notation.

IEBDG Example 1

In this example, binary zeros are to be placed in two fields of records copied from a sequential data set. After the operation, each record in the copied data set (OUTSET) contains binary zeros in locations 20 through 29 and 50 through 59.

```
                                                                       72
//CLEAROUT JOB   ,,MSGLEVEL=1
//         EXEC  PGM=IEBDG
//SYSPRINT DD    SYSOUT=A
//SEQIN    DD    DSNAME=INSET,UNIT=tape,DISP=(OLD,KEEP),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),LABEL=(,NL),
// VOLUME=SER=222222
//SEQOUT   DD    DSNAME=OUTSET,UNIT=tape,VOLUME=SER=222333,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),DISP=(,KEEP),
// LABEL=(,NL)
//SYSIN    DD    *
       DSD       OUTPUT=(SEQOUT),INPUT=(SEQIN)
       FD        NAME=FIELD1,LENGTH=10,STARTLOC=20
       FD        NAME=FIELD2,LENGTH=10,STARTLOC=50
       CREATE    QUANTITY=100,INPUT=SEQIN,NAME=(FIELD1,FIELD2)
       END
/*
```

The control statements are discussed below:

- SEQIN DD defines a sequential input data set (INSET). The data set was originally written on a unlabeled tape volume.
- SEQOUT DD defines the test data set (OUTSET). The output records are identical to the input records, except for locations 20 through 29 and 50 through 59, which contain binary zeros at the completion of the operation.
- SYSIN DD defines the control data set, which follows in the input stream.
- DSD marks the beginning of a set of utility control statements and refers to the DD statements defining the input and output data sets.
- The first FD statement defines an 80-byte field of input data.
- The first and second FD statements create two ten-byte fields (FIELD1 and FIELD2) that contain binary zeros. The fields are to begin in the 20th and 50th bytes of each output record.
- CREATE constructs 100 output rcords in which the contents of previously defined fields (FIELD1, FIELD2) are placed in their respective starting locations in each of the output records. Input records from data set INSET are used as the basis of the output records.
- END signals the end of a set of utility control statements.

IEBDG Example 2

In this example, a ten-byte alphabetic pattern is to be rippled. At the end of the job step the first output record contains "ABCDEFGHIJ", followed by data in location 11 through 80 from the input record; the second record contains "BCDEFGHIJK" followed by data in locations 11 through 80, etc.

```
                                                                      72
//RIPPLE    JOB   ,,MSGLEVEL=1
//          EXEC  PGM=IEBDG
//SYSPRINT  DD    SYSOUT=A
//SEQIN     DD    DSNAME=INSET,DISP=(OLD,KEEP),VOL=SER=222222,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),UNIT=tape
//SEQOUT    DD    DSNAME=OUTSET,UNIT=disk,VOLUME=SER=111111,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),DISP=(,KEEP),
// SPACE=(TRK,(10,10))
//SYSIN     DD    *
       DSD        OUTPUT=(SEQOUT),INPUT=(SEQIN)
       FD         NAME=FIELD1,LENGTH=10,FORMAT=AL,ACTION=RP,         C
                 STARTLOC=1
       CREATE     QUANTITY=100,INPUT=SEQIN,NAME=FIELD1
       END
/*
```

The control statements are discussed below:

- SEQIN DD defines an input sequential data set (INSET). The data set was originally written on a 9-track, standard labeled tape volume.
- SEQOUT DD defines the test output data set (OUTSET). Twenty tracks of primary space and ten tracks of secondary space are allocated for the sequential data set on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream.
- DSD marks the beginning of a set of utility control statements and refers to the DD statements defining the input and output data sets.
- The FD statement creates a ten-byte field in which the pattern ABCDEFGHIJ is placed. The data is rippled after each output record is written.
- CREATE constructs 100 output records in which the contents of a previously defined field (FIELD1) are included. The CREATE statement uses input records from data set INSET as the basis of the output records.
- END signals the end of a set of utility control statements.

IEBDG Example 3

In this example, output records are to be created entirely from utility control statements. Three fields are to be created and used in the construction of the output records. In two of the fields, alphabetic data is to be truncated; the other field is a numeric field that is to be incremented (indexed) by one after each output record is written. Figure 7-12 shows the contents of the output records at the end of the job step.

Field 1	Field 2	Field 3 (packed decimal)	
1	31	61	71 80
ABCDEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQRSTUVWXYZABCD	FF . . . FF	123 . . . 90
BCDEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQRSTUVWXYZABC	FF . . . FF	123 . . . 91
CDEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQRSTUVWXYZAB	FF . . . FF	123 . . . 92
DEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQRSTUVWXYZA	FF . . . FF	123 . . . 93
EFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQRSTUVWXYZ	FF . . . FF	123 . . . 94

Figure 7-12. Output Records at Job Step Completion

```
                                                           72
//UTLYONLY JOB  ,,MSGLEVEL=1
//         EXEC PGM=IEBDG
//SYSPRINT DD   SYSOUT=A
//SEQOUT   DD   DSNAME=OUTSET,UNIT=disk,DISP=(,KEEP),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),SPACE=(TRK,(10,10)),
// VOLUME=SER=111111
//SYSIN    DD   DATA
   DSD OUTPUT=(SEQOUT)
   FD  NAME=FIELD1,LENGTH=30,STARTLOC=1,FORMAT=AL,ACTION=TL
   FD  NAME=FIELD2,LENGTH=30,STARTLOC=31,FORMAT=AL,ACTION=TR
   FD  NAME=FIELD3,LENGTH=10,STARTLOC=71,PICTURE=10,        C
   P'1234567890',INDEX=1
   CREATE QUANTITY=100,NAME=(FIELD1,FIELD2,FIELD3),FILL=X'FF'
 END
/*
```

The control statements are discussed below:

- SEQOUT DD defines the test output data set. Ten tracks of primary space and ten tracks of secondary space are allocated for the sequential data set on a disk volume.
- SYSIN DD defines the control data set, which follows in the input stream.
- DSD marks the beginning of a set of utility control statements and refers to the DD statement defining the output data set.
- FD defines the contents of three fields to be used in the construction of output records. The first field contains 30 bytes of alphabetic data to be truncated left after each output record is written. The second field contains 30 bytes of alphabetic data to be truncated right after each output record is written. The third field is a ten-byte field containing a packed decimal number (1234567890) to be incremented by one after each record is written.
- CREATE constructs 100 output records in which the contents of previously defined fields (FIELD1, FIELD2, and FIELD3) are included.
- END signals the end of a set of utility control statements.

IEBDG Example 4

In this example, two partitioned members and input records from the input stream are to be used as the basis of a partitioned output member. Each block of 12 output records is to contain ten modified records from an input partitioned member and two records from the input stream. Figure 7-13 shows the content of the output partitioned member at the end of the job step.

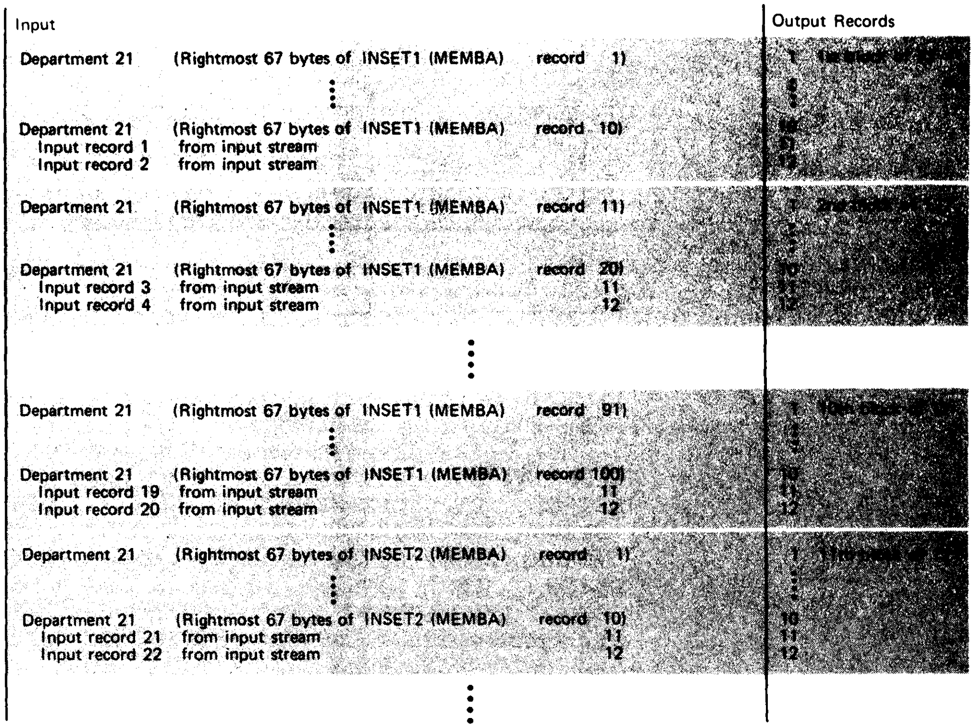


Figure 7-13. Output Partitioned Member at Job Step Completion

```
//MIX       JOB   ,,MSGLEVEL=1
//          EXEC PGM=IEBDG
//SYSPRINT DD    SYSOUT=A
//PARIN1    DD    DSNAME=INSET1(MEMBA),UNIT=disk,DISP=OLD,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=800,DSORG=PS),
// VOLUME=SER=111111
//PARIN2    DD    DSNAME=INSET2(MEMBA),UNIT=disk,DISP=OLD,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=960,DSORG=PS),
// VOLUME=SER=222222
//PAROUT    DD    DSNAME=PARSET(MEMBA),UNIT=disk,DISP=(,KEEP),
// VOLUME=SER=333333,SPACE=(TRK,(10,10,5)),DCB=(RECFM=FB,
// LRECL=80,BLKSIZE=960,DSORG=PS)
//SYSIN     DD    DATA
   DSD      OUTPUT=(PAROUT),INPUT=(PARIN1,PARIN2)
   FD       NAME=FIELD1,LENGTH=13,PICTURE=13,'DEPARTMENT 21'
   REPEAT   QUANTITY=10,CREATE=2
   CREATE   QUANTITY=10,INPUT=PARIN1,NAME=FIELD1
   CREATE   QUANTITY=2,INPUT=SYSIN
```

(input records 1 through 20)

```
$$$E
   REPEAT   QUANTITY=10,CREATE=2
   CREATE   QUANTITY=10,INPUT=PARIN2,NAME=FIELD1
   CREATE   QUANTITY=2,INPUT=SYSIN
```

(input records 21 through 40)

```
$$$E
   END
/*
```

The control statements are discussed below:

- PARIN1 DD defines one of the input partitioned members.
- PARIN 2 DD defines the second of the input partitioned members. (Note that the members are from different partitioned data sets.)
- PAROUT DD defines the output partitioned member. This example assumes that the partitioned data set does not exist prior to the job step; that is, this DD statement allocates space for the partitioned data set.
- SYSIN DD defines the control data set, which follows in the input stream.
- DSD marks the beginning of a set of utility control statements and refers to the DD statements defining the input and output data sets.
- FD creates a 13-byte field in which the picture "DEPARTMENT 21" is placed.
- The first REPEAT statement indicates that the following group of two CREATE statements is to be repeated ten times.
- The first CREATE statement creates ten output records. Each output record is constructed from an input record (from partitioned data set INSET1) and from previously defined FIELD1.
- The second CREATE statement indicates that two records are to be constructed from input records included next in the input stream.
- The $$$E record separates the input records from the REPEAT statement. The next REPEAT statement group is identical to the preceding group, except that records from a different partitioned member are used as input.
- END signals the end of a set of utility control statements.

IEBDG Example 5

In this example, output records are to be created from three sets of utility control statements and written in three partitioned data set members. Four fields are to be created and used in the construction of the output records. In two of the fields (FIELD1 and FIELD3), alphabetic data is to be shifted. The other two fields are to be fixed alphameric and zoned decimal fields. Figure 7-14 shows the partitioned data set members at the end of the job step.

MEMBA

Field 1	Field 3	Field 2	Binary zeros
1	31	51	71 80
ABCDEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQRST	00000000000000000001	fill
BCDEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQRS	00000000000000000001	fill
CDEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQR	00000000000000000001	fill
DEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQ	00000000000000000001	fill

MEMBB

Field 3	Field 3	Field 3	Field 2
1	21	41	61 80
ABCDEFGHIJKLMNOPQRST	ABCDEFGHIJKLMNOPQRST	ABCDEFGHIJKLMNOPQRST	00000000000000000001
ABCDEFGHIJKLMNOPQRS	ABCDEFGHIJKLMNOPQRS	ABCDEFGHIJKLMNOPQRS	00000000000000000001
ABCDEFGHIJLKMNOPQR	ABCDEFGHIJKLMNOPQR	ABCDEFGHIJKLMNOPQR	00000000000000000001
ABCDEFGHIJKLMNOPQ	ABCDEFGHIJKLMNOPQ	ABCDEFGHIJKLMNOPQ	00000000000000000001

MEMBC

Field 4	Field 1	Binary zeros
1	31	61 80
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123	ABCDEFGHIJKLMNOPQRSTUVWXYZABCD	fill
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123	BCDEFGHIJKLMNOPQRSTUVWXYZABCD	fill
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123	CDEFGHIJKLMNOPQRSTUVWXYZABCD	fill
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123	DEFGHIJKLMNOPQRSTUVWXYZABCD	fill

Figure 7-14. Partitioned Data Set Members at Job Step Completion

```
//UTSTS      JOB   ,,MSGLEVEL=1
//           EXEC  PGM=IEBDG
//SYSPRINT   DD    SYSOUT=A
//PAROUT1    DD    DSNAME=PARSET(MEMBA),UNIT=disk,DISP=(,KEEP),
// VOLUME=SER=111111,SPACE=(TRK,(10,10,5)),DCB=(RECFM=FB,
// LRECL=80,BLKSIZE=800,DSORG=PS)
//PAROUT2    DD    DSNAME=PARSET(MEMBB),UNIT=AFF=PAROUT1,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=800,DSORG=PS),DISP=OLD,
// VOLUME=SER=111111
//PAROUT3    DD    DSNAME=PARSET(MEMBC),UNIT=AFF=PAROUT1,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=800,DSORG=PS),DISP=OLD,
// VOLUME=SER=111111
//SYSIN      DD    DATA
       DSD         OUTPUT=(PAROUT1)
       FD          NAME=FIELD1,LENGTH=30,FORMAT=AL,ACTION=SL
       FD          NAME=FIELD2,LENGTH=20,FORMAT=ZD
       FD          NAME=FIELD3,LENGTH=20,FORMAT=AL,ACTION=SR
       FD          NAME=FIELD4,LENGTH=30,FORMAT=AN
       CREATE      QUANTITY=4,NAME=(FIELD1,FIELD3,FIELD2)
       END
       DSD         OUTPUT=(PAROUT2)
       CREATE      QUANTITY=4,NAME=((COPY=3,FIELD3),FIELD2)
       END
       DSD         OUTPUT=(PAROUT3)
       CREATE      QUANTITY=4,NAME=(FIELD4,FIELD1)
       END
/*
```

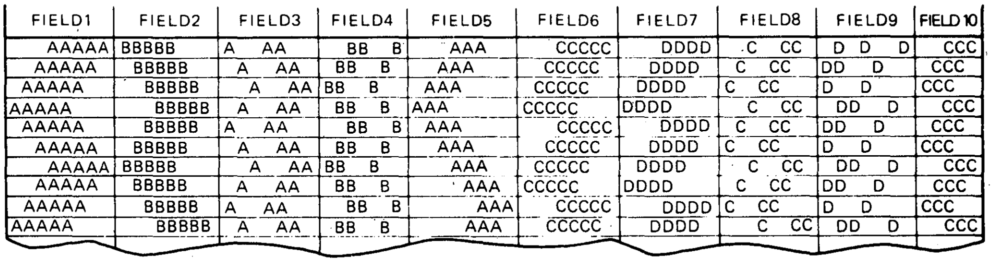
The control statements are discussed below:

- PAROUT1 DD defines the first member (MEMBA) of the partitioned output data set. This example assumes that the partitioned data set does not exist prior to this job step; that is, this DD statement allocates space for the data set.
- PAROUT2 and PAROUT3 DD define the second and third members, respectively, of the output partitioned data set. Note that each DD statement specifies DISP=OLD and UNIT=AFF=PAROUT1.
- SYSIN DD defines the control data set, which follows in the input stream.

- DSD marks the beginning of a set of utility control statements and refers to the DD statement defining the member applicable to that set of utility control statements.
- FD defines the contents of a field that is used in the subsequent construction of output records.
- CREATE constructs four records from combinations of previously defined fields.
- END signals the end of a set of utility control statements.

IEBDG Example 6

In this example, ten fields containing user-supplied EBCDIC pictures are to be used in the construction of output records. After a record is written, each field is rolled or waved, as specified in the applicable FD statement. Figure 7-15 shows the contents of the output records at the end of the job step.



FIELD1	FIELD2	FIELD3	FIELD4	FIELD5	FIELD6	FIELD7	FIELD8	FIELD9	FIELD 10
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	D D D	CCC
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	DD D	CCC
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	D D	CCC
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	DD D	CCC
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	DD D	CCC
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	D D	CCC
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	DD D	CCC
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	DD D	CCC
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	D D	CCC
AAAAA	BBBBB	A AA	BB B	AAA	CCCCC	DDDD	C CC	DD D	CCC

Figure 7-15. Contents of Output Records at Job Step Completion

```
                                                                      72
//ROLLWAVE JOB  ,,MSGLEVEL=1
//         EXEC PGM=IEBDG
//SYSPRINT DD   SYSOUT=A
//OUTSET   DD   DSNAME=SEQSET,UNIT=disk,DISP=(,KEEP),
// VOLUME=SER=SAMP,SPACE=(TRK,(10,10)),DCB=(RECFM=FB,
// LRECL=80,BLKSIZE=800)
//SYSIN    DD   *
 DSD     OUTPUT=(OUTSET)
 FD      NAME=FIELD1,LENGTH=8,PICTURE=8,'   AAAAA',ACTION=RO
 FD      NAME=FIELD2,LENGTH=8,PICTURE=8,'BBBBB   ',ACTION=RO
 FD      NAME=FIELD3,LENGTH=8,PICTURE=8,'A  AA   ',ACTION=RO
 FD      NAME=FIELD4,LENGTH=8,PICTURE=8,' BB    B',ACTION=RO
 FD      NAME=FIELD5,LENGTH=8,PICTURE=8,'  AAA   ',ACTION=RO
 FD      NAME=FIELD6,LENGTH=8,PICTURE=8,'   CCCCC',ACTION=WV
 FD      NAME=FIELD7,LENGTH=8,PICTURE=8,' DDDD   ',ACTION=WV
 FD      NAME=FIELD8,LENGTH=8,PICTURE=8,' C  CC  ',ACTION=WV
 FD      NAME=FIELD9,LENGTH=8,PICTURE=8,' DD    D',ACTION=WV
 FD      NAME=FIELD10,LENGTH=8,PICTURE=8,' CCC    ',ACTION=WV
 CREATE  QUANTITY=300,NAME=(FIELD1,FIELD2,FIELD3,                     C
               FIELD4,FIELD5,FIELD6,FIELD7,FIELD8,                    C
               FIELD9,FIELD10)
 END
/*
```

The control statements are discussed below:

- OUTSET DD defines the output sequential data set on a disk volume. Twenty tracks of primary space and ten tracks of secondary space are allocated to the data set.
- SYSIN DD defines the control data set, which follows in the input stream.

- DSD marks the beginning of a set of utility control statements and refers to the DD statement defining the output data set.

- FD defines a field to be used in the subsequent construction of output records. Note that the direction and frequency of the initial roll or wave depends on the location of data in the field.

- CREATE constructs 300 records from the contents of the previously defined fields.

- END signals the end of a set of utility control statements.

IEBDG Example 7

In this example, the first ten bytes of the output record contain data generated in zoned decimal format. This field serves as the key field for the output record in the output indexed sequential data set. The key field is incremented (indexed) by one for each record. The input sequential data set provides an additional 80-byte field to complete the output record.

```
                                                           72
//CREATEIS JOB  MSGLEVEL=1
//BEGIN    EXEC PGM=IEBDG
//TAPEIN   DD   DCB=(BLKSIZE=80,LRECL=80,RECFM=F),
// DISP=(OLD,KEEP),UNIT=tape,LABEL=(,SL),
// DSNAME=TAPEIT,VOL=SER=MASTER
//DISKOUT  DD   DCB=(BLKSIZE=270,LRECL=90,RECFM=FB,
// DSORG=IS,NTM=2,OPTCD=MY,RKP=0,KEYLEN=10,
// CYLOFL=1),UNIT=disk,SPACE=(CYL,1),DISP=(NEW,KEEP),
// VOL=SER=111111,DSNAME=CREATIS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
  DSD    OUTPUT=(DISKOUT),INPUT=(TAPEIN)
  FD     NAME=DATAFD,LENGTH=80,FROMLOC=1,                      C
               STARTLOC=11,INPUT=TAPEIN
  FD     NAME=KEYFD,LENGTH=10,STARTLOC=1,FORMAT=ZD,INDEX=1
  CREATE INPUT=TAPEIN,NAME=(KEYFD,DATAFD)
  END
/*
```

The control statements are discussed below:

- TAPEIN DD defines the sequential input data set.

- DISKOUT DD defines the indexed sequential output data set.

- SYSIN DD defines the control data set, which follows in the input stream.

- DSD marks the beginning of a set of utility control statements and refers to the DD statement defining the output data set.

- FD defines a field that will be used in the subsequent construction of output records. The first FD statement in this example defines and locates an 80-byte field of input data. The data is field selected from one of the input logical records and placed at start location 11 of the output logical record. The second FD statement defines and locates the ten-byte key field.

- CREATE constructs a 90-byte output record by referring to the previously defined fields.

- END signals the end of a set of utility control statements.

IEBEDIT PROGRAM

IEBEDIT is a data set utility used to create an output data set containing a selection of jobs or job steps. At a later time, the data set can be used as an input stream for job processing.

IEBEDIT creates an output job stream by editing and selectively copying a job stream provided as input. The program can copy:

- An entire job or jobs, including JOB statements and any associated JOBLIB or JOBCAT statements, and JES2 control statements.
- Selected job steps, including the JOB statement, JES2 control statements following the JOB statement, and any associated JOBLIB or JOBCAT statements.

All selected JOB statements, JES2 control statements, JOBLIB or JOBCAT statements, jobs, or job steps are placed in the output data set in the same order as they exist in the input data set. Note that a JES2 control statement or a JOBLIB or JOBCAT statement is copied only if it follows a selected JOB statement.

When IEBEDIT encounters a selected job step containing an input record having the characters "..*" in columns 1 through 3, the program automatically converts that record to a termination statement (/*ƀ statement) and places it in the output data set.

Note: A "/*nonblank" indicates a JES2 control statement.

Input and Output

IEBEDIT uses the following input:

- An input data set, which is a sequential data set consisting of a job stream. The input data set is used as source data in creating an output sequential data set.
- A control data set, which contains utility control statements that are used to specify the organization of jobs and job steps in the output data set.

IEBEDIT produces the following output:

- An output data set, which is a sequential data set consisting of a resultant job stream.
- A message data set, which is a sequential data set that contains applicable control statements, error messages, if applicable, and, optionally, the output data set.

IEBEDIT provides a return code to indicate the results of program execution. The return codes and their meanings are:

- 00, which indicates successful completion.
- 04, which indicates that an error occurred. The output data set may not be usable as a job stream. Processing continues.
- 08, which indicates that an unrecoverable error occurred while attempting to process the input, output, or control data set. The job step is terminated.

Control

IEBEDIT is controlled by job control statements and utility control statements. The job control statements are required to execute or invoke the program and to define the data sets used and produced by the program. The utility control statements are used to control the functions of the program.

Job Control Statements

Figure 8-1 shows the job control statements necessary for using IEBEDIT.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEBEDIT) or, if the job control statements reside in a procedure library, the procedure name.
SYSPRINT DD	Defines a sequential message data set. The data set can be written to a system output device, a tape volume, or a direct access volume.
SYSUT1 DD	Defines a sequential input data set on a card reader, tape volume, or direct access device.
SYSUT2 DD	Defines a sequential output data set on a card punch, printer, tape volume, or direct access device.
SYSIN DD	Defines the control data set. The data set normally is included in the input stream; however, it can be defined as a member of a procedure library or as a sequential data set existing somewhere other than in the input stream.

Figure 8-1. IEBEDIT Job Control Statements

Utility Control Statements

EDIT Statement

The EDIT statement indicates which step or steps of a specified job in the input data set are to be included in the output data set. Any number of EDIT statements can be included in an operation, thus including selected jobs in the output data set.

EDIT statements must be included in the same order as the input jobs that they represent. If no EDIT statement is present in the control data set, the entire input data set is copied.

The format of the EDIT statement is:

[*label*] **EDIT** **[START=***jobname* **]**
[,TYPE= {POSITION | INCLUDE | EXCLUDE}]
[,STEPNAME={(*name-name* **[,** *name-name* **] |** *name* **[,** *name* **],...)]}**
[,NOPRINT]

Notes:

- Any JES2 control statement or JOBLIB DD statement that follows a selected JOB statement is automatically copied to the output data set.
- JES2 control statements preceding the JOB statement are assumed to belong to the previous job.
- JES2 control statements preceding the first JOB statement are included only if a total copy is requested.
- JES2 control statements within selected job steps are included.
- JES2 control statements within a DD DATA stream are included only if a delimiter other than "/*" is coded in the DD DATA card. For a description of coding another delimiter see *OS/VS2 JCL.* If another delimiter is not coded, the first two characters of the JES2 control statement will act as a delimiter to DD DATA.

Operands	Applicable Control Statements	Description of Operands/Parameters
NOPRINT	EDIT	**NOPRINT** specifies that the message data set is not to include a listing of the output data set. **Default:** The resultant output is listed in the message data set.
START	EDIT	**START=***jobname* specifies the name of the input job to which the EDIT statement applies. (Each EDIT statement must apply to a separate job.) If START is specified without TYPE and STEPNAME, the JOB statement and all job steps for the specified job are included in the output. **Default:** If START is omitted and only one EDIT statement is provided, the first job encountered in the input data set is processed. If START is omitted from an EDIT statement other than the first statement, processing continues with the next JOB statement found in the input data set.
STEPNAME	EDIT	**STEPNAME=({** *name-name* **[,** *name-name* **] |** *name* **[,** *name* **]},...)** specifies the first job step to be placed in the output data set when coded with TYPE=POSITION. Job steps proceding this step are not copied to the output data set. When coded with TYPE=INCLUDE or TYPE=EXCLUDE, STEPNAME specifies the names of job steps that are to be included in or excluded from the operation. For example, STEPNAME=(STEPA,STEPF-STEPL,STEPZ) indicates that job steps STEPA, STEPF through STEPL, and STEPZ are to be included in or excluded from the operation. **Default:** If STEPNAME is omitted, the entire input job whose name is specified on the EDIT statement is copied. If no job name is specified, the first job encountered is processed.

Operands	Applicable Control Statements	Description of Operands/Parameters
TYPE	EDIT	**TYPE={POSITION | INCLUDE | EXCLUDE}** specifies the contents of the output data set. These values can be coded: **POSITION** specifies that the output is to consist of a JOB statement, the job step specified in the STEPNAME parameter, and all steps that follow it. All job steps preceding the specified step are omitted from the operation. **INCLUDE** specifies that the output data set is to contain a JOB statement and all job steps specified in the STEPNAME parameter. **EXCLUDE** specifies that the output data set is to contain a JOB statement and all job steps belonging to the job except those steps specified in the STEPNAME parameter.

Restrictions

The block size for the SYSPRINT data set must be a multiple of 121. If not, the job step is terminated with a return code of 8. The block size for the SYSIN, SYSUT1, and SYSUT2 data sets must be a multiple of 80. Any blocking factor can be specified for these block sizes.

IEBEDIT Examples

The following examples show some of the uses of IEBEDIT. Figure 8-2 can be used as a quick reference guide to IEBEDIT examples. The numbers in the "Example" column point to examples that follow.

Operation	Devices	Comments	Example
COPY	9-track Tape	The input data set contains three jobs. One job is to be copied.	1
COPY	7-track Tape	The output data set is the second data set on the volume. One job step is to be copied from each of three jobs.	2
COPY	Disk and 9-track Tape	Include a job step from one job and exclude a job step from another job.	3
COPY	Disk	Latter portion of a job stream is to be copied.	4
COPY	9-track Tape	All records in the input data set are to be copied. The "..*" record is converted to a "/*ƀ" statement in the output data set.	5
COPY	9-track Tape	The input contains a JES2 control statement and a new delimiter.	6

Figure 8-2. IEBEDIT Example Directory

Note: Examples which use *disk* or *tape*, in place of actual device-ids, must be changed before use. See the Device Support section, in the Introduction to this manual, for valid device-id notation.

IEBEDIT Example 1

In this example one job (JOBA), including all of its job steps (A, B, C, and D), is to be copied into the output data set. The input data set contains three jobs: JOBA, which has four job steps; JOBB, which has three job steps; and JOBC, which has two job steps.

```
//EDIT1     JOB  09#440,SMITH
//          EXEC PGM=IEBEDIT
//SYSPRINT DD    SYSOUT=A
//SYSUT1    DD   UNIT=tape,DISP=(OLD,KEEP),VOL=SER=001234
//SYSUT2    DD   UNIT=tape,DISP=(NEW,KEEP),VOL=SER=001235,
// DCB=(RECFM=F,LRECL=80,BLKSIZE=80),DSNAME=OUTTAPE
//SYSIN     DD   *
          EDIT   START=JOBA
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set. The data set resides on a standard labeled tape volume (001234).
- SYSUT2 DD defines the output data set. The data set is to reside as the first data set on a standard labeled tape volume (001235).

- SYSIN DD defines the control data set, which follows in the input stream.
- EDIT indicates that JOBA is to be copied in its entirety.

IEBEDIT Example 2

This example copies: (1) the JOB statement and steps STEPC and STEPD for JOBA, (2) the JOB statement and STEPE for JOBB, and (3) the JOB statement and STEPJ for JOBC. The input data set contains three jobs: JOBA, which includes STEPA, STEPB, STEPC, and STEPD; JOBB, which includes STEPE, STEPF, and STEPG; and JOBC, which includes STEPH and STEPJ.

```
//EDIT2     JOB  09#440,SMITH
//          EXEC PGM=IEBEDIT
//SYSPRINT DD   SYSOUT=A
//SYSUT1    DD   DISP=(OLD,KEEP),VOLUME=SER=001234,
// UNIT=tape
//SYSUT2    DD   DSN=OUTSTRM,UNIT=tape,DISP=(NEW,KEEP),
// DCB=(RECFM=F,LRECL=80,BLKSIZE=80),
// LABEL=(2,SL)
//SYSIN     DD   *
       EDIT   START=JOBA,TYPE=INCLUDE,STEPNAME=(STEPC,STEPD)
       EDIT   START=JOBB,TYPE=INCLUDE,STEPNAME=STEPE
       EDIT   START=JOBC,TYPE=INCLUDE,STEPNAME=STEPJ
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set. The data set resides on a standard labeled tape volume (001234).
- SYSUT2 DD defines the output data set. The data set is to reside as the second data set on a standard labeled tape volume (001235).
- SYSIN DD defines the control data set, which follows in the input stream.
- The EDIT statements copy the indicated JOB statements and job steps.

IEBEDIT Example 3

This example copies: (1) the JOB statement and steps STEPF and STEPG for JOBB and (2) the JOB statement and STEPH, excluding STEPJ, for JOBC. The input data set contains three jobs: JOBA, which includes STEPA, STEPB, STEPC, and STEPD; JOBB, which includes STEPE, STEPF, and STEPG; and JOBC, which includes STEPH and STEPJ.

```
//EDIT3     JOB  09#440,SMITH
//          EXEC PGM=IEBEDIT
//SYSPRINT DD   SYSOUT=A
//SYSUT1    DD   DSNAME=INSET,UNIT=disk,DISP=(OLD,KEEP),
// VOLUME=SER=111111
//SYSUT2    DD   DSNAME=OUTTAPE,UNIT=tape,LABEL(,NL),
// DCB=(DEN=2,RECFM=F,LRECL=80,BLKSIZE=80),DISP=(,KEEP)
//SYSIN     DD   *
       EDIT   START=JOBB,TYPE=INCLUDE,STEPNAME=(STEPF-STEPG)
       EDIT   START=JOBC,TYPE=EXCLUDE,STEPNAME=STEPJ
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set. The data set resides on a disk volume (111111).
- SYSUT2 DD defines the output data set. The data set is to reside as the first or only data set on an unlabeled (800 bits per inch) tape volume.

- SYSIN DD defines the control data set, which follows in the input stream.
- The EDIT statements copy selected JOB statements and job steps.

IEBEDIT Example 4

This example copies the JOBA JOB statement, the job step STEPF, and all the steps that follow it. The input data set contains one job (JOBA), which includes STEPA, STEPB, . . . STEPL. Job steps STEPA through STEPE are not included in the output data set.

```
//EDIT4     JOB  09#440,SMITH
//          EXEC PGM=IEBEDIT
//SYSPRINT DD   SYSOUT=A
//SYSUT1    DD   DSNAME=INSTREAM,UNIT=disk,DISP=(OLD,KEEP),
// VOLUME=SER=111111
//SYSUT2    DD   DSNAME=OUTSTREM,UNIT=disk,DISP=(,KEEP),
// DCB=(RECFM=F,LRECL=80,BLKSIZE=80),VOLUME=SER=222222,
// SPACE=(TRK,2)
//SYSIN     DD   *
         EDIT   START=JOBA,TYPE=POSITION,STEPNAME=STEPF
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set. The data set resides on a disk volume (111111).
- SYSUT2 DD defines the output data set. The data set is to reside on a disk volume (222222). Two tracks are allocated for the output data set.
- SYSIN DD defines the control data set, which follows in the input stream.
- EDIT copies the JOB statement and job steps STEPF through STEPL.

IEBEDIT Example 5

This example copies the entire input (SYSUT1) data set. The record containing the characters "..*" in columns 1 through 3 is converted to a "/*ƀ" statement in the output data set.

```
//EDIT5     JOB  09#440,SMITH
//          EXEC PGM=IEBEDIT
//SYSPRINT DD   SYSOUT=A
//SYSUT2    DD   DSNAME=OUTTAPE,UNIT=tape,VOLUME=SER=001234,
// DCB=(RECFM=F,LRECL=80,BLKSIZE=80),DISP=(NEW,KEEP)
//SYSIN     DD   DUMMY
//SYSUT1    DD   DATA
//BLDGDGIX JOB
//          EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1       DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//SYSIN     DD   *
         BLDG   INDEX=A.B.C,ENTRIES=10,EMPTY
..*
/*
```

The control statements are discussed below:

- SYSUT2 DD defines the output data set. The data set is to reside as the first data set on a tape volume (001234).
- SYSIN DD defines a dummy control data set.
- SYSUT1 DD defines the input data set, which follows in the input stream. The job is terminated when the termination statement (/*ƀ) is encountered.

- SYSIN DD defines a dummy control data set.
- SYSUT1 DD defines the input data set, which follows in the input stream. The job is terminated when the termination statement (/*ƀ) is encountered.

IEBEDIT Example 6

This example copies the entire input (SYSUT1) data set including the JES2 control statement since a new delimiter (JP) has been coded. Otherwise the "/*" in the JES2 control statement would have terminated the input.

```
//EDIT6     JOB  09#440,SMITH
//STEPA     EXEC PGM=IEBEDIT
//SYSPRINT DD   SYSOUT=A
//SYSUT2    DD   DSN=TAPEOUT,UNIT=tape,VOL=SER=001234,
// LABEL=(,SL),DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
// DISP=(NEW,KEEP)
//SYSIN     DD   DUMMY
//SYSUT1    DD   DATA,DLM=JP
//LISTVTOC JOB  09#550,BLUE
/*MESSAGE   JOB NEEDS VOLUME 231400
//FSTEP     EXEC PGM=IEHLIST
//SYSPRINT DD   SYSOUT=A
//DD2       DD   UNIT=disk,VOL=SER=111111,DISP=OLD
//SYSIN     DD   *
    LISTVTOC FORMAT,VOL=disk=111111
/*
JP
```

The control statements are discussed below:

- SYSUT2 DD defines the output data set. The data set will be the first data set on a standard label tape volume (001234).
- SYSIN DD defines a dummy control data set.
- SYSUT1 DD defines the input data set. The DLM parameter defines characters JP to act as a delimiter for the input data.
- EDIT copies the JOB statement through the "/*" statement.

IEBGENER PROGRAM

IEBGENER is a data set utility that can be used to:

- Create a backup copy of a sequential data set or a partitioned member.
- Produce a partitioned data set or member from a sequential input data set.
- Expand an existing partitioned data set by creating partitioned members and merging them into the data set that is to be expanded.
- Produce an edited sequential or partitioned data set.
- Reblock or change the logical record length of a data set.
- Copy user labels on sequential output data sets. (Refer to "Appendix D: Processing User Labels.")
- Provide optional editing facilities and exits for user routines that process labels, manipulate input data, create keys, and handle permanent input/output errors. Refer to "Appendix A: Exit Routine Linkage" for a discussion of linkage conventions that are applicable when user routines are provided.

At the completion or termination of IEBGENER, the highest return code encountered within the program is passed to the calling program.

Creating a Backup Copy

A backup copy of a sequential data set or partitioned member can be produced by copying the data set or member to any IBM-supported output device. For example, a copy can be made from tape to tape, from direct access to tape, etc.

A data set that resides on a direct access volume can be copied to its own volume, provided that its data set name is changed. A partitioned data set cannot reside on a magnetic tape volume.

Producing a Partitioned Data Set from Sequential Input

Through the use of utility control statements, the user can logically divide a sequential data set into *record groups* and assign member names to the record groups. IEBGENER places the newly created members in a partitioned output data set.

Note: A partitioned data set cannot be produced if an input or output data set contains spanned records.

Figure 9-1 shows how a partitioned data set is produced from a sequential data set used as input. The left side of the figure shows the sequential data set. Utility control statements are used to divide the sequential data set into record groups and to provide a member name for each record group. The right side of the figure shows the partitioned data set produced from the sequential input.

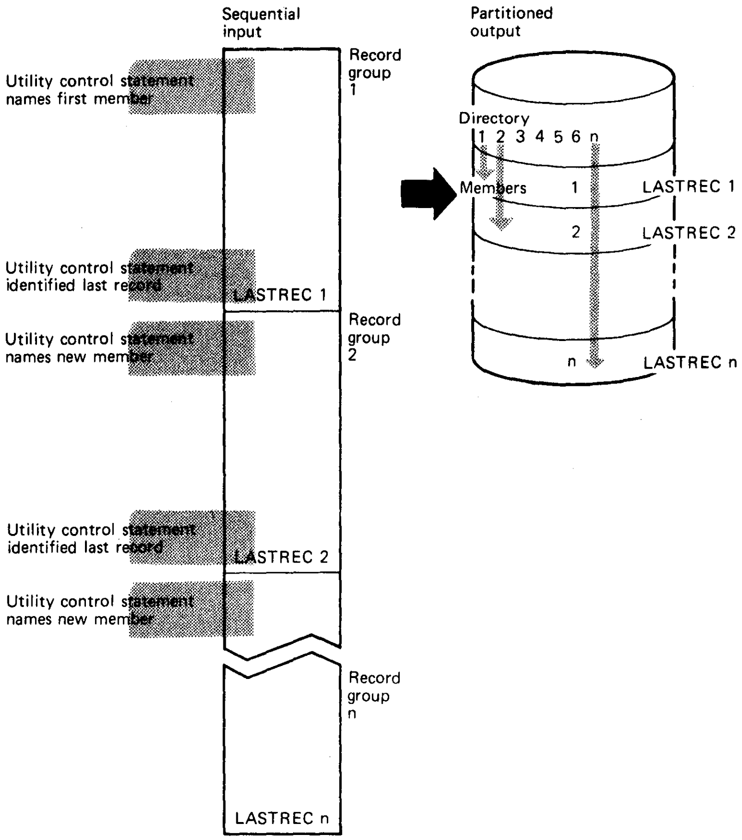


Figure 9-1. Creating a Partitioned Data Set from Sequential Input Using IEBGENER

Expanding a Partitioned Data Set

An expanded data set is a data set into which an additional member or members have been merged. IEBGENER creates the members from sequential input and places them in the data set being expanded. The merge operation—the ordering of the partitioned directory—is automatically performed by the program.

Figure 9-2 shows how sequential input is converted into members that are merged into an existing partitioned data set. The left side of the figure shows the sequential input that is to be merged with the partitioned data set shown in the middle of the figure. Utility control statements are used to divide the sequential data set into record groups and to provide a member name for each record group. The right side of the figure shows the expanded partitioned data set. Note that members B, D, and F from the sequential data set were placed in *available space* and that they are sequentially ordered in the partitioned directory.

Producing an Edited Data Set

IEBGENER can be used to produce an edited sequential or partitioned data set. Through the use of utility control statements, the user can specify editing information that applies to a record, a group of records, selected groups of records, or an entire data set.

An edited data set can be produced by:

- Rearranging or omitting defined data fields within a record.

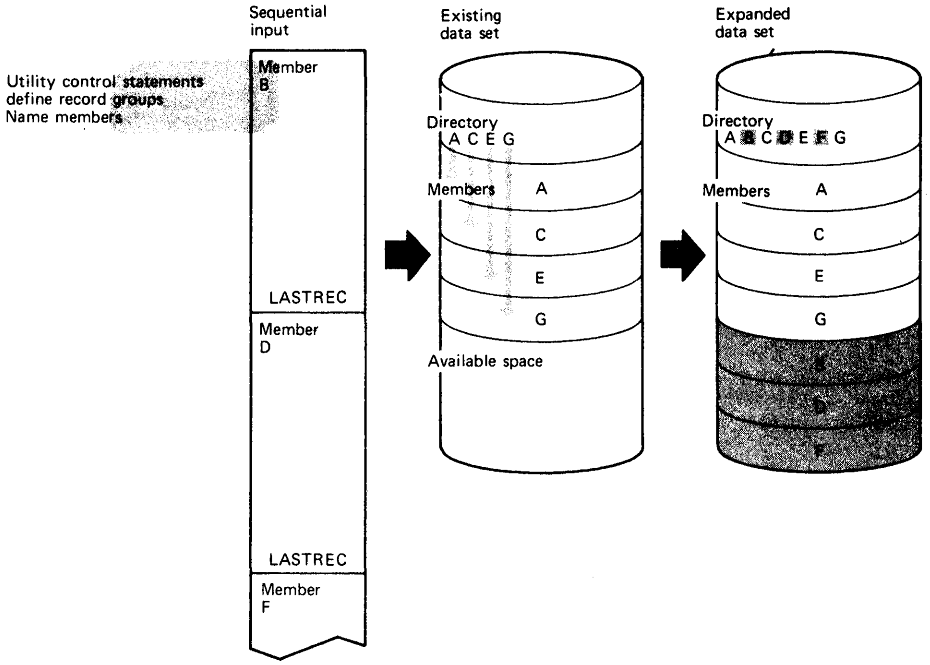


Figure 9-2. Expanding a Partitioned Data Set Using IEBGENER

- Supplying literal information as replacement data.
- Converting data from packed decimal to unpacked decimal mode, unpacked decimal to packed decimal mode, or H-set BCD to EBCDIC mode.

Figure 9-3 shows part of an edited sequential data set. The left-hand side of the figure shows the data set before editing is performed. Utility control statements are used to identify the record groups to be edited and to supply editing information. In this figure, literal replacement information is supplied for information within a defined field. (Data is rearranged, omitted, or converted in the same manner.) The BBBB field in each record in the record group is to be replaced by CCCC. The right-hand side of the figure shows the data set after editing.

Note: IEBGENER cannot be used to edit a data set if the input and output data sets consist of variable spanned (VS) or variable blocked spanned (VBS) records and have equal block sizes and logical record lengths. In this case, any utility control statements that specify editing are ignored, that is, for each physical record read from the input data set, the utility writes an unedited physical record on the output data set.

Reblocking or Changing Logical Record Length

IEBGENER can be used to produce a reblocked output data set containing either fixed or variable records. In addition, the program can produce an output data set having a logical record length that differs from the input logical record length.

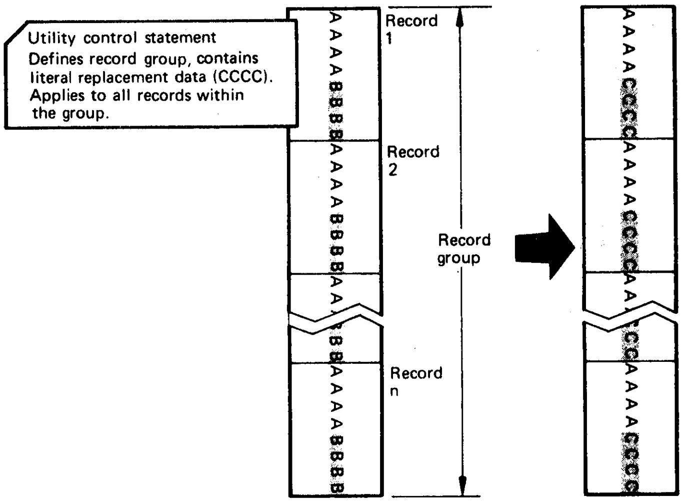


Figure 9-3. Editing a Sequential Data Set Using IEBGENER

Input and Output

IEBGENER uses the following input:

- An input data set, which contains the data that is to be copied, edited, converted into a partitioned data set, or converted into members to be merged into an existing data set. The input is either a sequential data set or a member of a partitioned data set.
- A control data set, which contains utility control statements. The control data set is required if editing is to be performed or if the output data set is to be a partitioned data set.

IEBGENER produces the following output:

- An output data set, which can be either sequential or partitioned. The output data set can be either a new data set (created during the current job step) or an existing partitioned data set that was expanded.
- A message data set, which contains informational messages (for example, the contents of utility control statements) and any error messages.
- Message IEC507D will be issued twice when adding data or members to an existing data set which has an unexpired expiration date. This occurs because the input and output data sets are opened twice.

IEBGENER provides a return code to indicate the results of program execution. The return codes and their meanings are:

- 00, which indicates successful completion.
- 04, which indicates probable successful completion. A warning message is written.
- 08, which indicates that processing was terminated after the user requested processing of user header labels only.
- 12, which indicates an unrecoverable error. The job step is terminated.

- 16, which indicates that a user routine passed a return code of 16 to IEBGENER. The job step is terminated.

Control

IEBGENER is controlled by job control statements and utility control statements. The job control statements are required to execute or invoke IEBGENER and to define the data sets that are used and produced by the program. The utility control statements are used to control the functions of IEBGENER.

Job Control Statements

Figure 9-4 shows the job control statements necessary for using IEBGENER.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEBGENER) or, if the job control statements reside in a procedure library, the procedure name.
SYSPRINT DD	Defines a sequential message data set. The data set can be written to a system output device, a tape volume, or a direct access volume.
SYSUT1 DD	Defines the input data set. It can define a sequential data set or a member of a partitioned data set.
SYSUT2 DD	Defines the output data set. It can define a sequential data set, a member of a partitioned data set, or a partitioned data set.
SYSIN DD	Defines the control data set, or specifies DUMMY when the output is sequential and no editing is specified. The control data set normally resides in the input stream; however, it can be defined as a member within a library of partitioned members.

Figure 9-4. IEBGENER Job Control Statements

IEBGENER always uses two buffers, regardless of what was specified in the DCB.

If both the SYSUT1 and the SYSUT2 DD statements specify standard user labels (SUL), IEBGENER copies user labels from SYSUT1 to SYSUT2. See "Appendix D: Processing User Labels" for a discussion of the available options for user label processing.

Both the input data set and the output data set can contain fixed, variable, undefined, or variable spanned records. These records can be reblocked by the specification of a new maximum block length on the SYSUT2 DD statement. During reblocking, if the output data set resides on a direct access volume:

- For fixed or variable records, keys can be retained only by using the appropriate user exit.
- For variable spanned records, keys can never be retained.

Refer to *OS/VS2 MVS Data Management Services Guide* for information on estimating space allocations.

Utility Control Statements

IEBGENER is controlled by utility control statements. The statements and the order in which they must appear are:

The control statements are included in the control data set as required. If no utility control statements are included in the control data set, the entire input data set is copied sequentially.

Statement	**Use**
GENERATE	Indicates the number of member names and alias names, record identifiers, literals, and editing information contained in the control data set.
EXITS	Indicates that user routines are provided.
LABELS	Specifies user-label processing.
MEMBER	Specifies the member name and alias of a member of a partitioned data set to be created.
RECORD	Defines a record group to be processed and supplies editing information.

Figure 9-5. IEBGENER Utitily Control Statements

When the output is to be sequential and editing is to be performed, one GENERATE statement and as many RECORD statements as required are used. If user exits are provided, an EXITS statement is used.

When the output is to be partitioned, one GENERATE statement, one MEMBER statement per output member, and RECORD statements, as required, are used. If user exits are provided, an EXITS statement is used.

Utility control statements may be continued on subsequent cards provided that the data starts in columns 4 through 16. A nonblank character in column 72 is optional for IEBGENER.

GENERATE Statement

The GENERATE statement is required when: (1) output is to be partitioned, (2) editing is to be performed, or (3) user routines are provided and/or label processing is specified. The GENERATE statement must appear before other statements. If it contains errors or is inconsistent with other statements, IEBGENER is terminated.

The format of the GENERATE statement is:

[*label*] **GENERATE** **[MAXNAME=***n***]**
[,MAXFLDS=*n***]**
[,MAXGPS=*n***]**
[,MAXLITS=*n***]**

EXITS Statement

The EXITS statement is used to identify exit routines supplied by the user. Linkages to and from exit routines are discussed in "Appendix A: Exit Routine Linkage."

For a detailed discussion of the processing of user labels as data set descriptors, and for discussion of user label totaling, refer to "Appendix D: Processing User Labels."

The EXITS statement is used when user routines are provided.

The format of the EXITS statement is:

[*label*] **EXITS** **[INHDR=***routinename***]**
[,OUTHDR=*routinename***]**
[,INTLR=*routinename***]**
[,OUTTLR=*routinename***]**
[,KEY=*routinename***]**
[,DATA=*routinename***]**
[,IOERROR=*routinename***]**
[,TOTAL=(*routinename*, *size***)]**

LABELS Statement

The LABELS statement specifies whether or not user labels are to be treated as data by IEBGENER. For a detailed discussion of this option, refer to "Processing User Labels as Data," in "Appendix D: Processing User Labels."

The LABELS statement is used when the user wants to specify that: (1) no user labels are to be copied to the output data set, (2) user labels are to be copied to the output data set from records in the data portion of the SYSIN data set, or (3) user labels are to be copied to the output data set after they are modified by the user's label processing routines. If more than one valid LABELS statement is included, all but the last LABELS statement are ignored.

The format of the LABELS statement is:

[*label*] **LABELS [DATA= {YES | NO | ALL | ONLY | INPUT}]**

Note: LABELS DATA=NO must be specified to make standard user labels (SUL) exits inactive when input/output data sets with nonstandard labels (NSL) are to be processed.

MEMBER Statement

The MEMBER statement is used when the output is to be partitioned. One MEMBER statement must be included for each member to be created by IEBGENER. The MEMBER statement provides the name and aliases of a member that is to be created.

All RECORD statements following a MEMBER statement pertain to the member named in that MEMBER statement. If no MEMBER statements are included, the output data set is organized sequentially.

The format of the MEMBER statement is:

[*label*] **MEMBER NAME=(***name* **[,** *alias* **]...)**

RECORD Statement

The RECORD statement is used to define a record group and to supply editing information. A record group consists of records that are to be processed identically.

The RECORD statement is used when: (1) the output is to be partitioned, (2) editing is to be performed, or (3) user labels for the output data set are to be created from records in the data portion of the SYSIN data set. The RECORD statement defines a record group by identifying the last record of the group with a literal name.

If no RECORD statement is used, the entire input data set or member is processed without editing. More than one RECORD statement may appear in the control statement stream for IEBGENER.

Within a RECORD statement, one IDENT parameter can be used to define the record group; one or more FIELD parameters can be used to supply the editing information applicable to the record group; and one LABELS parameter can be used to indicate that this statement is followed immediately by output label records.

The format of the RECORD statement is:

[*label*] **RECORD [IDENT=(***length* **,‘** *name* **’,** *input-location* **)]**

[**,FIELD=(**[*length*],[{ *input-location* | *'literal'* }],[conversion]
,[*output-location*]**)**[**,FIELD=**....]

[**,LABELS=***n*]

Operands	Applicable Control Statements	Description of Operands/Parameters
DATA	EXITS	**DATA=** *routinename* specifies the symbolic name of a routine that modifies the physical record (logical record for VS or VBS type records) before its processed by IEBGENER.
	LABELS	**DATA={YES | NO | ALL | ONLY | INPUT}** specifies whether user labels are to be treated as data by IEBGENER. These values can be coded: **YES** specifies that any user labels that are not rejected by a user's label processing routine are to be treated as data. Processing of labels as data ends in compliance with standard return codes. **NO** specifies that user labels are not to be treated as data. **ALL** specifies that user labels in the group currently being processed are to be treated as data regardless of any return code. A return code of 16 causes IEBGENER to complete processing the remainder of the group of user labels and to terminate the job step. **ONLY** specifies that only user header labels are to be treated as data. User header labels are processed as data regardless of any return code. The job terminates upon return from the OPEN routine. **INPUT** specifies that user labels for the output data set are supplied as 80-byte input records in the data portion of SYSIN. The number of input records that should be treated as user labels must be identified by a RECORD statement.

Operands	Applicable Control Statements	Description of Operands/Parameters
FIELD	RECORD	**FIELD=(**[*length*],[{*input-location* | *'literal'*}],[*conversion*],[*output-location*])**[,FIELD=...]**

specifies field-processing and editing information. Only the contents of specified fields in the input record is copied to the output record, that is, any field in the output record that is not specified will contain meaningless information. The values that can be coded are:

length
specifies the length (in bytes) of the input field or literal to be processed. If *length* is not specified, a length of 80 bytes is assumed. If a literal is to be processed, a length of 40 bytes or less must be specified.

input-location
specifies the starting byte of the field to be processed.

Default: Byte 1 is assumed.

'literal'
specifies a literal (maximum length of 40 bytes) to be placed in the specified output location. If a literal contains apostrophes, each apostrophe must be written as two consecutive apostrophes.

conversion
specifies a two-byte code that indicates the type of conversion to be performed on this field. If no conversion is specified, the field is moved to the output area without change. The values that can be coded are:

PZ
specifies that data (packed decimal) is to be converted to unpacked decimal data.

ZP
specifies that data (unpacked decimal) is to be converted to packed decimal data.

HE
specifies that data (H-set BCD) is to be converted to EBCDIC.

output-location
specifies the starting location of this field in the output records.

If *conversion* is specified in FIELD, the following restrictions apply:

- PZ-type (packed-to-unpacked) conversion is impossible for packed decimal records longer than 16K bytes.
- For ZP-type (unpacked-to-packed) conversion, the normal 32K-type maximum applies.
- When the ZP parameter is specified, the conversion is

performed in place. The original unpacked field is replaced by the new packed field. Therefore, the ZP parameter must be omitted from subsequent references to that field. If the field is needed in its original unpacked form, it must be referenced prior to the use of the ZP parameter.

If *conversion* is specified in the FIELD parameter, the length of the output record can be calculated for each conversion specification. When L is equal to the length of the input record, the calculation is made, as follows:

- For a PZ (packed-to-unpacked) specification, 2L-1.
- For a ZP (unpacked-to-packed) specification, (L/2) + C. If L is an odd number, C is 1/2; if L is an even number, C is 1.
- For an (H-set BCD to EBCDIC) specification, L.

If both output header labels and output trailer labels are to be contained in the SYSIN data set, the user must include one RECORD statement (including the LABELS parameter), indicating the number of input records to be treated as user labels, for header labels and one for trailer labels. The first such RECORD statement indicates the number of user header labels; the second indicates the number of user trailer labels. If only output trailer labels are included in the SYSIN data set, a RECORD statement must be included to indicate that there are no output header labels in the SYSIN data set (LABELS=0). This statement must precede the RECORD LABELS=n statement which signals the start of trailer label input records.

For a detailed discussion of the LABELS option, refer to "Processing User Labels As Data," in "Appendix D: Processing User Labels."

Default: Byte 1 is assumed.

Operands	Applicable Control Statements	Description of Operands/Parameters
IDENT	RECORD	**IDENT**=*(length,'name',input-location)* identifies that last record of the input group to which the FIELD parameters of MEMBER statement applies. If the RECORD statement is not followed by additional RECORD or MEMBER statements, IDENT also defines the last record to be processed. These values can be coded: *length* specifies the length (in bytes) of the identifying name. The length cannot exceed eight characters. *'name'* specifies the exact literal that identifies the last input record of a record group. **Default:** If no match for *name* is found, the remainder of the input data considered to be in one record group; subsequent RECORD and MEMBER statements are ignored. *input-location* specifies the starting locaiton of the field that contains the identifying name in the input records. **Default:** If IDENT is omitted, the remainder of the input data is considered to be in one record group; subsequent RECORD and MEMBER statements are ignored.
INHDR	EXITS	**INHDR**=*routinename* specifies the symbolic name of a routine that processes user input header labels.
INTLR	EXITS	**INTLR**=*routinename* specifies the symbolic name of a routine that processes user input trailer labels
IOERROR	EXITS	**IOERROR**=*routinename* specifies the symbolic name of a routine that handles permanent input/output error conditions.
KEY	EXITS	**KEY**=*routinename* specifies the symbolic name of a routine that creats the output record key. (This routine does not receive control when a data set consisting of VS or VBS type records is processed because no processing of keys is permitted for this type of data.)

Operands	Applicable Control Statements	Description of Operands/Parameters
LABELS	RECORD	**LABELS=***n* is an optional parameter that indicates the number of records in the SYSIN data set to be treated as user labels. The number *n*, which is a number from 1 to 8, must specify the exact number of label records that follow the RECORD statement. If this parameter is included, DATA=INPUT must be coded on a LABELS statement before it in the input stream.
MAXFLDS	GENERATE	**MAXFLDS=***n* specifies a number that is no less than the total number of FIELD parameter appearing in subsequent RECORD statements. MAXFLDS is required if there are any FIELD parameters in subsequent RECORD statements.
MAXGPS	GENERATE	**MAXGPS=***n* specifies a number that is no less than the total number of IDENT parameters appearing in subsequent RECORD statements. MAXGPS is required if there are any IDENT parameters in subsequent RECORD statements.
MAXLITS	GENERATE	**MAXLITS=***n* specifies a number that is no less than the total number of characters contained in the FIELD literals of subsequent RECORD statements. MAXLITS is required if the FIELD parameters of subsequent RECORD statements contain literals. MAXLITS does not pertain to literals used in IDENT parameters.
MAXNAME	GENERATE	**MAXNAME=***n* specifies a number that is no less than the total number of member names as aliases appearing in subsequent MEMBER statements. MAXNAME is required if there are one or more MEMBER statements.
NAME	MEMBER	**NAME=**(*name*[, *alias*]...) specifies a member name followed by a list of its aliases. If only one name appears in the statement, it need not be enclosed in parentheses.
OUTHDR	EXITS	**OUTHDR=** *routinename* specifies the symbolic name of a routine that creates user output header labels. OUTHDR is ignored if the output data set is partitioned.
OUTTLR	EXITS	**OUTTLR=***routinename* specifies the symbolic name of a routine that processes user output trailer labels. OUTTLR is ignored if the output data set is parititioned.

Operands	Applicable Control Statements	Description of Operands/Parameters
TOTAL	EXITS	TOTAL=(*routinename,size*) specifies that exits to a user's routine are to be provided prior to writing each record. The keyword OPTCD=T must be specified for the SYSUT2 DD statement. TOTAL is valid only when the utility is used to process sequential data sets. These values must be coded: *routinename* specifies the name of a user-supplied totaling routine. *size* specifies the number of bytes needed to contain totals, counters, pointers, etc.

Restrictions

- The SYSPRINT DD statement is required for each use of IEBGENER.
- The block size for the SYSPRINT data set must be multiple of 121. The block size for the SYSIN data set must be a multiple of 80. Any blocking factor can be specified for these block sizes.
- If the output data set is on a card punch or a printer, the user must specify DCB information on the SYSUT2 DD statement. DCB parameters in a SYSUT2 DD statement defining an expanded partitioned data set must be compatible with the specifications made when the data set was originally created.

  The SYSIN DD statement is required for each use of IEBGENER.

  Concatenated data sets with unlike attributes are not allowed as input to IEBGENER. For information on concatenated data sets, see *OS/VS2 MVS Data Management Services Guide.*
- When RECFM, BLKSIZE, and LRECL are not specified in the JCL for the output data set, values for each are copied from the input data set's DSCB.
- Always specify the output block size when the logical record length and record format (except for U) are specified. The default RECFM is U for the output data set. The output LRECL must be specified when editing is to be performed and the record format is FB, VS, or VBS. In all other cases, a default LRECL value is generated by IEBGENER.
- The input data set must always have a BLKSIZE parameter specified. The default RECFM is U for the input data set. The input LRECL must be specified when the record format is FB, VS, or VBS. In all other cases, a default LRECL is gneerated by IEBGENER.
- RECFM (except for undefined data sets), BLKSIZE, and LRECL (except for undefined data sets) must be specified on the SYSUT1/SYSUT2 DD statement when the data set is new, a dummy data set, a card punch, or a printer.
- A partitioned data set cannot be produced if an input or output data set contains spanned records.
- IEBGENER can not produce an output data set having a logical record length that differs from the input logical record length if both input and output RECFM are V or VB.

IEBGENER Examples

The examples that follow illustrate some of the uses of IEBGENER. Figure 9-6 can be used as a quick reference guide to IEBGENER examples. The numbers in the "Example" column point to the examples that follow.

Operation	Data Set Organization	Devices	Comments	Example
COPY	Sequential	Card Reader and Tape	Blocked output.	1
COPY–with editing	Sequential	Card Reader and Tape	Blocked output.	2
COPY–with editing	Sequential	Card Reader and Tape	Blocked output. Input includes //cards.	3
COPY–with editing	Sequential	Card Reader and Disk	Blocked output. Input includes // cards.	4
PRINT	Sequential	Card Reader and Printer	Input includes // cards. System output device is a printer.	5
CONVERT	Sequential input, Partitioned output	Tape and Disk	Blocked output. Three members are to be created.	6
COPY–with editing	Sequential	Disk	Blocked output. Two members are to be merged into existing data set.	7
COPY–with editing	Sequential	Tape	Blocked output. Data set edited as one record group.	8
COPY–with editing	Sequential	Disk	Blocked output. New record length specified for output data set. Two record groups specified.	9
COPY–with editing	Sequential	Tape	Blocked output. Data set edited as one record group.	10

Figure 9-6. IEBGENER Example Directory

Note: Examples which use *disk* or *tape*, in place of actual device-ids, must be changed before use. See the Device Support section, in the Introduction to this manual, for valid device-id notation.

IEBGENER Example 1

In this example, a card-input, sequential data set is to be copied to a 9-track tape volume.

The example follows:

```
//CDTOTAPE JOB  09#660,SMITH
//          EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DUMMY
//SYSUT2   DD   DSNAME=OUTSET,UNIT=tape,LABEL=(,SL),
// DISP=(,KEEP),VOLUME=SER=001234,DCB=(RECFM=FB,
// LRECL=80,BLKSIZE=2000)
//SYSUT1   DD   *
```

(input card data set)

```
/*
```

The control statements are discussed below:

- SYSIN DD defines a dummy data set. No editing is to be performed; therefore, no utility control statements are needed.
- SYSUT2 DD defines the output data set. The data set is written to a tape volume. The data set is to reside as the first (or only) data set on the volume.
- SYSUT1 DD defines the card-input data set. The data set can contain no // or /* cards.

IEBGENER Example 2

In this example, a card-input, sequential data set is to be copied to a tape volume. The control data set is a member of a partitioned data set.

```
//CDTOTAPE JOB   09#660,SMITH
//         EXEC  PGM=IEBGENER
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    DSNAME=CNTRLIBY(STMNTS),UNIT=disk,
// DISP=(OLD,KEEP),VOLUME=SER=111112,DCB=(RECFM=F,
// LRECL=80,BLKSIZE=80)
//SYSUT2   DD    DSNAME=OUTSET,UNIT=tape,LABEL=(,SL),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000),
// DISP=(,KEEP),VOLUME=SER=001234
//SYSUT1   DD    *
```

(input card data set)

```
/*
```

The control statements are discussed below:

- SYSIN DD defines the control data set, which contains the utility control statements. The control statements reside as a member, STMNTS, in a partitioned data set.
- SYSUT2 DD defines the output data set. The data set is written as the first data set on the tape volume.
- SYSUT1 DD defines the card-input data set. The data set can contain no // cards.

IEBGENER Example 3

In this example, a card-input, sequential data set is to be copied to a tape volume. The input contains cards that have slashes (//) in columns 1 and 2. The control data set is a member of a partitioned data set.

```
//CDTOTAPE JOB   09#660,SMITH
//         EXEC  PGM=IEBGENER
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    DSNAME=CNTRLIBY(STMNTS),UNIT=disk,
// DISP=(OLD,KEEP),VOLUME=SER=111112
//SYSUT2   DD    DSNAME=OUTSET,UNIT=tape,LABEL=(2,SL),
// VOLUME=SER=001234,DCB=(RECFM=FB,LRECL=80,
// BLKSIZE=2000),DISP=(,KEEP)
//SYSUT1   DD    DATA
```

(input card data set, including // cards)

```
/*
```

The control statements are discussed below:

- SYSIN DD defines the data set containing the utility control statements. The statements reside as a member, STMNTS, in a partitioned data set.
- SYSUT2 DD defines the copied sequential data set (output). The data set is written as the second data set on the specified tape volume.
- SYSUT1 DD defines the card-input data set. The data set is to be edited as specified in the utility control statements (not shown). The input data set contains // cards.

IEBGENER Example 4

In this example, a card-input, sequential data set is to be copied to a 2314 volume. The input data set contains // cards.

```
//CDTODISK JOB  09#660,SMITH
//          EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DSNAME=CNTRLIBY(STMNTS),UNIT=disk,
// DISP=(OLD,KEEP),VOLUME=SER=111112
//SYSUT2   DD   DSNAME=OUTSET,UNIT=disk,VOLUME=SER=111113,
// DISP=(,KEEP),SPACE=(TRK,(10,10)),DCB=(RECFM=FB,
// LRECL=80,BLKSIZE=2000)
//SYSUT1   DD   DATA
```

(input card data set, including // cards)

```
/*
```

The control statements are discussed below:

- SYSIN DD defines the control data set, which contains the utility control statements. The control statements reside as a member, STMNTS, in a partitioned data set.
- SYSUT2 DD defines the output data set. Ten tracks of primary storage space and ten tracks of secondary space are allocated for the data set on a disk volume.
- SYSUT1 DD defines the card-input data set. The data set is to be edited as specified in the utility control statements (not shown). The input data set contains // cards.

IEBGENER Example 5

In this example, the content of a card data set is to be printed. The printed output is to be left-aligned, with one 80-byte record appearing on each line of printed output.

```
//CDTOPTR  JOB  09#660,SMITH
//          EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DUMMY
//SYSUT2   DD   SYSOUT=A,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSUT1   DD   DATA
```

(input card data set, including // cards)

```
/*
```

The control statements are discussed below:

- SYSIN DD defines a dummy data set. No editing is to be performed; therefore, no utility control statements are required.
- SYSUT2 DD indicates that the output is to be written on the system output device (printer). Carriage control can be specified by changing the RECFM=F subparameter to RECFM=FA.
- SYSUT1 DD defines the input card data set. The input data set contains // cards.

IEBGENER Example 6

In this example, a partitioned data set (consisting of three members) is to be created from sequential input.

```
//TAPEDISK JOB   09#660,SMITH
//          EXEC PGM=IEBGENER
//SYSPRINT DD    SYSOUT=A
//SYSUT1    DD   DSNAME=INSET,UNIT=tape,LABEL=( ,SL),
// DISP=(OLD,KEEP),VOLUME=SER=001234
//SYSUT2    DD   DSNAME=NEWSET,UNIT=disk,DISP=( ,KEEP),
// VOLUME=SER=111112,SPACE=(TRK,(10,5,5)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//SYSIN     DD   *
     GENERATE    MAXNAME=3,MAXGPS=2
       MEMBER    NAME=MEMBER1
GROUP1 RECORD    IDENT=(8,'FIRSTMEM',1)
       MEMBER    NAME=MEMBER2
GROUP2 RECORD    IDENT=(8,'SECNDMEM',1)
       MEMBER    NAME=MEMBER3
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set (INSET). The data set is the first data set on a tape volume.
- SYSUT2 DD defines the output partitioned data set (NEWSET). The data set is to be placed on a disk volume. Ten tracks of primary space, five tracks of secondary space, and five blocks (256 bytes each) of directory space are allocated to allow for future expansion of the data set. The output records are blocked to reduce the space required by the data set.
- SYSIN DD defines the control data set, which follows in the input stream. The utility control statements are used to create members from sequential input data; the statements do not specify any editing.
- GENERATE indicates that: (1) three member names are included in subsequent MEMBER statements and (2) the IDENT parameter appears twice in subsequent RECORD statements.
- The first MEMBER statement assigns a member name (MEMBER1) to the first member.
- The first RECORD statement (GROUP1) identifies the last record to be placed in the first member. The name of this record (FIRSTMEM) appears in bytes 1 through 8 of the input record.
- The remaining MEMBER and RECORD statements define the second and third members.

IEBGENER Example 7

In this example, sequential input is to be converted into two partitioned members. The newly created members are to be merged into an existing partitioned data set. User labels on the input data set are to be passed to the user exit routines.

```
//DISKTODK JOB  09#660,SMITH
//            EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=INSET,UNIT=disk,DISP=(OLD,KEEP),
// VOLUME=SER=111112,
// LABEL=(,SUL)
//SYSUT2   DD   DSNAME=EXISTSET,UNIT=disk,DISP=(MOD,KEEP),
// VOLUME=SER=111113
//SYSIN    DD   *
     GENERATE   MAXNAME=3,MAXGPS=1
        EXITS   INHDR=ROUT1,INTLR=ROUT2
       MEMBER   NAME=(MEMX,ALIASX)
GROUP1 RECORD   IDENT=(8,'FIRSTMEM',1)
       MEMBER   NAME=MEMY
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set (INSET). The input data set, which resides on a disk volume, has standard and user labels.
- SYSUT2 DD defines the output partitioned data set (EXISTSET). The members created during this job step are merged into the partitioned data set.
- SYSIN DD defines the control data set, which follows in the input stream. The utility control statements are used to create members from sequential input data; the statements do not specify any editing.
- GENERATE indicates that: (1) two member names and one alias are included in subsequent MEMBER statements and (2) an IDENT parameter appears in a subsequent RECORD statement.
- EXITS defines the user routines that are to process user labels.
- The first MEMBER statement assigns a member name (MEMX) and an alias (ALIASX) to the first member.
- The first RECORD statement identifies the last record to be placed in the first member. The name of this record (FIRSTMEM) appears in bytes 1 through 8 of the input record.
- The second MEMBER statement assigns a member name (MEMY) to the second member. The remainder of the input data set is included in this member.

IEBGENER Example 8

In this example, a sequential input data set is to be edited and copied.

```
                                                                72
//TAPETAPE JOB  09#660,SMITH
//            EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=OLDSET,UNIT=tape,DISP=(OLD,KEEP),
// VOLUME=SER=001234,LABEL=(3,SL)
//SYSUT2   DD   DSNAME=NEWSET,UNIT=tape,DISP=(NEW,PASS),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000),
// VOLUME=SER=001235,LABEL=(,SL)
//SYSIN    DD   *
     GENERATE   MAXFLDS=3,MAXLITS=11
       RECORD   FIELD=(10,'**********',,1),                     C
               FIELD=(5,1,HE,11),FIELD=(1,'=',,16)
        EXITS   INHDR=ROUT1,OUTTLR=ROUT2
       LABELS   DATA=INPUT
       RECORD   LABELS=2
```

```
(first header label record)
(second header label record)
          RECORD     LABELS=2
(first trailer label record)
(second trailer label record)
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the sequential input data set (OLDSET). The data set was originally written as the third data set on a tape volume.
- SYSUT2 DD defines the sequential output data set (NEWSET). The data set is written as the first data set on a tape volume. The output records are blocked to reduce the space required by the data set and to reduce the access time required when the data set is subsequently referred to. The data set is passed to a subsequent job step.
- SYSIN DD defines the control data set, which follows in the input stream.
- GENERATE indicates that: (1) a maximum of three FIELD parameters is included in subsequent RECORD statements and (2) a maximum of 11 literal characters are included in subsequent FIELD parameters.
- EXITS indicates that the specified user routines require control when SYSUT1 is opened and when SYSUT2 is closed.
- LABELS indicates that labels are included in the input stream.
- The first RECORD statement controls the editing, as follows: (1) asterisks are placed in positions 1 through 10, (2) bytes 1 through 5 of the input record are converted from H-set BCD to EBCDIC mode and moved to positions 11 through 15, and (3) an equal sign is placed in byte 16.
- The second RECORD statement indicates that the next two records from SYSIN should be written out as user header labels on SYSUT2.
- The third RECORD statement indicates that the next two records from SYSIN should be written as user trailer labels on SYSUT2.

Note: This example shows the relationship between the RECORD LABELS statement and the EXITS statement. IEBGENER attempts to write a first and second label trailer as user labels at close time of SYSUT2 before returning control to the system; the user routine, ROUT2, can review these records and change them, if necessary.

IEBGENER Example 9

In this example, a sequential input data set is to be edited and copied.

```
                                                                      72
//DISKDISK JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=OLDSET,UNIT=disk,DISP=(OLD,KEEP),
// VOLUME=SER=111112
//SYSUT2   DD   DSNAME=NEWSET,UNIT=disk,DISP=(NEW,KEEP),
// VOLUME=SER=111113,DCB=(RECFM=FB,LRECL=80,
// BLKSIZE=640),SPACE=(TRK,(20,10))
//SYSIN    DD   *
     GENERATE   MAXFLDS=4,MAXGPS=1
        EXITS   IOERROR=ERRORRT
GROUP1 RECORD   IDENT=(8,'FIRSTGRP',1),                               C
               FIELD=(21,80,,60),FIELD=(59,1,,1)
GROUP2 RECORD   FIELD=(11,90,,70),FIELD=(69,1,,1)
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set (OLDSET).
- SYSUT2 DD defines the output data set (OUTSET). Twenty tracks of primary storage space and ten tracks of secondary storage space are allocated for the data set on a disk volume. The logical record length of the output records is 80 bytes, and the output is blocked.
- SYSIN DD defines the control data set, which follows in the input stream.
- GENERATE indicates that: (1) a maximum of four FIELD parameters is included in subsequent RECORD statements and (2) a maximum of one IDENT parameter appears in a subsequent RECORD statement.
- EXITS identifies the user routine that handles input/output errors.
- The first RECORD statement controls the editing of the first record group, as follows: (1) FIRSTGRP, which appears in bytes 1 through 8 of the input record, is defined as being the last record in the first group of records and (2) bytes 80 through 100 of each input record are moved into positions 60 through 80 of each corresponding output record. (This example implies that bytes 60 through 79 of the input records in the first record group are no longer required; thus, the logical record length is shortened by 20 bytes.) The remaining bytes within each input record are transferred directly to the output records, specified in the second FIELD parameter.
- The second RECORD statement indicates that the remainder of the input records are to be processed as the second record group. Bytes 90 through 100 of each input record are moved into positions 70 through 80 of the output records. (This example implies that bytes 70 through 89 of the input records from group 2 are no longer required; thus, the logical record length is shortened by 20 bytes.) The remaining bytes within each input record are transferred directly to the output records, specified in the second FIELD parameter.

If the logical record length of the output data set differs from that of the input data set, as in this example, all positions in the output records must undergo editing to justify the new logical record length.

IEBGENER Example 10

In the example, a sequential input data set is to be edited and copied.

```
                                                           72
//TAPETAPE JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=OLDSET,UNIT=tape,DISP=(OLD,KEEP),
// VOLUME=SER=001234,LABEL=(3,SUL)
//SYSUT2   DD   DSNAME=NEWSET,UNIT=tape,DISP=(NEW,PASS),
// VOLUME=SER=001235,LABEL=(,SUL),DCB=(RECFM=FB,
// LRECL=80,BLKSIZE=2000)
//SYSIN    DD   *
     GENERATE   MAXFLDS=3,MAXLITS=11
       RECORD   FIELD=(10,'**********',,1),                 C
               FIELD=(5,1,HE,11),FIELD=(1,'=',,16)
       LABELS   DATA=INPUT
       RECORD   LABELS=3
```

(first header label record)
(second header label record)
(third header label record)

```
       RECORD   LABELS=2
```

(first trailer label record)
(second trailer label record)

```
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set (OLDSET). The data set is the third data set on a tape volume.
- SYSUT2 DD defines the output data set (NEWSET). The data set is written as the first or only data set on a tape volume. The output records are blocked to reduce the space required by the data set and to reduce the access time required when the data set is subsequently referred to. The data set is passed to a subsequent job step.
- SYSIN DD defines the control data set, which follows in the input stream.
- GENERATE indicates that: (1) a maximum of three FIELD parameters is included in subsequent RECORD statements and (2) a maximum of 11 literal characters are included in subsequent FIELD parameters.
- LABELS indicates that label records are included in the input stream.
- The first RECORD statement controls the editing, as follows: (1) asterisks are placed in positions 1 through 10, (2) bytes 1 through 5 of the input record are converted from H-set BCD to EBCDIC mode and moved to positions 11 through 15, and (3) an equal sign is placed in byte 16.
- The second RECORD statement indicates that three 80-byte records (cards), to be written as user labels on the output data set, immediately follow. The third RECORD statement indicates that the following cards are to be treated as trailer labels.

IEBISAM PROGRAM

IEBISAM can be used to:

- Copy an indexed sequential (ISAM) data set directly from one direct access volume to another.
- Create a backup (transportable) copy of an ISAM data set by copying (unloading) it into a sequential data set on a direct access or magnetic tape volume.
- Create an ISAM data set from an unloaded data set. The sequential (unloaded) data set is in a form that can be subsequently loaded, that is, it can be converted back into an ISAM data set.
- Print an ISAM data set.

At the completion or termination of IEBISAM, the highest return code encountered within the program is passed to the calling program.

Copying an Indexed Sequential Data Set

IEBISAM can be used to copy an indexed sequential data set directly from one DASD volume to another. When the data set is copied, the records marked for deletion are only deleted if the DELETE parameter was specified in the OPTCD (optional control program service) field. Those records that are contained in the overflow area of the original data set are moved into the primary area of the copied data set. The control information characteristics such as BLKSIZE and OPTCD can be overridden by new specifications. Caution should be used, however, when overriding these characteristics (see "Specifying a LOAD operation").

Creating a Sequential Backup Copy

An *unloaded* sequential data set can be created to serve as a backup or transportable copy of source data from an indexed sequential data set. Records marked for deletion within the indexed sequential data set are automatically deleted when the unloaded data set is created. When the data set is subsequently *loaded*—reconstructed into an indexed sequential data set—records that were contained in the overflow area assigned to the original data set are moved sequentially into the primary area.

An unloaded data set consists of 80-byte logical records. The data set contains:

- Fixed records from an indexed sequential data set.
- Control information used in the subsequent loading of the data set.

Control information consists of characteristics that were assigned to the indexed sequential data set. These characteristics are:

- Optional control program service (OPTCD)
- Record format (RECFM)
- Logical record length (LRECL)
- Block size (BLKSIZE)
- Relative key position (RKP)
- Number of tracks in cylinder index (NTM)

- Key length (KEYLEN)
- Number of overflow tracks on each cylinder (CYLOFL)

Specifying a Load Operation

When a load operation is specified, these characteristics can be overridden by specifications in the DCB parameter of the SYSUT2 DD statement (refer to "Job Control Statements" for a discussion of the SYSUT2 DD statement). Caution should be used, however, because checks are made to ensure that:

1. Record format is the same as that of the original indexed sequential data set (either fixed (F) or variable (V) length).
2. Logical record length is greater than or equal to that of the original indexed sequential data set when the RECFM is variable (V) or variable blocked (VB).
3. For fixed records, the block size is equal to or a multiple of the logical record length of the records in the original indexed sequential data set. For variable records, the block size is equal to or greater than the logical record length plus four.
4. Relative key position is equal to or less than the logical record length minus the key length. Following are relative key position considerations:
   - If the RECFM is V or VB, the relative key position should be at least 4.
   - If the DELETE parameter was specified in the OPTCD field and the RECFM is F or fixed blocked (FB), the relative key position should be at least 1.
   - If the DELETE parameter was specified in the OPTCD field and the RECFM is V or VB, the relative key position should be at least 5.
5. The key length is less than or equal to 255 bytes.
6. For a fixed unblocked data set with RKP=0, the LRECL value is the length of the data portion, not, as in all other cases, the data portion and key length. When changing the RECFM from fixed unblocked and RKP=0 to fixed blocked, the new LRECL must be equal to the old LRECL **plus** the old key length.

If either RKP or KEYLEN is overridden, it might not be possible to reconstruct the data set.

The number of 80-byte logical records in an unloaded data set can be determined by the formula:

$$x = \frac{n(y+2) + 158}{78}$$

where x is the number of 80-byte logical records created, n is the number of records in the indexed sequential data set, and y is the length of a fixed record or the average length of variable records.

Figure 10-1 shows the format of an unloaded data set for the first three 100-byte records of an indexed sequential data set. Each is preceded by two bytes (bb) that indicate the number of bytes in that record. (The last record is followed by two bytes containing binary zeros to identify the last logical record in the unloaded data set.) The characteristics of the indexed sequential data set are contained in the first two logical records of the unloaded data set. Data from the indexed sequential data set begins in the third logical record. Each logical record in the unloaded data set contains a binary sequence number (aa) in the first two bytes of the record.

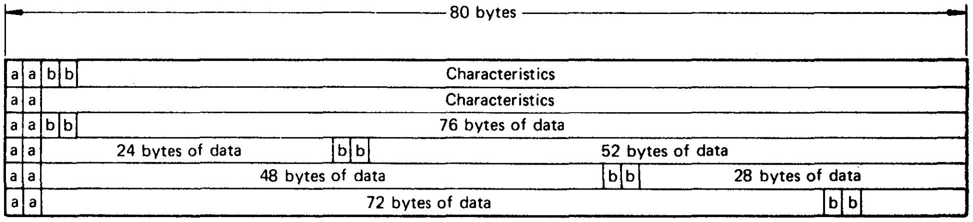


Figure 10-1. An Unloaded Data Set Created Using IEBISAM

Creating an Indexed Sequential Data Set from an Unloaded Data Set

An indexed sequential data set can be created from an unloaded version of an indexed sequential data set. When the unloaded data set is loaded, those records that were contained in the overflow area assigned to the original indexed sequential data set are moved sequentially into the primary area of the loaded indexed sequential data set.

Printing the Logical Records of an Indexed Sequential Data Set

The records of an indexed sequential data set can be printed or stored as a sequential data set for subsequent printing. Each input record is placed in a buffer from which it is printed or placed in a sequential data set. When the DELETE parameter is specified in the OPTCD field, each input record not marked for deletion is also placed in a buffer from which it is printed or placed in a sequential data set. Each printed record is converted to hexadecimal unless specified otherwise by the user.

IEBISAM provides user exits so that the user can include his own routines to:

- Modify records before printing.
- Select records for printing or terminate the printing operation after a certain number of records have been printed.
- Convert the format of a record to be printed.
- Provide a record heading for each record if the record length is at least 18 bytes. If no user routines are provided, each record is identified in sequential order on the printout.

When a user routine is supplied for a print operation, IEBISAM issues a LOAD macro instruction. A BALR 14,15 instruction is used to give control to the user's routine. When the user's routine receives control, register 0 contains a pointer to a record heading buffer; register 1 contains a pointer to an input record buffer. (Note that the user must save registers 2 through 14 when control is given to the user routine.)

The input record buffer has a length equal to the length of the input logical record.

Figure 10-2 shows the record heading buffer.

The user returns control to IEBISAM by issuing a RETURN macro instruction (via register 14) or by using a BR 14 instruction after restoring registers 2 through 14.

A user routine must place a return code in register 15 before returning control to IEBISAM. The possible return codes and their meanings are:

- 00, which indicates that buffers are to be printed.

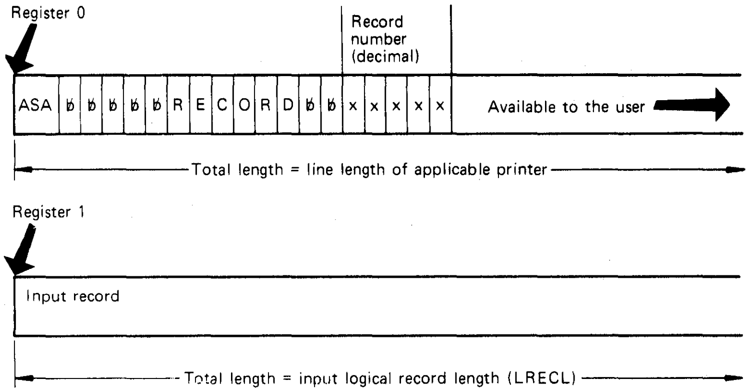


Figure 10-2. Record Heading Buffer Used by IEBISAM

- 04, which indicates that the buffers are to be printed and the operation is to be terminated.
- 08, which indicates that this input record is not to be printed; processing continues.
- 12, which indicates that this input record is not to be printed; terminate the operation.

Input and Output

IEBISAM uses an input data set (the organization of the input data set depends on the operation to be performed) as follows:

- If a data set is to be copied, unloaded, or printed in logical sequence, the input is an indexed sequential data set.
- If a data set is to be loaded, the input is an unloaded version of an indexed sequential data set.

IEBISAM produces as output:

- An output data set, which is the result of the IEBISAM operation.
- A message data set, which contains information messages and any error messages.

IEBISAM provides a return code to indicate the results of program execution. The return codes and their meanings are:

- 00, which indicates successful completion.
- 04, which indicates that a return code of 04 or 12 was passed to IEBISAM by the user routine.
- 08, which indicates that an error condition occurred that caused termination of the operation.
- 12, which indicates that a return code other than 00, 04, 08, or 12 was passed to IEBISAM from a user routine. The job step is terminated.
- 16, which indicates that an error condition caused termination of the operation.

Control

IEBISAM is controlled by job control statements. No utility control statements are required.

Job Control Statements

Figure 10-3 shows the job control statements necessary for using IEBISAM.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEBISAM). Additional information is required on the EXEC statement to control the execution of IEBISAM; see "PARM Information on the EXEC Statement" below.
SYSUT1 DD	Defines the input data set.
SYSUT2 DD	Defines the output data set.
SYSPRINT DD	Defines a sequential message data set, which can be written to a system output device, a tape volume, or a direct access device.

Figure 10-3. IEBISAM Job Control Statements

If the block size of the SYSPRINT data set is not a multiple of 121, a default value of 121 is taken (no error message is issued, and no condition code is set).

PARM Information on the EXEC Statement

The PARM parameter on the EXEC statement is used to control the execution of IEBISAM.

Note: Exit routines must be included in either the job library or the link library.

For a COPY operation, the SYSUT2 DD statement must include a primary space allocation that is sufficient to accommodate records that were contained in overflow areas in the original indexed sequential data set. New overflow areas can be specified when the data set is copied.

For an UNLOAD operation, specifications that are implied by default or included in the DCB parameter of the SYSUT2 DD statement (for example, tape density) must be considered when the data set is subsequently loaded. If a block size is specified in the DCB parameter of the SYSUT2 DD statement, it must be a multiple of 80 bytes.

For a LOAD operation, if the input data set resides on an unlabeled tape, the SYSUT1 DD statement must specify a BLKSIZE that is a multiple of 80 bytes. Specifications that are implied by default or included in the DCB parameter of the SYSUT1 DD statement must be consistent with specifications that were implied or included in the DCB parameter of the SYSUT2 DD statement used for the UNLOAD operation. The SYSUT2 DD statement must include a primary space allocation that is sufficient to accommodate records that were contained in overflow areas in the original indexed sequential data set. If new overflow areas are desired, they must be specified when the data set is loaded.

For a PRINTL operation, if the device defined by the SYSUT2 DD statement is a printer, the specified BLKSIZE must be equal to or less than the physical printer size; that is 121, 133, or 145 bytes. If BLKSIZE is not specified, 121 bytes is assumed. LRECL (or BLKSIZE when no LRECL was specified) must be between 55 and 255 bytes.

If a user routine is supplied for a PRINTL operation, IEBISAM issues a LOAD macro instruction to make the user routine available. A BALR 14,15 instruction is subsequently used to give control to the routine. When the user routine receives control, register 0 contains a pointer to a *record heading buffer;* register 1 contains a pointer to an *input record buffer.*

<table>
<tr><th>Operands</th><th>Applicable Control Statements</th><th>Description of Operands/Parameters</th></tr>
<tr><td>PARM</td><td>EXEC</td><td>PARM={COPY | UNLOAD | LOAD | PRINTL | PRINTL[,N] [,EXIT=<i>routinename</i>]}
The PARM values have the following meaning:
• COPY specifies a copy operation.
• UNLOAD specifies an unload operation.
• LOAD specifies a load operation.
• PRINTL specifies a print operation in which each record is converted to hexadecimal before printing. The N is an optional value that specifies that records are not to be converted to hexadecimal before printing.
• EXIT is an optional value that specifies the name of an exit routine that is to receive control before each record is printed.</td></tr>
</table>

IEBISAM Examples

The following examples illustrate some of the uses of IEBISAM. Figure 10-4 can be used as a quick reference guide to IEBISAM examples. The numbers in the "Example" column point to the examples that follow.

Operation	Data Set Organization	Devices	Comments	Example
COPY	Indexed sequential	Disks	Unblocked input; blocked output. Prime area and index separation.	1
UNLOAD	Indexed-sequential, Sequential	Disk and 9-track Tape	Blocked output.	2
UNLOAD	Indexed sequential, Sequential	Disk and 7-track Tape	Blocked output. Data set written as second data set on input volume.	3
LOAD	Sequential, Indexed sequential	9-track Tape and Disk	Input data set is second data set on tape volume.	4
PRINTL	Indexed sequential, Sequential	Disk and System Printer	Blocked input. Output not converted.	5

Figure 10-4.IEBISAM Example Directory

Note: Examples which use *disk* or *tape*, in place of actual device-ids, must be changed before use. See the Device Support section, in the Introduction to this manual, for valid device-id notation.

IEBISAM Example 1

In this example, an indexed sequential data set is to be copied from two DASD volumes. The output data is blocked.

```
//CPY      JOB  09#770,SMITH
//         EXEC PGM=IEBISAM,PARM=COPY
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=ISAM01,VOLUME=SER=(222222,333333),
// DISP=(OLD,DELETE),UNIT=(disk,2),DCB=(DSORG=IS,
// LRECL=500,BLKSIZE=500,RECFM=F,RKP=4)
//SYSUT2   DD   DSNAME=ISAM02(INDEX),UNIT=disk,DISP=(NEW,
// KEEP),VOLUME=SER=444444,DCB=(DSORG=IS,BLKSIZE=1000,
// RECFM=FB),SPACE=(CYL,(2))
//         DD   DSNAME=ISAM02(PRIME),UNIT=(disk,2),
// DCB=(DSORG=IS,BLKSIZE=1000,RECFM=FB),SPACE=(CYL,(10)),
// VOLUME=SER=(444444,555555),DISP=(NEW,KEEP)
/*
```

The control statements are discussed below:

- EXEC specifies the program name and the COPY operation.
- SYSUT1 DD defines an indexed sequential input data set, which resides on two disk volumes.
- SYSUT2 DD defines the output data set index area; the index and prime areas are separated.
- The second SYSUT2 DD defines the output data set prime area. Ten cylinders are allocated for the prime area on each of the two disk volumes.

IEBISAM Example 2

In this example, indexed sequential input is to be converted into a sequential data set; the output is to be placed on a 9-track tape volume.

```
//STEP1     JOB  09#770,SMITH
//          EXEC PGM=IEBISAM,PARM=UNLOAD
//SYSPRINT DD    SYSOUT=A
//SYSUT1    DD   DSNAME=INDSEQ,UNIT=disk,DISP=(OLD,KEEP),
// VOLUME=SER=111112
//SYSUT2    DD   DSNAME=UNLDSET,UNIT=tape,LABEL=( ,SL),
// DISP=( ,KEEP),VOLUME=SER=001234,DCB=(RECFM=FB,
// LRECL=80,BLKSIZE=640)
/*
```

The control statements are discussed below:

- EXEC specifies the program name and the UNLOAD operation.
- SYSUT1 DD defines the indexed sequential input data set, which resides on a disk volume.
- SYSUT2 DD defines the unloaded output data set. The data set consists of fixed blocked records, and is to reside as the first or only data set on a 9-track tape volume. The data set is to be written at a density of 800 bits per inch.

IEBISAM Example 3

In this example, indexed sequential input is to be converted into a sequential data set and placed on a 7-track; tape volume.

```
//STEPA     JOB  09#770,SMITH
//          EXEC PGM=IEBISAM,PARM=UNLOAD
//SYSPRINT DD    SYSOUT=A
//SYSUT1    DD   DSNAME=INDSEQ,UNIT=disk,DISP=(OLD,KEEP),
// VOLUME=SER=111112
//SYSUT2    DD   DSNAME=UNLDSET,UNIT=2400-2,LABEL=(2,SL),
// VOLUME=SER=001234,DCB=(DEN=2,RECFM=FB,LRECL=80,
// BLKSIZE=1040,TRTCH=C),DISP=( ,KEEP)
/*
```

The control statements are discussed below:

- EXEC specifies the program name and the UNLOAD operation.
- SYSUT1 DD defines the input data set, which is an indexed sequential data set. The data set resides on a disk volume.
- SYSUT2 DD defines the unloaded output data set. The data set consists of fixed blocked records, and is to reside as the second data set on a 7-track tape volume. The data set is to be written at a density of 800 bits per inch.

IEBISAM Example 4

In this example, an unloaded data set is to be converted to the form of the original indexed sequential data set.

```
//STEPA     JOB  09#770,SMITH
//          EXEC PGM=IEBISAM,PARM=LOAD
//SYSPRINT DD    SYSOUT=A
//SYSUT1    DD   DSNAME=UNLDSET,UNIT=tape,LABEL=(2,SL),
// DISP=(OLD,KEEP),VOLUME=SER=001234
//SYSUT2    DD   DSNAME=INDSEQ,DISP=( ,KEEP),DCB=(DSORG=IS),
// SPACE=(CYL,(1)),VOLUME=SER=111112,UNIT=disk
/*
```

The control statements are discussed below:

- EXEC specifies the program name and the LOAD operation.
- SYSUT1 DD defines the input data set, which is a sequential (unloaded) data set. The data set is the second data set on a tape volume.
- SYSUT2 DD defines the output data set, which is an indexed sequential data set. One cylinder of space is allocated for the data set on a disk volume.

IEBISAM Example 5

In this example, the logical records of an indexed sequential data set are to be printed on a system output device.

```
//PRINT    JOB  09#770,SMITH
//         EXEC PGM=IEBISAM,PARM='PRINTL,N'
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=ISAM03,UNIT=disk,DISP=OLD,
// VOLUME=SER=222222
//SYSUT2   DD   SYSOUT=A
/*
```

The control statements are discussed below:

- EXEC specifies the program name and the PRINTL operation. The output records are not to be converted to hexadecimal prior to printing.
- SYSUT1 DD defines the input data set, which resides on a disk volume.
- SYSUT2 DD defines the output data set. A logical record length (LRECL) of 121 bytes is assumed.

IEBPTPCH PROGRAM

IEBPTPCH is a data set utility used to print or punch all, or selected portions, of a sequential or partitioned data set. Records can be printed or punched to meet either standard specifications or user specifications.

The standard specifications are:

- Each logical record begins on a new printed line or punched card.
- Each printed line consists of groups of 8 characters separated by 2 blanks. Each punched card contains up to 80 contiguous bytes of information.
- Characters that cannot be printed appear as blanks.
- When the input is blocked, each logical record is delimited by "*" and each block is delimited by "**".

User formats can be specified, provided that no output record exceeds the capability of the output device.

IEBPTPCH provides optional editing facilities and exits for user routines that can be used to process labels or manipulate input or output records.

IEBPTPCH can be used to:

- Print or punch a sequential or partitioned data set in its entirety.
- Print or punch selected members from a partitioned data set.
- Print or punch selected records from a sequential or partitioned data set.
- Print or punch the directory of a partitioned data set.
- Print or punch an edited version of a sequential or partitioned data set.

At the completion or termination of the program, the highest return code encountered within the program is passed to the calling program.

Printing or Punching a Data Set

IEBPTPCH can be used to print or punch a sequential data set or a partitioned data set in its entirety. Data to be printed or punched can be either hexadecimal or a character representation of valid alphameric bit configurations. For a print operation, packed decimal data should be converted to unpacked decimal or hexadecimal mode to ensure that all characters are printable.

For a standard print operation, each logical record is printed in groups of eight characters. Each set of eight characters is separated from the next by two blanks. Up to 96 data characters can be included on a printed line. (An edited output can be produced to omit the blank delimiters and print up to 144 characters per line.)

Data from an input logical record is punched in contiguous columns in the punched card(s) representing that record. Sequence numbers can be created and placed in columns 73 through 80 of the punched cards.

Printing or Punching Selected Members

IEBPTPCH can be used to print or punch selected members of a partitioned data set. Utility control statements are used to specify members to be printed or punched.

Printing or Punching Selected Records

IEBPTPCH can be used to print selected records from a sequential or partitioned data set. Utility control statements can be used to specify:

- The termination of a print or punch operation after a specified number of records has been printed or punched.
- The printing or punching of every *nth* record.
- The starting of a print or punch operation after a specified number of records.

Printing or Punching a Partitioned Directory

IEBPTPCH can be used to print or punch the contents of a partitioned directory. Each directory block is printed in groups of eight characters. If the directory is printed in hexadecimal representation, the first four printed characters of each directory block indicate the total number of used bytes in that block. For details of the format of the directory, see *OS/VS2 Data Areas.*

Data from a directory block is punched in contiguous columns in the punched cards representing that block.

Printing or Punching an Edited Data Set

IEBPTPCH can be used to print or punch an edited version of a sequential or a partitioned data set. Utility control statements can be used to specify editing information that applies to a record, a group of records, selected groups of records, or an entire member or data set.

An edited data set is produced by:

- Rearranging or omitting defined data fields within a record.
- Converting data from packed decimal to unpacked decimal or from alphameric to hexadecimal representation.

Input and Output

IEBPTPCH uses the following input:

- An input data set, which contains the data that is to be printed or punched. The input data set can be either sequential or partitioned.
- A control data set, which contains utility control statements. The control data set is required for each use of IEBPTPCH.

IEBPTPCH produces the following output:

- An output data set, which is the printed or punched data set.
- A message data set, which contains informational messages (for example, the contents of the control statements) and any error messages.

IEBPTPCH provides a return code to indicate the results of program execution. The return codes and their meanings are:

- 00, which indicates successful completion.
- 04, which indicates that either a physical sequential data set is empty or a partitioned data set has no members.
- 08, which indicates that a member specified for printing does not exist in the input data set. Processing continues with the next member.

- 12, which indicates that an unrecoverable error occurred or that a user routine passed a return code of 12 to IEBPTPCH. The job step is terminated.
- 16, which indicates that a user routine passed a return code of 16 to IEBPTPCH. The job step is terminated.

Control

IEBPTPCH is controlled by job control statements and utility control statements. The job control statements are required to execute or invoke the IEBPTPCH program and to define the data sets that are used and produced by the program. The utility control statements are used to control the functions of IEBPTPCH.

Job Control Statements

Figure 11-1 shows the job control statements necessary for using IEBPTPCH.

Statement	Use
JOB	Initiates the job step.
EXEC	Specifies the program name (PGM=IEBPTPCH) or, if the job control statements reside in a procedure library, the procedure name.
SYSPRINT DD	Defines a sequential message data set. The data set can be written to a system output device, a tape volume, or a direct access device.
SYSUT1 DD	Defines a sequential or partitioned input data set.
SYSUT2 DD	Defines the output (printed or punched) data set.
SYSIN DD	Defines the control data set. The control data set normally resides in the input stream; however, it can be defined as a member in a partitioned data set.

Figure 11-1. IEBPTPCH Job Control Statements

The input data set can contain fixed, variable, undefined, or variable spanned records.

Both the output data set and the message data set can be written to the system output device if it is a printer. Variable spanned records are allowed only when the input is sequential.

If the logical record length of the input records is such that the output would exceed the output record length, the utility divides the record into multiple lines or cards in the case of standard printed output, standard punched output, or when the PREFORM operand was specified. Otherwise, only part of the input record is printed (a maximum of 144 characters) or punched (a maximum of 80 characters).

Utility Control Statements

IEBPTPCH is controlled by utility control statements. The control statements in Figure 11-2 are shown in the order in which they must appear.

The control statements are included in the control data set, as required. Any number of MEMBER and RECORD statements can be included in a job step.

A nonblank character in column 72 is optional for IEBPTPCH continuation statements.

Statement	Use
PRINT or PUNCH	Specifies that the data is to be either printed or punched.
TITLE	Specifies that a title is to precede the printed or punched data.
EXITS	Specifies that user routines are provided.
MEMBER	Specifies that the input is a partitioned data set and that a selected member is to be printed or punched.
RECORD	Specifies whether editing is to be performed, that is, records are to be printed or punched to nonstandard specifications.
LABELS	Specifies whether user labels are to be treated as data.

Figure 11-2. IEBPTPCH Utility Control Statements

PRINT Statement

The PRINT statement is used to initiate the IEBPTPCH operation. If this is a print operation, PRINT must be the first statement in the control data set.

The format of the PRINT statement is:

[*label*] **PRINT [PREFORM={A | M}]**
[,TYPORG={PS | PO}]
[,TOTCONV={XE | PZ}]
[,CNTRL={*n* | 1}
[,STRTAFT=*n*]
[,STOPAFT=*n*]
[,SKIP=*n*]
[,MAXNAME=*n*]
[,MAXFLDS=*n*]
[,MAXGPS=*n*]
[,MAXLITS=*n*]
[,INITPG=*n*]
[,MAXLINE=*n*]

PUNCH Statement

The PUNCH statement is used to initiate the IEBPTPCH operation. If this is a punch operation, PUNCH must be the first statement in the control data set.

The format of the PUNCH statement is:

[*label*] **PUNCH** **[PREFORM={A | M}]**
[,TYPORG={PS | PO}]
[,TOTCONV={XE | PO}]
[,CNTRL={ *n* **| 1 }]**
[,STRTAFT=*n* **]**
[,STOPAFT=*n* **]**
[,SKIP=*n* **]**
[,MAXNAME=*n* **]**
[,MAXFLDS=*n* **]**
[,MAXGPS=*n* **]**
[,MAXLITS=*n* **]**
[,CDSEQ=*n* **]**
[,CDINCR=*n* **]**

TITLE Statement

The TITLE statement is used to request title and subtitle records. Two TITLE statements can be included for each use of IEBPTPCH. A first TITLE statement defines the title, and a second defines the subtitle. The TITLE statement, if included, follows the PRINT or PUNCH statement in the control data set.

The format of the TITLE statement is:

[*label*] **TITLE ITEM=('** *title*'[, *output-location*])**[,ITEM...]**

EXITS Statement

The EXITS statement is used to identify exit routines supplied by the user. Exits to label processing routines are ignored if the input data set is partitioned. Linkage to and from user routines are discussed in "Appendix A: Exit Routine Linkage."

The EXITS statement, if included, must immediately follow any TITLE statement or follow the PRINT or PUNCH statement.

The format of the EXITS statement is:

[*label*] **EXITS** **[INHDR=***routinename* **]**
[,INTLR=*routinename* **]**
[,INREC=*routinename* **]**
[,OUTREC=*routinename* **]**

MEMBER Statement

The MEMBER statement is used to identify members to be printed or punched. All RECORD statements that follow a MEMBER statement pertain to the member indicated in that MEMBER statement only. When RECORD and MEMBER statements are used, at least one MEMBER statement must precede the first

RECORD statement. If no RECORD statement is used, the member is processed to standard specifications.

If no MEMBER statement appears, and a partitioned data set is being processed, all members of the data set are printed or punched. Any number of MEMBER statements can be included in a job step.

If the NAME parameter is specified in the MEMBER statement, MAXNAME must be specified in a PRINT or PUNCH statement.

The format of the MEMBER statement is:

[*label*] **MEMBER NAME=** {*membername* | *aliasname* }

RECORD Statement

The RECORD statement is used to define a group of records, called a *record group,* that is to be printed or punched to the user's specifications. A record group consists of any number of records to be edited identically.

If no RECORD statements appear, the entire data set, or named member, is printed or punched to standard specifications. If a RECORD statement is used, all data following the record group it defines (within a partitioned member or within an entire sequential data set) must be defined with other RECORD statements. Any number of RECORD statements can be included in a job step.

A RECORD statement referring to a partitioned data set for which no members have been named need contain only FIELD parameters. These are applied to the records in all members of the data set.

If a FIELD parameter is included in the RECORD statement, MAXFLDS must be specified in the PRINT or PUNCH statement.

If an IDENT parameter is included in the RECORD statement, MAXGPS must be specified in the PRINT or PUNCH statement. If a literal is specified in the IDENT parameter, MAXLITS must be specified in the PRINT or PUNCH statement.

The format of the RECORD statement is:

[*label*] **RECORD** [**IDENT=**(*length*,'*name*', *input-location*)]
[**,FIELD=**(*length*,[*input-location*],[*conversion*]
,[*output-location*])[**,FIELD=...**]

LABELS Statement

The LABELS statement specifies whether user labels are to be treated as data. For a detailed discussion of this option, refer to "Processing User Labels as Data," in "Appendix D: Processing User Labels."

Note: LABELS DATA=NO must be specified to make standard user label (SUL) exits inactive when an input data set with nonstandard labels (NSL) is to be processed.

If more than one valid LABELS statement is included, all but the last LABELS statement are ignored.

The format of the LABELS statement is:

[*label*] **LABELS** [**CONV** = {**PZ** | **XE** } |
,DATA = {<u>**YES**</u> | **NO** | **ALL** | **ONLY**}]

Operands	Applicable Control Statements	Description of Operands/Parameters
CDINCR	PUNCH	**CDINCR=*n*** specifies the increment to be used in generating sequence numbers. **Default:** 10 is the increment value.
CDSEQ	PUNCH	**CDSEQ=*n*** specifies the initial sequence number of a deck of punched cards. This value must be contained in columns 73 through 80. Sequence numbering is initialized for each member of a partitioned data set. If the value of *n* is zero, 00000000 is assumed as a starting sequence number. **Default:** Cards are not numbered.
CNTRL	PRINT	**CNTRL={*n* | 1}** specifies a control character for the output device that indicates line spacing, as follows: 1 indicates single spacing; 2 indicates double spacing; and 3 indicates triple spacing.
	PUNCH	specifies a control character for the output device that is used to select the stacker, as follows: 1 indicates the first stacker and 2 indicates the second stacker.
CONV	LABELS	**CONV={PZ | XE}** specifies a two-byte code that indicates the type of conversion to be performed on this field before it is printed or punched. The values that can be coded are: **PZ** specifies that data (packed decimal) is to be converted to unpacked decimal data. The converted portion of the input record (length L) occupies 2L - 1 output characters. **XE** specifies that data (alphameric) is to be converted to hexadecimal data. The converted portion of the input record (length L) occupies 2L output characters. **Default:** The field is moved to the output area without change.

Operands	Applicable Control Statements	Description of Operands/Parameters
DATA	LABELS	**DATA={YES | NO | ALL | ONLY}** specifies whether user labels are to be treated as data. The values that can be coded are: **YES** specifies that any user labels that are not rejected by a user's label processing routine are to be treated as data. Processing of labels as data stops in compliance with standard return codes. **NO** specifies that user labels are not to be treated as data. **ALL** specifies that user labels are to be treated as data regardless of any return code. A return code of 16 causes the utility to complete the processing of the remainder of the group of user labels and to terminate the job step. **ONLY** specifies that only user header labels are to be treated as data. User header labels are processed as data regardless of any return code. The job terminates upon return from the OPEN routine.

Operands	Applicable Control Statements	Description of Operands/Parameters
FIELD	RECORD	**FIELD=(***length* **,[***input-location* **],[***conversion* **],[***output-location* **])** **[,FIELD=...]**

specifies field-processing and editing information. These values can be coded.

length
specifies the length (in bytes) of the input field to be processed.

Note: The length must be equal to or less than the initial input LRECL.

input-location
specifies the starting byte of the input field to be processed.

Default: 1

Note: The sum of the length and the input location must be equal to or less than the input LRECL plus one.

conversion
specifies a two-byte code that indicates the type of conversion to be performed on this field before it is printed or punched. The values that can be coded are:

PZ
specifies that data (packed decimal) is to be converted to unpacked decimal data. The converted portion of the input record (length L) occupies 2L - 1 output characters when punching, and 2L output characters when printing.

XE
specifies that data (alphameric) is to be converted to hexadecimal data. The converted portion of the input record (length L) occupies 2L output characters.

Default: The field is moved to the output area without change.

output-location
specifies the starting location of this field in the output records. unspecified fields in the output records appear as blanks in the printed or punched output. Data that exceeds the SYSUT2 printer or punch size is not printed or punched. The specified fields may not exceed the logical output record length minus one. When specifying one or more FIELDs, the sum of all lengths and all extra characters needed for conversions must be equal to or less than the output LRECL minus one.

Default: 1

Operands	Applicable Control Statements	Description of Operands/Parameters
IDENT	RECORD	**IDENT**=*(length,'name',input-location)* identifies the last record of the record group to which the FIELD parameters apply. The values that can be coded are: *length* specifies the length (in bytes) of the field that contains the identifying name in the input records. The length cannot exceed eight bytes. *'name'* specifies the exact literal that identifies the last record of a record group. If the literal contains apostrophes, each must be written as two consecutive apostrophes. *input-location* specifies the starting location of the field that contains the identifying name in the input records. **Note:** The sum of the length and the input location must be equal to or less than the input LRECL plus one. **Default:** If IDENT is omitted and STOPAFT is not included with the PRINT or PUNCH statement, record processing halts after the last record in the data set. If IDENT is omitted and STOPAFT is included with the PRINT or PUNCH statement, record processing halts when the STOPAFT count is satisfied or after the last record of the data set is processed, whichever occurs first.
INHDR	EXITS	**INHDR**=*routinename* specifies the symbolic name of a routine that processes user input header labels.
INITPG	PRINT	**INITPG**=*n* specifies the initial page number; the pages are numbered sequentially thereafter. The INITPG parameter must not exceed a value of 9999. **Default:** 1
INREC	EXITS	**INREC**=*routinename* specifies the symbolic name of a routine that manipulates each logical record (or physical block in the case of VS or VBS records longer than 32K bytes) before it is processed.
INTLR	EXITS	**INTLR**=*routinename* specifies the symbolic name of a routine that processes user input trailer labels.

Operands	Applicable Control Statements	Description of Operands/Parameters
ITEM	TITLE	**ITEM=**('*title*'[,*output-location*])[,**ITEM...**] specifies title or subtitle information. The values that can be coded are: '*title*' specifies the title or subtitle literal (maximum length of 40 bytes), enclosed in apostrophes. If the literal contains apostrophes, each apostrophe must be written as two consecutive apostrophes. *output-location* specifies the starting position at which the literal for this item is to be placed in the output record. The specified title may not exceed the output logical record length minus one. **Default:** 1
MAXFLDS	PRINT PUNCH	**MAXFLDS=***n* specifies a number no less than the total number of FIELD parameters appearing in subsequent RECORD statements. The value must not exceed 32,767. **Default:** If MAXFLDS is omitted when there is a FIELD parameter present, the print or punch request is terminated.
MAXGPS	PRINT PUNCH	**MAXGPS=***n* specifies a number no less than the total number of IDENT parameters appearing in subsequent RECORD statements. The value must not exceed 32,767. **Default:** If MAXGPS is omitted when there is an IDENT parameter present, the print or punch request is terminated.
MAXLINE	PRINT	**MAXLINE=***n* specifies the maximum number of lines to a printed page. Spaces, titles, and subtitles are included in this number. **Default:** 60
MAXLITS	PRINT PUNCH	**MAXLITS=***n* specifies a number no less than the total number of characters contained in the IDENT literals of subsequent RECORD statements. The value must not exceed 32,767. **Default:** If MAXLITS is omitted when there is a literal present, the print or punch request is terminated.
MAXNAME	PRINT PUNCH	**MAXNAME=***n* specifies a number no less than the total number of subsequent MEMBER statements. The value must not exceed 32,767. **Default:** If MAXNAME is omitted when there is a MEMBER statement present, the print or punch request is terminated.

Operands	Applicable Control Statements	Description of Operands/Parameters
NAME	MEMBER	NAME={*membername* | *aliasname*} specifies a member to be printed or punched. These values can be coded: *membername* specifies a member by its member name. *aliasname* specifies a member by its alias.
OUTREC	EXITS	OUTREC=*routinename* specifies the symbolic name of a routine that manipulates each logical record (or physical block in the case of VS or VBS records longer than 32K bytes) before it is printed or punched.
PREFORM	PRINT PUNCH	**PREFORM={A | M}** specifies that a control character is provided as the first character of each record to be printed or punched. The control characters are used to control the spacing, number of lines per page, page ejection, and selecting a stacker. That is, the output has been previously formatted, and the "standard specifications" are superseded. If an error occurs, the print/punch operation is terminated. If PREFORM is coded, any additional PRINT or PUNCH operands and all other control statements, except for syntax checking, LABELS statements and TYPORG operands, are ignored. PREFORM must not be used for printing or punching data sets with VS or VBS records longer than 32K bytes. These values can be coded: **A** specifies that an ASA control character is provided as the first character of each record to be printed or punched. If the input record length exceeds the output record length, the utility uses the ASA character for printing the first line, with a single space character on all subsequent lines of the record (for PRINT), and duplicates the ASA character on each output card of the record (for PUNCH). **M** specifies that a machine-code control character is provided as the first character of each record to be printed or punched. If the input record length exceeds the output record length, the utility prints all lines of the record with a *print-skip-one-line* character until the last line of the record, which will contain the actual character provided as input (for PRINT), and duplicates the machine control character on each output card of the record (for PUNCH).
SKIP	PRINT PUNCH	**SKIP=*n*** specifies that every *n*th record (or physical block in the case of VS or VBS records longer than 32K bytes) is to be printed or punched. **Default:** Successive logical records are printed or punched.

Operands	**Applicable Control Statements**	**Description of Operands/Parameters**
STOPAFT	PRINT PUNCH	**STOPAFT=*n*** specifies, for sequential data sets, the number of logical records (or physical blocks in the case of VS or VBS records longer than 32K bytes) to be printed or punched. For partitioned data sets, this specifies the number of logical records (or physical blocks in the case of VS or VBS records longer than 32K bytes) to be printed or punched in each member to be processed. The *n* value must not exceed 32,767. If STOPAFT is specified and RECORD statements are present, the operation is terminated when the STOPAFT count is satisfied or at the end of the first record group, whichever occurs first.
STRTAFT	PRINT PUNCH	**STRTAFT=*n*** specifies, for sequential data sets, the number of logical records (physical blocks in the case of VS or VBS type records longer than 32K bytes) to be skipped before printing or punching begins. For partitioned data sets, STRTAFT=*n* specifies the number of logical records to be skipped in each member before printing or punching begins. The *n* value must not exceed 32,767. If STRTAFT is specified and RECORD statements are present, the first RECORD statement of a member describes the format of the first logical record to be printed or punched.
TOTCONV	PRINT PUNCH	**TOTCONV={XE | PZ}** specifies the representation of data to be printed or punched. TOTCONV can be overridden by any user specifications (RECORD statements) that pertain to the same data. These values can be coded: **XE** specifies that data is to be punched in 2-character-per-byte hexadecimal representation (for example, C3 40 F4 F6). If XE is not specified, data is punched in 1-character per byte alphameric representation. The above example would appear as C 46. **PZ** specifies that data (packed decimal mode) is to be converted to unpacked decimal mode. IEBPTPCH does not check for packed decimal mode. The output is unpredictable when the input is other than packed decimal. **Default:** If TOTCONV is omitted, data is not converted.

Operands	Applicable Control Statements	Description of Operands/Parameters
TYPORG	PRINT PUNCH	**TYPORG={PS | PO}** specifies the organization of the input data set. These values can be coded: **PS** specifies that the input data set is organized sequentially. **PO** specifies that the input data set is partitioned.

Restrictions

- The SYSPRINT DD statement is required for each use of IEBPTPCH. The RECFM is always FBA, the LRECL is always 121. Output can be blocked by specifying a block size that is a multiple of 121 on the SYSPRINT DD statement. The default block size is 121.
- The SYSUT1 DD statement is required for each use of IEBPTPCH. The RECFM (except for undefined records), BLKSIZE, and LRECL (except for undefined and fixed unblocked records) must be present on the DD statement, in the DSCB, or on the tape label.
- The SYSUT2 DD statement is required every time IEBPTPCH is used. The RECFM is always FBA or FBM. The LRECL parameter, or, if no logical record length is specified, the BLKSIZE parameter, specifies the number of characters to be written per printed line or per punched card (this count includes a control character). The number of characters specified must be in the range of 2 through 145. The default values for edited output lines are 121 characters per printed line and 81 characters per punched card. The SYSUT2 data set can be blocked by specifying both the LRECL and the BLKSIZE parameters, in which case, block size must be a multiple of logical record length.
- The SYSIN DD statement is required for each use of IEBPTPCH. The RECFM is always FB, the LRECL is always 80. Any blocking factor that is a multiple of 80 can be specified for the BLKSIZE. The default block size is 80.
- A partitioned directory to be printed or punched must be defined as a sequential data set (TYPORG=PS). You must specify RECFM=U, BLKSIZE=256, and LRECL=256 on the SYSUT1 DD statement.

IEBPTPCH Examples

The following examples illustrate some of the uses of IEBPTPCH. Figure 11-3 can be used as a quick reference guide to IEBPTPCH examples. The numbers in the "Example" column point to the examples that follow:

Operation	Data Set Organization	Devices	Comments	Example
PRINT	Sequential	9-track Tape and System Printer	Standard format. Conversion to hexadecimal.	1
PUNCH	Sequential	7-track Tape and Card Reader	Standard format. Conversion to hexadecimal.	2
PRINT	Partitioned	Disk and System Printer	Standard format. Conversion to hexadecimal. Ten records from each member are to be printed.	3
PRINT	Partitioned	Disk and System Printer	Standard format. Conversion to hexadecimal. Two members are to be printed.	4
PRINT	Sequential	9-track Tape and System Printer	User-specified format. Input data set is the second data set on the volume.	5

Figure 11-3 (Part 1 of 2). IEBPTPCH Example Directory

Operation	Data Set Organization	Devices	Comments	Example
PUNCH	Sequential	Disk and Card Reader Punch	User-specified format. Sequence numbers are to be assigned and punched.	6
PRINT	Sequential, Partitioned	Disk and System Printer	Standard format. Conversion to hexadecimal.	7
PUNCH	Sequential	Card Reader and Card Read Punch	Standard format. Control data set is a member in a cataloged partitioned data set.	8
PRINT	Sequential	Disk and System Printer	User-specified format. User routines are provided. Processing ends after the third record group is printed or STOPAFT is satisfied.	9
PRINT	Sequential	9-track tape and System Printer	SYSOUT format. SYSOUT data set is on tape volume.	10

Figure 11-3 (Part 2 of 2). IEBPTPCH Example Directory

Note: Examples which use *disk* or *tape*, in place of actual device-ids, must be changed before use. See the Device Support section, in the Introduction to this manual, for valid device-id notation.

IEBPTPCH Example 1

In this example, a sequential data set is to be printed according to standard specifications. The input data set resides on a tape volume. The printed output is to be converted to hexadecimal.

```
//PRINT    JOB  09#660,SMITH
//         EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=tape,LABEL=(,NL),VOLUME=SER=001234,
// DISP=(OLD,KEEP),DCB=(RECFM=U,BLKSIZE=2000)
//SYSUT2   DD   SYSOUT=A
//SYSIN    DD   *
     PRINT   TOTCONV=XE
     TITLE   ITEM=('PRINT SEQ DATA SET WITH CONV TO HEX',10)
/*
```

The control statements are discussed below.

- SYSUT1 DD defines the input data set. The data set contains undefined records; no record is larger than 2,000 bytes.
- SYSUT2 DD defines the output data set. The data set is written to the system output device (printer assumed). Each printed line contains groups (8 characters each) of hexadecimal information. Each record begins a new line of printed output.
- SYSIN DD defines the control data set, which follows in the input stream. The control data set contains the PRINT and TITLE statements.
- PRINT initiates the print operation and specifies conversion from alphameric to hexadecimal representation.
- TITLE specifies a title to be placed beginning in column 10 of the printed output. The title is not converted to hexadecimal.

IEBPTPCH Example 2

In this example, a sequential data set is to be punched according to standard specifications. The input data set resides on a tape volume. The punched output is converted to hexadecimal.

```
//PUNCHSET JOB  09#660,SMITH
//          EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=INSET,UNIT=tape,VOLUME=SER=001234,
// LABEL=(,NL),DISP=(OLD,KEEP),DCB=(RECFM=FB,
// LRECL=80,BLKSIZE=2000)
//SYSUT2   DD   SYSOUT=B
//SYSIN    DD   *
      PUNCH    TOTCONV=XE
      TITLE    ITEM=('PUNCH SEQ DATA SET WITH CONV TO HEX',10)
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set. The data set contains 80-byte, fixed blocked records.
- SYSUT2 DD defines the system output class (punch is assumed). Each record from the input data set is represented by two punched cards.
- SYSIN DD defines the control data set, which follows in the input stream. The control data set contains the PUNCH and TITLE statements.
- PUNCH initiates the punch operation and specifies conversion from alphameric to hexadecimal representation.
- TITLE specifies a title to be placed beginning in column 10. The title is not converted to hexadecimal.

IEBPTPCH Example 3

In this example, a partitioned data set (ten records from each member) is to be printed according to standard specifications. The input data set resides on a disk volume. The printed output is converted to hexadecimal.

```
//PRINTPDS JOB  09#660,SMITH
//          EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=PDS,UNIT=disk,DISP=(OLD,KEEP),
// VOLUME=SER=111112
//SYSUT2   DD   SYSOUT=A
//SYSIN    DD   *
        PRINT   TOTCONV=XE,TYPORG=PO,STOPAFT=10
        TITLE   ITEM=('PRINT PDS - 10 RECS EACH MEM',20)
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set.
- SYSUT2 DD defines the output data set on the system output device (printer assumed). Each printed line contains groups (8 characters each) of hexadecimal information. Each record begins a new line of printed output. The size of the record determines how many lines of printed output are required per record.
- SYSIN DD defines the control data set, which follows in the input stream. The control data set contains the PRINT and TITLE statements.

- **PRINT** initiates the print operation, specifies conversion from alphameric to hexadecimal representation, indicates that the input data set is partitioned, and specifies that ten records from each member are to be printed.
- **TITLE** specifies a title to be placed beginning in column 20 of the printed output. The title is not converted to hexadecimal.

IEBPTPCH Example 4

In this example, two partitioned members are to be printed according to standard specifications. The input data set resides on a disk volume. The printed output is to be converted to hexadecimal.

```
//PRNTMEMS JOB  09#660,SMITH
//          EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=PDS,DISP=(OLD,KEEP),VOLUME=SER=111112,
//              UNIT=disk
//SYSUT2   DD   SYSOUT=A
//SYSIN    DD   *
        PRINT   TYPORG=PO,TOTCONV=XE,MAXNAME=2
        TITLE   ITEM=('PRINT TWO MEMBS WITH CONV TO HEX',10)
       MEMBER   NAME=MEMBER1
       MEMBER   NAME=MEMBER2
/*
```

The control statements are discussed below:

- **SYSUT1 DD** defines the input data set.
- **SYSUT2 DD** defines the output data set on the system output device (printer assumed). Each printed line contains groups (8 characters each) of hexadecimal information. Each record begins a new line of printed output.
- **SYSIN DD** defines the control data set, which follows in the input stream. The control data set contains PRINT, TITLE, and MEMBER statements.
- **PRINT** initiates the print operation, indicates that the input data set is partitioned, specifies conversion from alphameric to hexadecimal representation, and indicates that two MEMBER statements appear in the control data set.
- **TITLE** specifies a title to be placed beginning in column 10 of the printed output. The title is not converted to hexadecimal.
- **MEMBER** specifies the member names of the members to be printed.

IEBPTPCH Example 5

In this example, a sequential data set is to be printed according to user specifications. The input data set is the second data set on a tape volume.

```
//PTNONSTD JOB  09#660,SMITH
//          EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=SEQSET,UNIT=tape,LABEL=(2,SUL),
// DISP=(OLD,KEEP),VOLUME=SER=001234
//SYSUT2   DD   SYSOUT=A
//SYSIN    DD   *
        PRINT   MAXFLDS=1
        EXITS   INHDR=HDRIN,INTLR=TRLIN
       RECORD   FIELD=(80)
       LABELS   DATA=YES
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set.
- SYSUT2 DD defines the output data set on the system output device (printer assumed). Each printed line contains 80 contiguous characters (one record) of information.
- SYSIN DD defines the control data set, which follows in the input stream. The control data set contains the PRINT, RECORD, EXITS, and LABELS statements.
- PRINT initiates the print operation and indicates that one FIELD parameter is included in a subsequent RECORD statement.
- RECORD indicates that each input record is to be processed in its entirety (80 bytes). Each input record is printed in columns 1 through 80 on the printer.
- LABELS specifies that user header and trailer labels are to be printed according to the return code issued by the user exits.
- EXITS indicates that exits will be taken to user header-label and trailer-label processing routines when these labels are encountered on the SYSUT1 data set.

IEBPTPCH Example 6

In this example, a sequential data set is to be punched according to user specifications. The input data set resides on a disk volume.

```
//PHSEQNO  JOB  09#660,SMITH
//           EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=SEQSET,UNIT=disk,LABEL=(,SUL),
// VOLUME=SER=111112,DISP=(OLD,KEEP),DCB=(RECFM=FB,
// LRECL=80,BLKSIZE=2000)
//SYSUT2   DD   SYSOUT=B
//SYSIN    DD   *
          PUNCH    MAXFLDS=1,CDSEQ=00000000,CDINCR=20
         RECORD    FIELD=(72)
         LABELS    DATA=YES
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set. The data set contains 80-byte, fixed blocked records.
- SYSUT2 DD defines the system output class (punch is assumed). Each record from the input data set is represented by one punched card.
- SYSIN DD defines the control data set, which follows in the input stream. The control data set contains the PUNCH, RECORD, and LABELS statements.
- PUNCH initiates the punch operation, indicates that one FIELD parameter is included in a subsequent RECORD statement, and assigns a sequence number for the first punched card (00000000) and an increment value for successive sequence numbers (20). Sequence numbers are placed in columns 73 through 80 of the output records.
- RECORD indicates that bytes 1 through 72 of the input records are to be punched. Bytes 73 through 80 of the input records are replaced by the new sequence numbers in the output card deck.
- LABELS specifies that user header labels and user trailer labels are to be punched.

Labels cannot be edited; they are always moved to the first 80 bytes of the output buffer. In this example, no sequence numbers are present on the cards containing user header and user trailer records.

IEBPTPCH Example 7

In this example, the directory of a partitioned data set is to be printed. The input data set resides on a disk volume. The printed output is to be converted to hexadecimal.

```
//PRINTDIR JOB  09#660,SMITH
//          EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1    DD   DSNAME=PDS,UNIT=disk,VOLUME=SER=111112,
// DISP=(OLD,KEEP),DCB=(RECFM=U,BLKSIZE=256)
//SYSUT2    DD   SYSOUT=A
//SYSIN     DD   *
   PRINT    TYPORG=PS,TOTCONV=XE
   TITLE    ITEM=('PRINT PARTITIONED DIRECTORY OF PDS',10)
   TITLE    ITEM=('FIRST TWO BYTES SHOW NUM OF USED BYTES',10)
  LABELS    DATA=NO
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set (the partitioned directory).
- SYSUT2 DD defines the output data set on the system output device (printer assumed). Each printed line contains groups (8 characters each) of hexadecimal information. Six lines of print are required for each record. Each record begins a new line of printed output.
- SYSIN DD defines the control data set, which follows in the input stream. The control data set contains the PRINT, TITLE, and LABELS statements.
- PRINT initiates the print operation, indicates that the partitioned directory is organized sequentially, and specifies conversion from alphameric to hexadecimal representation.
- The first TITLE statement specifies a title, which is not converted to hexadecimal.
- The second TITLE statement specifies a subtitle, which is not converted to hexadecimal.
- LABELS specifies that no user labels are to be printed.

Note: Not all of the bytes in a directory block need contain data pertaining to the partitioned data set; unused bytes are sometimes used by the operating system as temporary work areas. The first four characters of printed output indicate how many bytes of the 256-byte block pertain to the partitioned data set. Any unused bytes occur in the latter portion of the directory block; they are not interspersed with the used bytes.

IEBPTPCH Example 8

In this example, a card deck containing valid punch card code or BCD is to be duplicated. The input card deck resides in the input stream.

```
//PUNCH      JOB   09#660,SMITH
//           EXEC  PGM=IEBPTPCH
//SYSPRINT   DD    SYSOUT=A
//SYSIN      DD    DSNAME=PDSLIB(PNCHSTMT),DISP=(OLD,KEEP)
//SYSUT2     DD    SYSOUT=B
//SYSUT1     DD    DATA
```

(input card data set including // cards)

```
/*
```

The control statements are discussed below:

- SYSIN DD defines the control data set. The control data set contains a PUNCH statement and is defined as a member of the partitioned data set PDSLIB. (The data set is cataloged.) The RECFM must be FB and the LRECL must be 80.
- SYSUT2 DD defines the system output class (punch is assumed).
- SYSUT1 DD defines the input card data set, which follows in the input stream.

IEBPTPCH Example 9

In this example, three record groups are to be printed. A user routine is provided to manipulate output records before they are printed.

```
                                                            72
//PRINT      JOB   09#660,SMITH
//           EXEC  PGM=IEBPTPCH
//SYSPRINT   DD    SYSOUT=A
//SYSUT1     DD    DSNAME=SEQDS,UNIT=disk,DISP=(OLD,KEEP),
// LABEL=(,SUL),VOLUME=SER=111112
//SYSUT2     DD    SYSOUT=A
//SYSIN      DD    *
        PRINT      MAXFLDS=9,MAXGPS=9,MAXLITS=23,STOPAFT=32767
      TITLE      ITEM=('TIMECONV-DEPT D06),ITEM=(JAN10-17',80)
        EXITS      OUTREC=NEWTIME,INHDR=HDRS,INTLR=TLRS
       RECORD      IDENT=(6,'498414',1),                         C
                  FIELD=(8,1,,10),FIELD=(30,9,XE,20)
       RECORD      IDENT=(2,'**',39),                            C
                  FIELD=(8,1,,10),FIELD=(30,9,XE,20)
       RECORD      IDENT=(6,'498414',1),                         C
                  FIELD=(8,1,,10),FIELD=(30,9,XE,20)
       LABELS      CONV=XE,DATA=ALL
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set. The data set resides on a disk volume.
- SYSUT2 DD defines the output data set on the system output device (printer assumed).
- SYSIN DD defines the control data set, which follows in the input stream. The control data set contains the PRINT, TITLE, EXITS, and RECORD statements.
- The PRINT statement: (1) initializes the print operation, (2) indicates that not more than nine FIELD parameters are included in subsequent RECORD statements, (3) indicates that not more than nine IDENT parameters are included in a subsequent RECORD statement, (4) indicates that not more than 23 literal characters are included in the subsequent IDENT parameter, and (5) indicates that processing is to be terminated after 32,767 records are processed

or after the third record group is processed, whichever comes first. Because MAXLINE is omitted, 60 lines are printed on each page.

- TITLE specifies a title.
- EXITS specifies the name of a user routine (NEWTIME), which is used to manipulate output records before they are printed.
- The first RECORD statement defines the first record group to be processed and indicates where information from the input records is to be placed in the output records. Bytes 1 through 8 of the input records appear in columns 10 through 17 of the printed output, and bytes 9 through 38 are printed in hexadecimal representation and placed in columns 20 through 79.
- The second RECORD statement defines the second group to be processed. The parameter in the IDENT operand specifies that an input record containing the two characters ** in positions 39 and 40 is to be the last record edited according to the FIELD operand in this RECORD statement. The FIELD operand specifies that bytes 1 through 8 of the input records are to be printed in hexadecimal representation and placed in columns 2 through 17 of the printed output, and bytes 9 through 38 are to appear in columns 20 through 49.
- The third and last RECORD statement is equal to the first RECORD statement. An input record that meets the parameter in the IDENT operand ends processing, unless the STOPAFT parameter in the PRINT statement has not already done so.
- LABELS specifies that all user header or trailer labels are to be printed regardless of any return code, except 16, issued by the user's exit routine. It also indicates that the labels are to be converted from alphameric to hexadecimal representation.

IEBPTPCH Example 10

In this example, the input is a SYSOUT (sequential) data set, which was previously written as the second data set of a standard label tape. It is to be printed in SYSOUT format.

```
//PTSYSOUT JOB  09#660,SMITH
//          EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=tape,LABEL=(2,SL),DSNAME=LISTING,
//  DISP=(OLD,KEEP),VOL=SER=001234
//SYSUT2   DD   SYSOUT=A
//SYSIN    DD   *
   PRINT   PREFORM=A
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input data set. It is the second data set of a standard label tape, which has been assigned the name LISTING.
- SYSUT2 DD defines the output data set on the system output device (printer assumed).
- SYSIN DD defines the control data set, which follows in the input stream. The control data set contains the PRINT statement.
- The PRINT statement initiates the print operation and indicates that an ASA control character is provided as the first character of each record to be printed.

IEBTCRIN PROGRAM

IEBTCRIN is a data set utility used to read input from the IBM 2495 Tape Cartridge Reader (TCR), edit the data as specified by the user, and produce a sequentially organized output data set.

IEBTCRIN can be used to construct records from the stream of data bytes read sequentially from the Tape Cartridge Reader. The user has the option of gaining temporary control (via a user-supplied exit routine) to process each logical record.

The input to IEBTCRIN is in the form of cartridges written by either the IBM Magnetic Tape SELECTRIC Typewriter (MTST) or the IBM 50 Magnetic Data Inscriber (MTDI). An input data set (one or more cartridges) must consist of either all MTST cartridges or all MTDI cartridges. (For more information concerning the MTDI use and an explanation of terminology used in this chapter, refer to *IBM 50 Magnetic Data Inscriber Component Description.*)

When MTDI input is edited, IEBTCRIN maintains information about each record as it is being edited. This information is summarized in the Error Description Word (EDW) which is described later. When the EDW contains a value other than zero in either the level status (byte 0) or the type status (byte 1), the record is considered an error record by the program and the EDW is appended to the start of the record to aid the user in analyzing the error.

MTDI Editing Criteria

The cartridges created on the IBM 50 Magnetic Data Inscriber contain a continuous stream of data bytes (that is, there are no interblock gaps). Therefore, when editing is specified, IEBTCRIN extracts records one at a time from the data stream. To accomplish this, IEBTCRIN scans for control codes written by MTDI. IEBTCRIN uses start-of-record (SOR) and end-of-record (EOR) locations to extract MTDI records from the input stream.

The (SOR) location is defined as:

- The location of the first character on a cartridge.
- The location of the first character after the previous record's (EOR) location.
- The location of an SOR code.
- The location of a group separator (GS) code.

The character in the SOR location is checked to determine if it is a valid start-of-record character. A P1 through P8, a cancel code, or a GS code are valid start-of-record characters; all others are invalid.

The EOR location by priority sequence is:

1. The same location as the SOR location, if the SOR character was a valid GS code.
2. The location of the first encountered record mark (RM) or verify okay (VOK) code if that location is within the length of the maximum user-specified record size.
3. The location of any code preceding either a valid SOR code or the end-of-media (EOM) code, if that location is within the length of the maximum user-specified record size.

4. The location determined in 2 or 3, regardless of the maximum user-specified record size if the SOR location contains a cancel code.
5. If one of the previous EOR locations cannot be defined, an EOR condition will be forced at the location where the record length equals the maximum user-specified record size.

The character in the EOR location is checked to determine if it is a valid end-of-record character. Valid EOR characters are the GS character (if the SOR character was a GS code) and VOK or RM codes; all others are invalid. Each GS code is considered a valid SOR code or EOR code and will be bypassed.

MTDI Editing Restrictions

Following are the restrictions that apply when editing MTDI records:

- All canceled records are bypassed; they are not passed to any exit routines or written on any data sets. The level status is set to 0.
- All input records less than three bytes in length (SOR location, one data byte, and EOR location) are treated as canceled records. The remaining portion of a record that was longer than the user-specified maximum record size can result in an input record of this size.
- Data duplication is accomplished by replacing the DUP (duplication) code with the character from the corresponding location of the previous record.
- The record used for data duplication is the record returned from any user exits.
- GS codes will not affect the level status or duplication of following records.
- Data duplication does not occur for any of the following conditions:
  1. The DUP code is encountered in the first record of a cartridge.
  2. The DUP code is encountered in a record immediately following a canceled record. A canceled record is one that contains a cancel code in the SOR location or an input record of less than three bytes as described above.
  3. The DUP code is encountered in a position that would cause duplication of a position beyond the last data byte of the previous record.
  4. The DUP code is encountered in a position that would cause duplication of an error-replace character.

  In each case, the DUP code is replaced with the user specified error-replace character, and a field error is indicated.
- Left-zero justification does not occur; the left-zero fill code (LZ) is replaced with the user-specified error-replace character and a field error is indicated for either of the following conditions:
  1. The left-zero fill code (LZ) is encountered without first having encountered its corresponding left-zero start code (LZS).
  2. The user-specified maximum record size is exceeded before encountering the valid end of a left-zero field.

If MTDI is edited, an EDW which is four bytes long is appended to the front of each error record describing the error condition. For further definition of the EDW, see "Error Records" earlier in this chapter. If the SYSUT3 DD statement specified variable length records, an RDW which is four bytes long is also appended to the front of the record. For further description of the RDW, see *OS/VS2 Supervisor Services and Macro Instructions.*

The user-supplied routines specified in ERROR and OUTREC can be used to examine and modify any byte in the record or EDW. The record length can be changed, subject to the following restrictions:

- A work area used to construct the records is allocated by the program equal in size to the largest of (1) MAXLN, (2) LRECL on SYSUT2, or (3) LRECL on SYSUT3.
- The record length must not be increased beyond this size. Overlaying of other work areas may then occur, causing unpredictable results.

The new record length must be placed in the location pointed to by the second parameter word as received at entry to the routine. This length must include the EDW and RDW (if applicable). It is not necessary to modify the RDW because it is re-created if the record is to be written by IEBTCRIN. However, if the user does his own output from this routine, he must ensure that the RDW is correct for the record.

If IEBTCRIN is to write the record, the length of the output record depends on the RECFM specification, as follows:

- Fixed and variable records may have a maximum length equal to LRECL. Records larger than this are truncated.
- Undefined records may have a maximum length equal to BLKSIZE. Records larger than this are truncated.

These record lengths include the EDW and RDW, where applicable.

The record length returned from the error exit is used to establish the location of the last data byte in the record. The location is used to control data duplication in the following record. However, it is not used for checking the record length of subsequent records.

Modifications to the EDW, record, or record length may affect the editing of subsequent records. If the input is not edited, the user can examine and modify any byte in the record. The record length can also be changed, subject to the MTDI-editing restrictions.

If STDUC, STDLC, or *name* is specified, certain of the MTST codes are processed in a special way before translation. Feed codes (FD), switch codes (SW), and autosearch codes (AS), both uppercase and lowercase, are deleted from the data. Each 61-character reference code is reduced to a single search code (SRC).

A stop code, whether uppercase (ST) or lowercase (st), indicates that all data on a cartridge has been read. Therefore, when an MTST cartridge to be processed by IEBTCRIN is created, the user must not use a stop code for any purpose other than signaling end-of-data on the cartridge. Stop codes within meaningful data cause any subsequent data on the cartridge to be lost because the cartridge is rewound and unloaded when a stop code is encountered.

If EDITD or EDITR is specified, the edit consists of the following functions:

- Records are extracted one at a time from the input buffers by scanning for the record-delimiting codes (SOR and EOR).
- DUP codes are replaced with the character from the corresponding location in the preceding record.
- Left-zero fields are right aligned and leading zeros are inserted where necessary.
- Left-zero start codes are deleted from the records.

- Group separator codes and records that start with cancel record codes are bypassed.

For MTDI input with editing specified, MAXLN is used to specify in bytes the length of the longest valid record after editing. If the program encounters a record in which a valid end-of-record cannot be determined within this length, an end-of-record condition is forced and the record is considered an error record.

The values that can be specified for MINLN and MAXLN are:

- For MTST processing or MTDI processing without editing, MINLN is not specified. MAXLN should equal the number of bytes to be passed as a record.
- For MTDI processing when EDIT=EDITD, MINLN should equal the number of bytes in the shortest valid record after editing, excluding SOR and EOR codes. MAXLN should equal the number of bytes in the longest valid record after editing, excluding SOR and EOR codes.
- For MTDI processing when EDIT=EDITR, MINLN should equal the number of bytes in the shortest valid record after editing, including SOR and EOR codes. MAXLN should equal the number of bytes in the longest valid record after editing, including SOR and EOR codes.

Note: The values for MINLN and MAXLN should not include the four bytes long record descriptor word added to a variable length record.

Figure 12-1 shows the hexadecimal characters representing special purpose codes that must not be used as replacement bytes.

MTDI Codes

X‘00’	(LZ)	X‘1E’	(VOK)	X‘74’	(P4)
X‘11’	(DUP)	X‘3C’	(RM)	X‘75’	(P5)
X‘12’	(LZS)	X‘71’	(P1)	X‘76’	(P6)
X‘18’	(CAN)	X‘72’	(P2)	X‘77’	(P7)
X‘1D’	(GS)	X‘73’	(P3)	X‘78’	(P8)

MTST Codes

X‘10’	(cr)	X‘14’	(CR)	X‘51’	(as)
X‘11’	(sw)	X‘15’	(SW)	X‘55’	(AS)
X‘13’	(fd)	X‘17’	(FD)	X‘80’	(src)
				X‘81 through X‘FF’	

Figure 12-1. Special Purpose Codes

The special purpose codes listed in Figure 12-1 are used by IEBTCRIN when constructing records. Use of these codes causes a message to be issued and the utility to be terminated.

Figure 12-2 shows the values that can be chosen to replace error bytes for MTDI input.

Figure 12-3 shows the values that can be chosen to replace error bytes for MTST input.

Figure 12-4 shows MTST codes after they have been translated by IEBTCRIN when TRANS=STDLC is specified.

Bit Positions 0,1		00				01				10				11			
Bit Positions 2, 3		00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
First Hexadecimal Digit		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Bit Positions 4, 5, 6, 7	Second Hexadecimal Digit																
0000	0	LZ				SP	&								0	082	0
0001	1		DUP					/	P1					A	J		1
0010	2		LZS						P2					B	K	S	2
0011	3								P3					C	L	T	3
0100	4								P4					D	M	U	4
0101	5								P5					E	N	V	5
0110	6								P6					F	O	W	6
0111	7								P7					G	P	X	7
1000	8		CAN						P8					H	Q	Y	8
1001	9		ED											I	R	Z	9
1010	A					¢	!		:								
1011	B					.	$	,	#								
1100	C				RM	<	*	%	@								
1101	D		GS			(	)	-	/								
1110	E		VOK			+	;	>	=								
1111	F					|	¬	?	"								

Special Control:

LZ = Left zero fill
DUP = Duplicate
LZS = Left zero start
ED = End data
GS = Group Separator

Start of Record (SOR):

P1 = Program level 1
P2 = Program level 2
P3 = Program level 3
P4 = Program level 4
P5 = Program level 5
P6 = Program level 6
P7 = Program level 7
P8 = Program level 8
CAN = Cancel

End of Record (EOR):

RM = Record mark
VOK = Verify OK

This figure represents the character set and control codes as read from an MTDI created cartridge.

Figure 12-2. MTDI Codes from TCR

Bit Positions 4, 5, 6, 7	Second Hexadecimal Digit	00				01				10				11				Bit Positions 0,1
		00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11	Bit Positions 2, 3
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	First Hexadecimal Digit
0000	0	z	cr	5	0	l	tab	'	s	src								
0001	1	2	sw	6	9	.	as	i	w									
0010	2	t		e	h	j	sp	p	y									
0011	3	n	fd	k	b	=		q	-									
0100	4	Z	CR	%	)	°	TAB	"	S	SRC								
0101	5	@	SW	¢	(	●	AS	I	W									
0110	6	T		E	H	J	SP	P	Y									
0111	7	N	FD	K	B	+		Q	-									
1000	8	1		7	4	m	bsp	r	o									
1001	9	3	st	8		v		a										
1010	A	x		d	l	g		:	/									
1011	B	u		c		f	stx	,										
1100	C	±		&	$	M	BSP	R	O									
1101	D	#	ST	*		V		A										
1110	E	X		D	L	G		:	?									
1111	F	U		C		F	STX	,										

cr and CR = Carrier return code
sw and SW = Switch code
fd and FD = Feed code
st and ST = stop code
tab and TAB = Tab code
as and AS = Automatic search
sp and SP = Space
bsp and BSP = Backspace
stx and STX = Stop transfer
src and SRC = Search

This figure represents the character set and control codes as read from an MTST created cartridge.

Figure 12-3. MTST Codes from TCR

Bit Positions 4, 5, 6, 7	Second Hexadecimal Digit	00				01				10				11				Bit Positions 0,1
		00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11	Bit Positions 2, 3
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	First Hexadecimal Digit
0000	0					SP	&										0	
0001	1							/		a	j	°		A	J		1	
0010	2			STX						b	k	s		B	K	S	2	
0011	3									c	l	t		C	L	T	3	
0100	4									d	m	u		D	M	U	4	
0101	5	TAB								e	n	v		E	N	V	5	
0110	6		BSP							f	o	w		F	O	W	6	
0111	7									g	p	x		G	P	X	7	
1000	8									h	q	y		H	Q	Y	8	
1001	9									i	r	z		I	R	Z	9	
1010	A					¢	!		:									
1011	B					.	$	,	#									
1100	C						*	%	@									
1101	D	CR				(	)	_	'									
1110	E		SRC			+	;		=		±							
1111	F							?	"									

TAB = Tab code
CR = Carrier return
BSP = Backspace
SRC = Search
STX = Stop transfer
SP = Space

Note: The STDUC option permits translating both lowercase and uppercase alphabetic characters to uppercase.

Figure 12-4. MTST Codes after Translation by IEBTCRIN with TRANS=STDCL

End-of-Cartridge

Unique codes, written by the MTST or the MTDI device, signal the program when all data on a cartridge has been read. For MTST cartridges, this end-of-cartridge code is a lowercase stop code (st) or an uppercase stop code (ST). For MTDI cartridges, the end-of-cartridge code is the end-data code (ED).

IEBTCRIN terminates input from a cartridge upon encountering the end-of-cartridge code and rewinds the cartridge. IEBTCRIN continues to process cartridges until end-of-file is encountered.

End-of-file is signaled following a rewind operation when there are no more cartridges in the feed hopper, the END OF FILE button is pressed, and end-of-cartridge for the last cartridge is recognized. An end-of-file indication will be passed to the OUTREC and/or ERROR exits if specified by setting register 1 equal to 0.

Error Records

If a record is found to be in error, the record is passed to the user error exit routine if one is specified. If an error exit is not specified, the action to be taken is determined by the option specified in a utility control statement.

When either MTST input or MTDI input without editing is specified, the only error that can be recognized is a record containing one or more permanent data checks. The data check bytes are replaced as described in a utility control statement. The record is considered an error record, but because a data check is the only error that can occur, no EDW is appended to the error record.

Error Description Word (EDW)

The Error Description Word (EDW) consists of four bytes that are appended to the start of an error record.

The error description word is in EBCDIC format; for example, a 2 is represented as X'F2' and a C is represented as X'C3'. The information provided in each of the four bytes of the EDW is discussed below.

Byte	**Indicator meaning**
Level Status (Byte 0)	Identifies error records that result from interrecord dependency that cannot be identified in the *type status* byte.

Value	**Meaning**
0	Indicates any error record that will not cause questionable data in the following records. A *type* status other than zero accompanies this byte.
1	Indicates any error record that may cause questionable data in the following records, and for which the *level status* of the previous record was 0.
2	Indicates any error that contains questionable data because the error level of the preceding record was 1 or 2, or for any error record that may cause questionable data in the following records and for which the *level status* of the previous record was 1 or 2.

A level status of 1 or 2 is presented with error records resulting from the following:

- The start-of-record (SOR) location has a character defined as an error.
- The record contains two or more data check bytes side by side. These may have been an SOR and EOR (end-of-record).

- The record is longer than the user-specified maximum length record.
- The length of the record is not equal to the length of the first valid record of the same program level encountered on this cartridge. For this purpose, a valid record is one that contains no errors as identified in the type status, with the possible exception of being shorter than the user-specified minimum length.
- The record has a data-duplication dependency on a previous record with one of the above errors.

The level status is set to 0 when IEBTCRIN encounters: (1) a record without one of the previous errors, (2) a canceled record, or (3) the first record of a cartridge.

Byte	**Indicator Meaning**
Type Status (Byte 1)	Identifies records in error because of SOR, EOR, length, field, or data check error conditions.

Value	**Meaning**
0	Indicates any record that contains none of the following identifiable errors, but contains questionable data due to a level status other than zero. (See Level Status above.)
1	Indicates any record that has: (1) an SOR character of other than P1 through P8 or a GS code, (2) an EOR character of other than a VOK code for records when the user specified a record verification check, or (3) an EOR character of other record-verification check.
2	Indicates any record that has an incorrect length because it is: (1) longer than the user-specified maximum, (2) shorter than the user-specified minimum, or (3) not encountered on this cartridge.
4	Indicates any record that has a field error. A field error occurs when duplication or left-zero justification functions did not occur in a field because of an error condition. See "MTDI Editing Criteria" below.
8	Indicates any record that has a permanent data check error.

The type-status indicator can also have values of 3, 5, 6, 7, 9, A, B, C, D, E, and F. These values indicate a combination of SOR, EOR, length, field, and data check errors. For example, a value of A indicates a record with a data check error (8), as well as, an incorrect length (2).

Start-of-Record (Byte 2)	Indicates the start-of-record (SOR) character associated with this record. The SOR character can be 1 through 8, where 1 indicates P1, 2 indicates P2, etc., or E, which indicates the SOR character is in error.

End-of-Record (Byte 3)	Indicates the end-of-record (EOR) character associates with this record. The EOR character can be: U (unverified record); V (verified record); or E (EOR character is in error).

Sample Error Records

Figure 12-5 shows a stream of data bytes read sequentially from the tape cartridge reader.

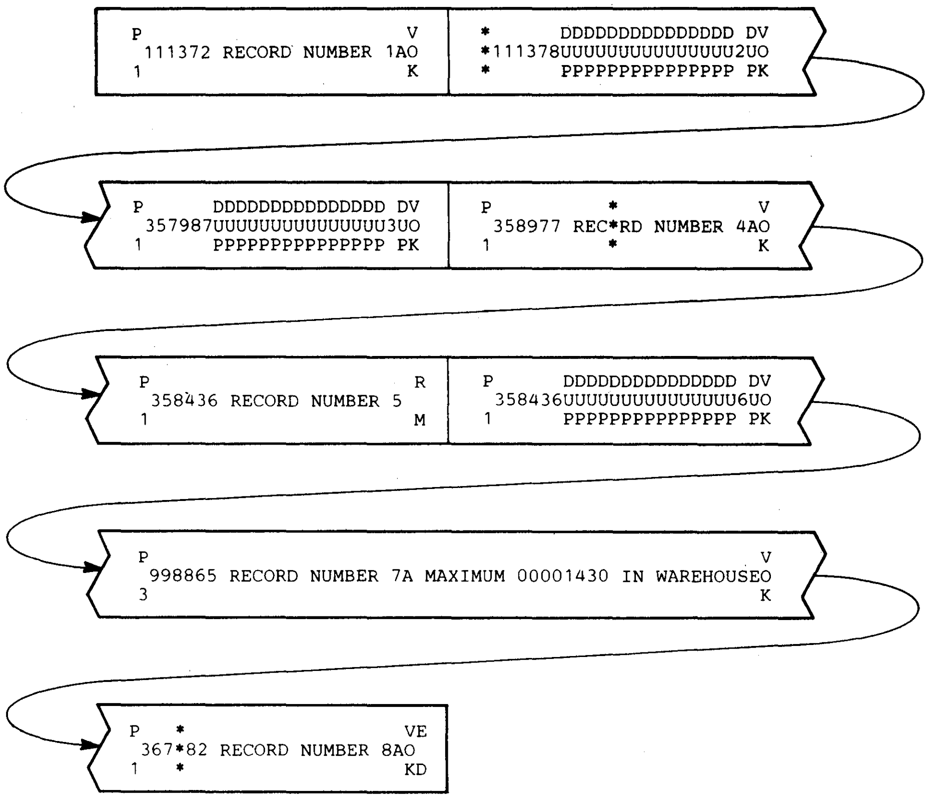


Figure 12-5. Tape Cartridge Reader Data Stream

Figure 12-6 shows the records constructed by IEBTCRIN from the input records shown in Figure 12-5. These records show some of the errors that can occur during processing and their effect on the Error Description Word. The following parameters were specified for these records:

```
                                                          72
TCRGEN      TYPE=MTDI,EDIT=EDITR,VERCHK=VOKCHK,           C
            MAXLN=50,REPLACE=X'5B'
```

IEBTCRIN classifies records 2 through 9 in Figure 12-6 as error records. The records are classified as follows:

- Record 1 is a valid record. It contains a program-level 1 code, and thus establishes the valid length for all program-level 1 records in this cartridge to be 25 bytes.
- Record 2 has a data check in the SOR location. Level status is set to 1 because the SOR location might have contained a cancel code that would cause any data duplicated on the following record to be questionable. The type status (9)

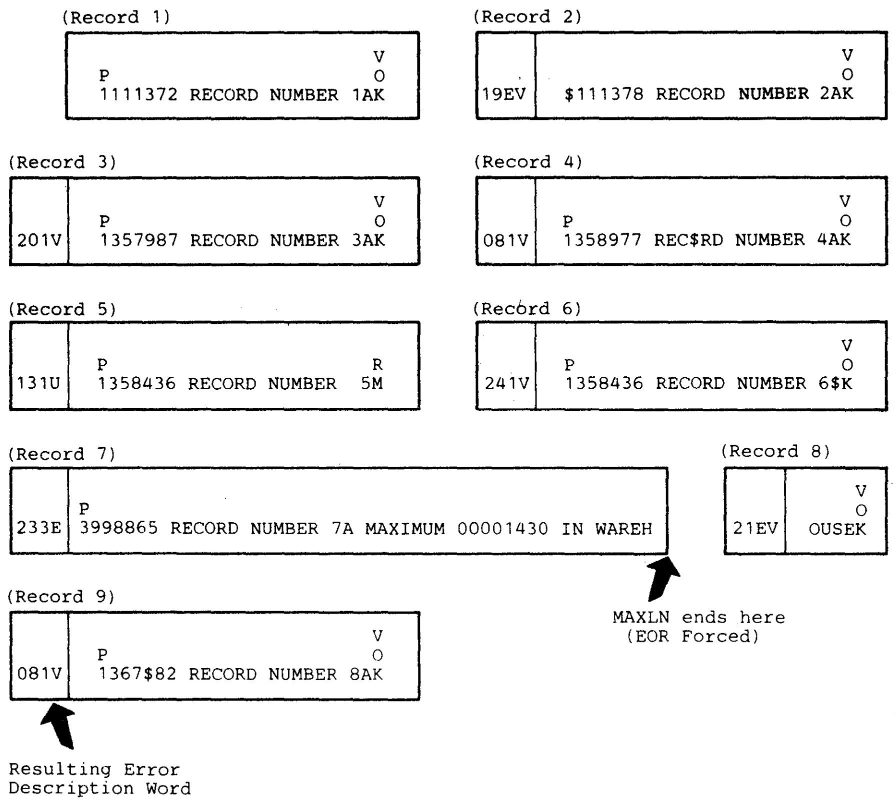


Figure 12-6. Record Construction

indicates the record has an incorrect SOR/EOR character (1) and a data check error (8).

- Record 3 contains no identifiable error, but contains questionable data because it requires duplication from the previous record, which had a level status of 1.
- Record 4 has a data check. Because it contained no DUP codes, the level status is set to 0.
- Record 5 is shorter than the first program-level 1 record on this cartridge (length error). This record also contains an RM code rather than a VOK code in the EOR location (VOKCHK was specified on the TCRGEN statement. Because IEBTCRIN cannot determine why the record is short, all data duplicated from this record is questionable; the level status is set to 1. The type status is set to 3 indicating an SOR/EOR error (1) and length error (2).
- Record 6 contains a DUP code that is beyond the last position of the preceding record.
- The seventh input record is longer than the maximum user-specified record length. Note that it is passed as two records. The first record (record 7) indicates an EOR error and a length error; the second (record 8) indicates an SOR error. Because record 7 is an error record, its length (50 bytes) is not established as the valid length for all program-level 3 records on this cartridge.
- Record 9 has a data check. Because it contained no DUP codes, the level status is set to 0.

Input and Output

IEBTCRIN uses the following input:

- An input data set, which contains data on tape cartridges to be read from the Tape Cartridge Reader (TCR). The input data set was created on either MTST or MTDI.
- A control data set, which contains utility control statements that are used to control the functions of IEBTCRIN.

IEBTCRIN produces the following output:

- An output data set, which contains the sequential output produced by the utility as a result of processing the cartridge input according to the utility control statements.
- An error output data set, which contains records that do not conform to the specifications for a valid record.
- A message data set, which contains diagnostic messages.

Return Codes

IEBTCRIN produces the following return codes:

- 00, which indicates normal termination.
- 04, which indicates warning message issued; execution permitted. Conditions leading to issuance of this code are: (1) SYSPRINT, SYSIN, SYSUT2, or SYSUT3 DD statements missing and (2) DCB parameters missing SYSUT2 or SYSUT3 DD statements.
- 12, which indicates diagnostic error message issued; execution terminated. Conditions leading to issuance of this code are: (1) SYSUT1 DD statement missing, (2) conflicting DCB parameters in DD statements, and (3) invalid or conflicting utility control statements.
- 16, which indicates terminal error message issued; execution terminated. Conditions leading to issuance of this code are: (1) permanent input/output errors (not including data checks on the TCR), (2) unsuccessful opening of data sets, (3) requests for termination by user exit routine, (4) insufficient storage available for execution, and (5) user exit routine not found.

Control

IEBTCRIN is controlled by job control statements and utility control statements. The job control statements are required to execute or invoke IEBTCRIN and to define the data sets that are used and produced by the program. The utility control statements are used to indicate the source of the input data cartridges (MTST or MTDI) and to specify the type of processing to be done.

Job Control Statements

Figure 12-7 shows the job control statements necessary for using IEBTCRIN.

If the SYSPRINT DD statement is missing, a message is written on the operator console and processing continues.

If some parameters are specified but others are omitted, IEBTCRIN attempts to set defaults for the missing parameters that are consistent with those supplied. For example, if RECFM=VBA is specified, IEBTCRIN assumes BLKSIZE=129 and

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEBTCRIN) or, if the job control statements reside in a procedure library, the procedure name.
SYSPRINT DD	Defines a sequential message data set, which can be written to any QSAM-supported output device.
SYSUT1 DD	Defines the input data set.
SYSUT2 DD	Defines a sequential output data set for valid records.
SYSUT3 DD	Defines a sequential output data set for error records.
SYSIN DD	Defines the control data set. The control data set normally resides in the input stream; however, it can be defined as a sequential data set or as a member of a partitioned data set. If this statement is not included, all utility control statement defaults are assumed and a message is issued to SYSPRINT. If DUMMY is specified, all utility control statement defaults are assumed.

Figure 12-7. IEBTCRIN Job Control Statements

LRECL=125. If LRECL, BLKSIZE, and RECFM are not specified, the defaults are LRECL=121, BLKSIZE=121, and RECFM=FBA.

For the SYSUT1 DD statement, only the UNIT keyword is required. The value specified in UNIT=xxxx can be '2495', the device address, or any other name that was generated in the system as a unit device name. The VOLUME=SER=keyword may be specified to identify the tape cartridges to be mounted. The volume serial number must be an externally recognizable name associated with the cartridges to be processed. A message is issued to the operator instructing that the cartridges identified by that name be mounted. If VOLUME is not specified, the name TCRINP is assumed and used in the mount message. The BUFL DCB parameter can be specified to indicate the size of input buffers; if BUFL is not specified, a value of 2000 is assumed.

Fixed and variable records on the SYSUT2 or SYSUT3 data set can be blocked through the specification of the BLKSIZE and RECFM DCB parameters.

SYSUT2 DD and SYSUT3 DD statements may be omitted or specified as DUMMY for other than sequential data sets. A message is issued on SYSPRINT and processing continues.

The DCB parameters defining the SYSIN, SYSPRINT, SYSUT2, and SYSUT3 data sets can be supplied from any valid source (for example, DD statements or a data set label). Because the output (SYSUT2 and/or SYSUT3) data sets are not opened until the first record is ready for output (after any OUTREC and/or ERROR exits), DCB parameters to be supplied from an existing data set label are not available for records constructed before the data set is opened. Therefore, the DCB parameters should always be provided in the DD statement even though they may already exist in the label. Otherwise, defaults are used to construct records until the data set is opened.

If a permanent error occurs on SYSIN, SYSUT1 (not including a data check), SYSUT2, or SYSUT3, a message is issued on SYSPRINT and the program is terminated. If a permanent input/output error occurs on SYSPRINT, both the failing message and a SYNADAF message indicating the error are written on the programmer's console and processing is terminated.

Utility Control Statements

Figure 12-8 shows the utility control statements necessary for using IEBTCRIN.

Statement	Use
TCRGEN	Specifies whether MTDI or MTST input is to be processed and the type of processing to be performed.
EXITS	Specifies any exit routines provided by the user.

Figure 12-8. IEBTCRIN Utility Control Statements

If these statements contain errors or inconsistencies, the program is terminated and the appropriate diagnostics are sent to the message data set. If TCRGEN is not specified, standard defaults are used.

TCRGEN Statement

The TCRGEN statement is used to indicate the device (MTDI or MTST) on which the input data was created and the type of processing to be performed on the input data.

The format of the TCRGEN statement is:

[*label*] **TCRGEN** **[TYPE={MTDI | MTST }]**
[,TRANS={STDUC | STDLC | *name* **| NOTRAN }]**
[,EDIT={EDITD | EDITR | NOEDIT }]
[,VERCHK= {NOCHK | VOKCHK }]
[,MINLN=*n* **]**
[,MAXLN=*n* **]**
[,REPLACE=X'*xx*'**]**
[,ERROPT= {NORMAL | NOERR }]

EXITS Statement

The EXITS statement is used to identify user-supplied exit routines, which must exist in either the user job library or the link library.

Upon entry, a parameter list is supplied to the exit routine. Upon returning from the exit routine, the user must provide an acceptable return code. See "Appendix A: Exit Routine Linkage."

The format of the EXITS statement is:

[*label*] **EXITS** **[ERROR=***routinename* **]**
[,OUTREC=*routinename* **]**
[,OUTHDR2=*routinename* **]**
[,OUTHDR3=*routinename* **]**
[,OUTTLR2=*routinename* **]**
[,OUTTLR3=*routinename* **]**

Operands	Applicable Control Statements	Description of Operands/Parameters
EDIT	TCRGEN	**EDIT={EDITD | EDITR | NOEDIT}** specifies the type of processing to be performed on MTDI input. These values can be coded: **EDITD** specifies that the input is to be edited and that SOR and EOR codes are to be deleted and not included as part of the output record. **EDITR** specifies that the input is to be edited and SOR and EOR codes are to be kept as part of the output record. **NOEDIT** specifies that no editing is to be performed. Data, including any group separator (GS) codes, is passed exactly as read from the cartridge.
ERROPT	TCRGEN	**ERROPT={NORMAL | NOERR}** specifies the disposition of all error records. ERROPT is ignored if a user error routine is specified in the EXITS statement. These values can be coded: **NORMAL** specifies that all error records are to be placed in the error data set (SYSUT3). **NOERR** specifies that all records (including error records) are placed in the normal output data set (SYSUT2). No records are placed in the error data set (SYSUT3).
ERROR	EXITS	**ERROR=***routinename* specifies the symbolic name of a routine that receives control before an error record is passed to the error output data set (SYSUT3). This exit routine can be used to analyze and, if possible, correct the error record. This parameter nullifies any ERROPT value.
MAXLN	TCRGEN	**MAXLN=***n* specifies the number of bytes, *n*, plus four for the record descriptor word when variable records are specified, to be contained in all but the last record passed to the output routine when editing is not performed. IEBTCRIN does not indicate the end of data from one cartridge and the beginning of data from the next. Usually this transition from one cartridge to another occurs within an output record. The last record passed to the output routine contains only the number of bytes remaining (plus four if the record format is variable) and is the only record that can be shorter than the length specified by MAXLN. The size of the records actually written depends on the record length (LRECL) specified for the output data set. **Default:** 120 bytes

Operands	Applicable Control Statements	Description of Operands/Parameters
MINLN	TCRGEN	**MINLN**=*n* specifies in bytes the length, *n*, of the shortest valid, edited record. This parameter is valid only when TYPE=MTDI and either EDIT=EDITD or EDIT=EDITR are specified. If IEBTCRIN encounters a record shorter than this specified length, the record is considered an error record. **Default:** No minimum length checking is performed.
OUTREC	EXITS	**OUTREC**=*routinename* specifies the symbolic name of a routine that receives control before the record is passed to the normal output data set (SYSUT2). In this exit routine, the user can process the record and perform his own output if output other than the SYSUT2 data set is desired. Any modification of an edited MTDI record may affect the editing of following records. The record returned from this exit is used to accomplish data duplication in the record that follows. If the SYSUT2 data set has specified variable length records, an RDW which is four bytes long is appended to the front of the record.
OUTHDR2	EXITS	**OUTHDR2**=*routinename* specifies the symbolic name of a routine that receives control during the opening of the SYSUT2 data set; this exit routine can be used to create user output header labels for the normal output data set (SYSUT2).
OUTHDR3	EXITS	**OUTHDR3**=*routinename* specifies the symbolic name of a routine that receives control during the opening of the SYSUT3 data set; this exit routine can be used to create user output header labels for the error data set (SYSUT2).
OUTTLR2	EXITS	**OUTTLR2**=*routinename* specifies the symbolic name of a routine that receives control during the closing of the SYSUT2 data set; this exit routine can be used to create user output trailer labels for the normal output data set (SYSUT2).
OUTTLR3	EXITS	**OUTTLR3**=*routinename* specifies the symbolic name of a routine that receives control during the closing of the SYSUT3 data set; this exit routine can be used to create user output trailer labels for the error data set (SYSUT3).

Operands	**Applicable Control Statements**	**Description of Operands/Parameters**
REPLACE	TCRGEN	**REPLACE=X'*xx*'** specifies the hexadecimal representation of the character to be used by IEBTCRIN to replace error bytes. REPLACE allows the user to identify and possibly correct error bytes on the error exit routine or in subsequent processing. The specified REPLACE character should be one that does not normally appear in the data. To replace error bytes on MTDI data, select a value for *xx* from Figure 12-2. To replace error bytes on MTST data, select a value for *xx* from Figure 12-3. The replacement of error bytes is accomplished before any specified MTST translation. Default: X'19', end-of-data
TRANS	TCRGEN	**TRANS={STDUC | STDLC | *name* | NOTRAN}** specifies the type of processing to be performed on MTST input. These values can be coded: **STDUC** specifies that the MTST code is to be translated to standard EBCDIC; alphabetic characters are translated to uppercase. **STDLC** specifies that the MTST code is to be translated to standard EBCDIC; alphabetic characters are not translated to uppercase. *name* specifies a user translate table to be used by IEBTCRIN. The translate table must exist as a load module named in a user job library or the link library. This load module must consist of a translate table which begins at the entry point and conforms to the specifications for the translate instruction (TR) found in *IBM System/370 Principles of Operation.* **NOTRAN** specifies that no translation and no special processing are to be performed. Data is passed exactly as read from the cartridge.
TYPE	TCRGEN	**TYPE={MTDI | MTST}** specifies the device on which the magnetic tape cartridge(s) was written. These values can be coded: **MTDI** specifies that the input was created on a Magnetic Data Inscriber. **MTST** specifies that the input was created on a Magnetic Tape SELECTRIC® typewriter.

Operands	Applicable Control Statements	Description of Operands/Parameters
VERCHK	TCRGEN	**VERCHK={NOCHK | VOCHK}** specifies whether a record-verification check is to be made on MTDI input that is to be edited. This parameter is valid only when TYPE=MTDI and either EDIT=EDITD or EDIT=EDITR are specified. These values can be coded: **NOCHK** specifies that no record-verification check is to be made. Either a record mark (RM) or a verify OK (VOK) code is considered a valid end-of-record code. **VOKCHK** specifies that a record-verification check is to be made. A record that does not contain a verify OK code is to be considered an error record.

Restrictions

- Because IEBTCRIN always constructs the SYSPRINT records with USASI (type A) control characters, type A control characters should be indicated when RECFM is specified.
- If a parameter that is not consistent with the other parameters is specified on SYSPRINT DD, a message is issued and processing is ended.
- The SYSUT1 DD statement is required for each use of IEBTCRIN.
- The SYSUT2 DD and SYSUT3 DD statements must identify sequential data sets; the data sets can have fixed, variable, variable spanned, or undefined records. These data sets can be written on any QSAM-supported device.
- If editing of MTDI input is specified on the utility control statements, the SYSUT3 LRECL parameter should be four bytes greater than the SYSUT2 LRECL parameter to include a four bytes long Error Description Word appended to the front of the record by IEBTCRIN. (See "Error Records" earlier in this chapter.) For variable records on either SYSUT2 or SYSUT3, the LRECL and BLKSIZE DCB parameters must be large enough to include the four bytes long record descriptor word.
- If inconsistent parameters are specified on SYSUT2 DD or SYSUT3 DD, a message is issued and processing is ended.

IEBTCRIN Examples

The following examples illustrate some of the uses of IEBTCRIN. Figure 12-9 can be used as a quick reference guide to IEBTCRIN examples. The numbers in the "Example" column point to examples that follow.

Operation	Data Set Organization	Device	Comments	Example
Edit MDTI input	Sequential	Disk and 9-track Tape	Fixed blocked output. Error exit routine specified	1
Invoke IEBTCRIN with LINK macro instruction	—	—	Assembler language interface instructions	2

Figure 12-9. IEBTCRIN Example Directory

IEBTCRIN Example 1

In this example, input from a tape cartridge is to be edited with normal records written to a disk volume and error records written to a tape volume.

```
//JOBNAME   JOB     O,SMITH,MSGLEVEL=1
//STPNAME   EXEC    PGM=IEBTCRIN
//SYSPRINT  DD      SYSOUT=A
//SYSUT1    DD      UNIT=TCR,VOLUME=SER=MYTAPE,DCB=(BUFL=3000)
//SYSUT2    DD      DSNAME=GOODSET,DISP=(NEW,CATLG),UNIT=disk
//VOLUME=SER=111222,SPACE=(TRK(10,10)),DCB=(LRECL=100,
//BLKSIZE=1000,RECFM=FB)
//SYSUT3    DD      DSNAME=ERRSET,UNIT=tape,VOLUME=SER=000001,
// DISP=(NEW,KEEP),DCB=(BLKSIZE=104,RECFM=U)
//SYSIN     DD      *
       TCRGEN TYPE=MTDI,EDIT=EDITD,MAXLN=100,REPLACE=X'5B'
       EXITS  ERROR=MYERR
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input tape cartridge data set. A console message instructs the operator to mount a set of cartridges named MYTAPE. The two input buffers are each 3000 bytes long (BUFL). The UNIT parameter assumes that TCR has been system generated as a unit name for the Tape Cartridge Reader.
- SYSUT2 DD defines a sequential data set for the normal output records. The data will be written to a disk volume.
- SYSUT3 DD defines a sequential data set for the error records. The records are undefined with a maximum block size of 104 bytes, including a 4-byte error description word.
- SYSIN DD defines the control data set, which follows in the input stream.
- TCRGEN indicates MTDI input. The input is to be edited with SOR and EOR codes deleted, the maximum valid record length is to be 100 bytes, and the replace character is a hexadecimal "5B". VERCHK is defaulted to NOCHK. Minimum record-length checking is not requested.
- EXITS indicates that a user has provided an exit routine to handle error records. Because no job library has been specified, the exit routine (MYERR) must reside in the link library.

IEBTCRIN Example 2

In this example, IEBTCRIN is invoked via the LINK macro instruction in an Assembler language program. An alternate name has been assigned to each of the DD statements used by IEBTCRIN. The job control for this step must include DD statements with the alternate DD names.

```
         LINK EP=IEBTCRIN,PARAM=(OPTLIST,DDNAME),VL=1
         CNOP 2,4  (OPTLIST must be on halfword boundary)
OPTLIST  DC   H'0'  (Length must be zero for IEBTCRIN)
         CNOP 2,4  (DDNAME list must be on halfword boundary)
DDNAME   DC   H'82'  (Length of DDNAME list)
         DC   8F'0'
         DC   C'NEWIN   '  (Alternate DDNAME for SYSIN)
         DC   C'NEWPRINT'  (Alternate DDNAME for SYSPRINT)
         DC   2F'0'
         DC   C'NEWUT1  '  (Alternate DDNAME for SYSUT1)
         DC   C'NEWUT2  '  (Alternate DDNAME for SYSUT2)
         DC   C'NEWUT3  '  (Alternate DDNAME for SYSUT3)
```

IEBUPDTE PROGRAM

IEBUPDTE is a data set utility used to incorporate IBM and user-generated source language modifications into sequential or partitioned data sets. Exits are provided for user routines that process user header and trailer labels.

IEBUPDTE can be used to:

- Create and update symbolic libraries.
- Incorporate changes to partitioned members or sequential data sets.
- Change the organization of a data set from sequential to partitioned or vice versa.

At the completion or termination of IEBUPDTE, the highest return code encountered within the program is passed to the calling program.

Creating and Updating Symbolic Libraries

IEBUPDTE can be used to create a library of partitioned members consisting of (at the most) 80-byte logical records. In addition, members can be added directly to an existing library, provided that the original space allocations are sufficient to incorporate the new members. In this manner, a cataloged procedure can be placed in a procedure library, or a set of job or utility control statements can be placed as a member in a partitioned library.

Incorporating Changes

IEBUPDTE can be used to modify an existing partitioned or sequential data set. Logical records can be replaced, deleted, renumbered, or added to the member or data set.

A sequential data set residing on a tape volume can be used to create a new master (that is, a modified copy) of the data set. A sequential data set residing on a direct access device can be modified either by creating a new master or by modifying the data set directly on the volume on which it resides.

A partitioned data set can be modified either by creating a new master or by modifying the data set directly on the volume on which it resides.

Changing Data Set Organization

IEBUPDTE can be used to change the organization of a data set from sequential to partitioned, or to change a member of a partitioned data set to a sequential data set (the original data set, however, remains unchanged). In addition, logical records can be replaced, deleted, renumbered, or added to the member or data set.

Input and Output

IEBUPDTE uses the following input:

- An input data set (also called the old master data set), which is to be modified or used as source data for a new master. The input data set is either a sequential data set or a member of a partitioned data set.
- A control data set, which contains utility control statements and, if applicable, input data. The data set is required for each use of IEBUPDTE.

IEBUPDTE produces the following output:

- An output data set, which is the result of the IEBUPDTE operation. The data set can be either sequential or partitioned. It can be either a new data set (that is, created during the present job step) or an existing data set, modified during the present job step.
- A message data set, which contains the utility program identification, control statements used in the job step, modification made to the input data set, and diagnostic messages, if applicable. The message data set is sequential.

IEBUPDTE provides a return code to indicate the results of program execution. The return codes and their meanings are:

- 00, which indicates successful completion.
- 04, which indicates that a control statement is coded incorrectly or used erroneously. If either the input or output is sequential, the job step is terminated. If both are partitioned, the program continues processing with the next function to be performed.
- 12, which indicates an unrecoverable error. The job step is terminated.
- 16, which indicates that a label processing code of 16 was received from a user's label processing routine. The job step is terminated.

Control

IEBUPDTE is controlled by job control statements and utility control statements. The job control statements are required to execute or invoke IEBUPDTE and to define the data sets that are used and produced by the program. The utility control statements are used to control the functions of IEBUPDTE and, in certain cases, to supply new or replacement data.

Job Control Statements

Figure 13-1 shows the job control statements necessary for using IEBUPDTE.

The input and output data sets contain blocked or unblocked logical records with record lengths of up to 80 bytes. The input and output data sets may have different block sizes as long as they are multiples of the logical record length.

If an ADD operation is specified with PARM=NEW in the EXEC statement, the SYSUT1 DD statement need not be coded.

If an UPDATE operation is specified, the SYSUT2 DD statement should not be coded.

If the SYSUT1 DD statement defines a sequential data set, the file sequence number of that data set must be included in the LABEL keyword (unless the data set is the first or only data set on the volume).

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEBUPDTE), or, if the job control statements reside in a procedure library, the procedure name. Additional information can be specified in the PARM parameter of the EXEC statement.
SYSPRINT DD	Defines a sequential message data set. The data set can be written to a system output device, a tape volume, or a direct access volume.
SYSUT1 DD	Defines the input (old master) data set. It can define a sequential data set on a card reader, a tape volume, or a direct access volume. Or, it can define a partitioned data set on a direct access volume.
SYSUT2 DD	Defines the output data set. It can define a sequential data set on a card punch, a printer, a tape volume, or a direct access device. It can define a partitioned data set on a direct access device.
SYSIN DD	Defines the control data set. The control data set normally resides in the input stream; however, it can be defined as a member of a partitioned data set.

Figure 13-1. IEBUPDTE Job Control Statements

If both the SYSUT1 and SYSUT2 DD statements specify standard user labels (SUL), IEBUPDTE copies user labels from SYSUT1 to SYSUT2.

If the SYSUT1 and SYSUT2 DD statements define the same partitioned data set, the old master data set can be updated without creating a new master data set; in this case, a copy of the updated member or members is written within the extent of the space originally allocated to the old master data set. Subsequent referrals to the updated member(s) will point to the newly written member(s). The member names themselves should not appear on the DD statements; they should be referenced only through IEBUPDTE control statements.

PARM Information on the EXEC Statement

Additional information can be coded in the PARM parameter of the EXEC statement, as follows:

PARM={NEW | <u>MOD</u>},[inhdr],[intlr]

Following are the PARM values:

- NEW, which specifies that the input consists solely of the control data set. The input data set is not defined if NEW is specified.
- MOD, which specifies that the input consists of both the control data set and the input data set. If neither NEW nor MOD is coded, MOD is assumed.
- "inhdr," which specifies the symbolic name of a routine that processes the user header label on the volume containing the control data set.
- "intlr," which specifies the symbolic name of a routine that processes the user trailer label on the volume containing the control data set.

Utility Control Statements

Figure 13-2 shows the utility control statements used to control IEBUPDTE.

Statement	Use
Function	Initiates an IEBUPDTE operation.
Detail	Used with the Function statement for special applications.
Data	A logical record of data to be used as a new or replacement record in the output data set.
LABEL	Indicates that the following data statements are to be treated as user labels.
ALIAS	Assigns aliases.
ENDUP	Terminates IEBUPDTE.

Figure 13-2. IEBUPDTE Utility Control Statements

Function Statement

The Function statement is used to initiate an IEBUPDTE operation. At least one Function statement must be provided for each member or data set to be processed.

A member or a data set can be added directly to an old master data set if the space originally allocated to the old master is sufficient to incorporate that new member or data set. ADD specifies that a member or a data set is to be added to an old master data set. If a member is to be added and the member name already exists in the old master data set, processing is terminated. If, however, PARM=NEW is specified on the EXEC statement, the member is replaced. For a sequential output master data set, PARM=NEW must always be specified on the EXEC statement. At least one blank must precede and follow ADD.

When a member replaces an identically named member on the old master data set or a member is changed and rewritten on the old master, the alias (if any) of the original member still refers to the original member. However, if an identical alias is specified for the newly written member, the original alias entry in the directory is changed to refer to the newly written member. REPL specifies that a member of a data set is being entered in its entirety as a replacement for a sequential data set or for a member of the old master data set. The member name must already exist in the old master data set. At least one blank must precede and follow REPL. CHANGE specifies that modifications are to be made to an existing member or data set. Use of the CHANGE Function statement without a NUMBER or DELETE Detail statement, or a Data statement causes an error condition. At least one blank space must precede and follow CHANGE. REPRO specifies that a member or a data set is to be copied in its entirety to a new master data set. At least one blank must precede and follow REPRO.

Members can be deleted from a copy of a library by being omitted from a series of REPRO Function statements within the same job step.

One sequential data set can be copied in a given job step. A sequential data set is deleted by being omitted from a series of job steps which copy only the desired data sets to a new volume. If the NEW subparameter is coded in the EXEC statement, only the ADD Function statement is permitted.

Figure 13-3 shows how the system status information (SSl=0A3CI23B) is packed.

Change level	Flag byte	Serial number	
byte 1	byte 2	byte 3	byte 4
0 A	3 C	1 2	3 B

Figure 13-3. Format of System Status Information

The format of the Function statement is:

./ [*label*] {**ADD** | **CHANGE** | **REPL** | **REPRO**}

[LIST=ALL]

[,SEQFLD={*ddl* | *ddl,ddl*}**]**

[,NEW= {PO | PS}]

[,MEMBER=*cccccccc* **]**

[,COLUMN=*dd***]**

[,UPDATE=INPLACE]

[,INHDR=*cccccccc* **]**

[,INTLR=*cccccccc* **]**

[,OUTHDR=*cccccccc* **]**

[,OUTTLR=*cccccccc* **]**

[,TOTAL=(*routinename*,

[,NAME=*cccccccc* **]**

[,LEVEL=*hh* **]**

[,SOURCE=*x* **]**

[,SSI=*hhhhhhhh* **]**

Function Restrictions

When UPDATE is specified:

- The SYSUT2 DD statement is not coded.
- The PARM parameter of the EXEC statement must imply or specify MOD.
- The NUMBER statement can be used to specify a renumbering operation.
- Data statements can be used to specify replacement information only.
- One CHANGE Function statement and one UPDATE parameter are permitted per job step.
- No functions other than replacement, renumbering, and header label modification (via the LABEL statement) can be specified.
- Only replaced records are listed unless the entire data set is renumbered.
- System status information cannot be changed.

Within an existing logical record, the data in the field defined by COLUMN is replaced by data from a subsequent data statement, as follows:

1. IEBUPDTE matches a sequence number of a Data statement with a sequence number of an existing logical record. In this manner, the COLUMN specification is applied to a specific logical record.
2. The information in the field within the Data statement replaces the information in the field within the existing logical record. For example, COLUMN=40 indicates that columns 40 through 80 (assuming 80-byte logical records) of a subsequent Data statement are to be used as replacement data for columns 40 through 80 of a logical record identified by a matching sequence number. (A sequence number in an existing logical record or Data statement need not be within the defined field.)

The COLUMN specification applies to the entire function, with the exception of:

- Logical records deleted by a subsequent DELETE Detail statement.
- Subsequent Data statements not having a matching sequence number for an existing logical record.
- Data statements containing information to be inserted in the place of a deleted logical record or records.

Figure 13-4 shows the use of NEW, MEMBER, and NAME parameters for different input and output data set organizations.

Input Data Set Organization	Output Data Set Organization	Parameter Combinations
Partitioned	Partitioned	With an ADD Function statement, use NAME to specify the name of the member to be placed in the partitioned data set defined by the SYSUT2 DD statement. If an additional name is required, an ALIAS statement can also be used. With a CHANGE, REPL, or REPRO Function statement, use NAME to specify the name of the member within the partitioned data set defined by the SYSUT1 DD statement. If a different or additional name is desired for the member in the partitioned data set defined by the SYSUT2 DD statement, use an ALIAS statement also.
None	Partitioned (New)	With each ADD Function statement, use NAME to assign a name for each member to be placed in the partitioned data set.
Partitioned	Sequential	With a Function statement, use NAME to specify the name of the member in the partitioned data set defined by the SYSUT1 DD statement. Use NEW=PS to specify the change in organization from partitioned to sequential. (The name and file sequence number assigned to the output master data set are specified in the SYSUT2 DD statement.)
Sequential	Partitioned	With a Function statement, use MEMBER to assign a name to the member to be placed in the partitioned data set defined by the SYSUT2 DD statement. Use NEW=PO to specify the change in organization from sequential to partitioned.

Figure 13-4. NEW, MEMBER, and NAME Parameters

Detail Statement

A Detail statement is used with a Function statement for certain applications, such as deleting or renumbering selected logical records. NUMBER specifies, when coded with a CHANGE Function statement, that the sequence number of one or more logical records is to be changed. It specifies, when coded with an ADD or REPL Function statement, the sequence numbers to be assigned to the records within new or replacement members or data sets. When used with an ADD or REPL Function statement, no more than one NUMBER Detail statement is permitted for each ADD or REPL Function statement. If NUMBER is coded, it must be preceded and followed by at least one blank. DELETE specifies, when coded with a CHANGE Function statement, that one or more logical records are to be deleted from a member or data set. If DELETE is coded, it must be preceded and followed by at least one blank.

Note: Logical records cannot be deleted in part; that is, a COLUMN specification in a Function statement is not applicable to records that are to be deleted. Each specific sequence number is handled only once in any single operation.

The format of a Detail statement is:

./[*label*]{**NUMBER** | **DELETE**}[**SEQ1**= {*cccccccc* | **ALL**}]
[,**SEQ2**=*cccccccc*]
[,**NEW1**=*cccccccc*]
[,**INCR**=*cccccccc*]
[,**INSERT=YES**]

Detail Restrictions

When INSERT is coded:

- The SEQ1 parameter specifies the existing logical record after which the insertion is to be made. The SEQ2 parameter need not be coded; SEQ1=ALL cannot be coded.
- The NEW1 parameter assigns a sequence number to the first logical record to be inserted. If the parameter is alphameric, the SEQFLD=(*ddl,ddl*) parameter should be coded.
- The INCR parameter is used to renumber as much as is necessary of the member or data set from the point of the first insertion; the member or data set is renumbered until an existing logical record is found whose sequence number is equal to or greater than the next sequence number to be assigned. If no such logical record is found, the entire member or data set is renumbered.
- Additional NUMBER Detail statements, if any, must specify INSERT. If a prior numbering operation renumbers the logical record specified in the SEQ1 parameter of a subsequent NUMBER Detail statement, any NEW1 or INCR parameter specifications in the latter NUMBER statement are overridden. The prior increment value is used to assign the next successive sequence numbers. If a prior numbering operation does not renumber the logical record specified in the SEQ1 parameter of a subsequent NUMBER Detail statement, the latter statement must contain NEW1 and INCR specifications.
- The block of Data statements to be inserted must contain blank sequence numbers.

- The insert operation is terminated when a Function statement, a Detail statement, and end-of-file indication, or a Data statement containing a sequence number is encountered.
- The SEQ1, SEQ2, and NEW1 parameters (with the exception of SEQ1=ALL) specify eight (maximum) alphameric characters. The INCR parameter specifies eight (maximum) numeric characters. Only the significant part of a numeric sequence number need be coded; for example, SEQ1=00000010 can be shortened to SEQ1-10. If, however, the numbers are alphameric, the alphabetic characters must be specified; for example, SEQ1=00ABC010 can be shortened to SEQ1=ABC010.

Data Statement

A Data Statement is used with a Function statement, or with a Function statement and a Detail statement. It contains a logical record used as replacement data for an existing logical record, or new data to be incorporated in the output master data set.

Each Data statement contains one logical record, which begins in the first column of the Data statement. The length of the logical record is equal to the logical record length (LRECL) specified for the output master data set. Each logical record contains a sequence number to determine where the data is to be placed in the output master data set.

When used with a CHANGE Function statement, a Data statement contains new or replacement data, as follows:

- If the sequence number in the Data statement is identical with a sequence number in an existing logical record, the Data statement replaces the existing logical record in the output master data set.
- If no corresponding sequence number is found within the existing records, the Data statement is inserted in the proper collating sequence within the output master data set. (For proper execution of this function, all records in the old master data set must have a sequence number.)
- If a Data statement with a sequence number is used and INSERT=YES was specified, the insert operation is terminated. IEBUPDTE will continue processing if this sequence number is at least equal to the next old master record (record following the referred to sequence record).

When used with an ADD or REPL Function statement, a Data statement contains new data to be placed in the output master data set.

Sequence numbers within the old master data set are assumed to be in ascending order. No validity checking of sequence numbers is performed for data statements or existing records.

Sequence numbers in Data statements must be in the same relative position as sequence numbers in existing logical records. (Sequence numbers are assumed to be in columns 73 through 80; if the numbers are in columns other than these, the length and relative position must be specified in a SEQFLD parameter within a preceding Function statement.)

LABEL Statement

The LABEL statement indicates that the following data statements are to be treated as user labels. These new user labels are placed on the output data set. The next Function statement indicates to IEBUPDTE that the last label Data statement of the group has been read.

The format of the LABEL statement is:

./[*name*] **LABEL**

There can be no more than two LABEL statements per execution of IEBUPDTE. There can be no more than eight label Data statements following any LABEL statement. The first four bytes of each 80-byte label Data statement must contain "UHLn" or "UTLn", where *n* is 1 through 8, for input header or input trailer labels respectively, to conform to IBM standards for user labels. Otherwise, data management will overlay the data with the proper four characters.

When IEBUPDTE encounters a LABEL statement, it reads up to eight Data statements and saves them for processing by user output label routines. If there are no such routines, the saved records are written by OPEN or CLOSE as user labels on the output data set. If there are user output label processing routines, IEBUPDTE passes a parameter list to the output label routines. This parameter list is described fully in "Appendix A: Exit Routine Linkage." The label buffer contains a label data record which the user routine can process before the record is written as a label. If the user routine specifies (via return codes to IEBUPDTE) more entries than there are label data records, the label buffer will contain meaningless information for the remaining entries to the user routine.

The position of the LABEL statement in the SYSIN data set, relative to Function statements, indicates the type of user label that follows the LABEL statement:

- To create output header labels, place the LABEL statement and its associated label Data statements before any Function statements in the input stream. A Function statement, other than LABEL, must follow the last label Data statement of the group.
- To create output trailer labels, place the LABEL statement and its associated label Data statements after any Function statements in the input stream, but before the ENDUP statement. The ENDUP statement is not optional in this case. It must follow the last label Data statement of the group if IEBUPDTE is to create output trailer labels.

When UPDATE is specified in a Function statement, user input header labels can be updated by user routines, but input trailer and output labels cannot be updated by user routines. User labels cannot be added or deleted. User input header labels are made available to user routines by the label buffer address in the parameter list. See "Appendix D: Processing User Labels" for a complete discussion of the linkage between utility programs and user-label processing routines. The return codes when UPDATE is used differ slightly from the standard codes discussed in "Appendix D: Processing User Labels," as follows:

- 0, which specifies that the system resumes normal processing; any additional user labels are ignored.
- 4, which specifies that the system does not write the label. The next user label is read into the label buffer area and control is returned to the user's routine. If there are no more user labels, the system resumes normal processing.
- 8, which specifies that the system writes the user labels from the label buffer area and resumes normal processing.
- 12, which specifies that the system writes the user label from the label buffer area, then reads the next input label into the label buffer area and returns control to the label processing routine. If there are no more user labels, the system resumes normal processing.

If the user wants to examine the replaced labels from the old master data set, he must:

1. Specify an update of the old master by coding the UPDATE parameter in a Function statement.
2. Include a LABEL statement in the input data set for either header or trailer labels.
3. Specify a corresponding user label routine.

If the above conditions are met, fourth and fifth parameter words will be added to the standard parameter list. The fourth parameter word is not now used; the fifth contains a pointer to the replaced label from the old master. In this case, the number of labels supplied in the SYSIN data set must not exceed the number of labels on the old master data set. If the user specifies, via return codes, more entries to the user's header label routine than there are labels in the input stream, the first parameter will point to the current header label on the old master data set for the remaining entries. In this case, the fifth parameter is meaningless.

The format of the LABEL statement is:

./[*label*] **LABEL**

ALIAS Statement

The ALIAS statement is used to create or retain an alias in an output (partitioned) master directory. The ALIAS statement can be used with any of the Function statements. Multiple aliases can be assigned to each member up to a maximum of 16 aliases.

Note: If an ALIAS statement is specifying a name which already exists on the data set, the original TTR of that directory entry will be destroyed.

ALIAS must be preceded and followed by at least one blank. If multiple ALIAS statements are used, they must follow the data records.

The format of the ALIAS statement is:

./[*label*] **ALIAS NAME=***cccccccc*

ENDUP Statement

An ENDUP statement can be used to indicate the end of SYSIN input to this job step. It serves as an end-of-data indication if there is no other preceding delimiter statement. The ENDUP statement follows the last group of SYSIN control statements.

ENDUP must be preceded and followed by at least one blank. The ENDUP statement must follow the last label Data statement if IEBUPDTE is used to create output trailer labels.

The format of the ENDUP statement is:

./[*label*] **ENDUP**

Operands	Applicable Control Statements	Description of Operands/Parameters
./	ADD REPL CHANGE REPRO NUMBER DELETE LABEL ALIAS ENDUP	**./** is required and must appear in columns 1 and 2.
COLUMN	CHANGE	**COLUMN=**{*nn* | **1**} specifies, in decimal, the starting column of a data field within a logical record image. The field extends to the end of the image. Within an existing logical record, the data in the defined field is replaced by data from a subsequent Data statement.
INCR	NUMBER	**INCR=***cccccccc* specifies an increment value used for assigning successive sequence numbers to new or replacement logical records, or specifies an increment value used for renumbering existing logical records.
INHDR	ADD REPL CHANGE REPRO	**INHDR=***cccccccc* specifies the symbolic name of the user routine that handles any user input (SYSUT1) header labels. When used with UPDATE, this routine assumes a special function. This parameter is valid only when a sequential data set is being processed.
INSERT	CHANGE NUMBER	**INSERT=YES** specifies the insertion of a block of logical records. The records, which are Data statements containing blank sequence numbers, are numbered and inserted in the output master data set. INSERT is valid only when coded with both a CHANGE Function statement and a NUMBER Detail statement. SEQ1, NEW1, and INCR are required on the first NUMBER Detail statement.
INTLR	ADD REPL CHANGE REPRO	**INTLR=***cccccccc* specifies the symbolic name of the user routine that handles any user input (SYSUT1) trailer labels. INTLR is valid only when a sequential data set is being processed, but not when UPDATE is coded.
LEVEL	ADD REPL CHANGE REPRO	**LEVEL=***hh* specifies the change (update) level in hexadecimal (00-FF). The level number is recorded in the directory entry of the output member. This parameter is valid only when a member of a partitioned data set is being processed. This parameter has no effect when SSI is specifed.

Operands	Applicable Control Statements	Description of Operands/Parameters
LIST	ADD REPL CHANGE REPRO	**LIST=ALL** specifies that the SYSPRINT data set is to contain the entire updated member or data set and the control statements used in its creation. **Default:** For old data sets, if LIST is omitted, the SYSPRINT data set contains modifications and control statements only. If UPDATE was specified, the entire updated member is listed only when renumbering has been done. For new data sets, the entire member or data set and the control statements used in its creation are always written to the SYSPRINT data set.
MEMBER	ADD REPL CHANGE REPRO	**MEMBER=***cccccccc* specifies a name to be assigned to the member placed in the partitioned data set defined by the SYSUT2 DD statement. MEMBER is used only when SYSUT1 defines a sequential data set, SYSUT2 defines a partitioned data set, and NEW=PO is specified. Refer to Figure 13-4 for the use of MEMBER with NEW.
NAME	ADD REPL CHANGE REPRO ALIAS	For ALIAS: **NAME=***cccccccc* specifies a one- to eight character alias. For all others: **NAME=***cccccccc* indicates the name of the member placed into the partitioned data set. The member name need not be specified in the DD statement itself. NAME must be provided to identify each input member. Refer to Figure 13-4 for the use of NAME with NEW. This parameter is valid only when a member of a partioned data set is being processed.
name	ADD REPL CHANGE REPRO NUMBER DELETE LABEL ALIAS ENDUP	*name* specifies an optional name which begins in column 3 and extends no further than column 10.

Operands	Applicable Control Statements	Description of Operands/Parameters
NEW	ADD REPL CHANGE REPRO	**NEW={PO | PS}** specifies the organization of the old master data set and the organization of the updated output. NEW should not be specified unless the organization of the new master data set is different from the organization of the old master. Refer to Figure 13-4 for the use of NEW with NAME and MEMBER. These values can be coded: **PO** specifies that the old master data set is a sequential data set, and that the updated output is to become a member of a partitioned data set. **PS** specifies that the old master data set is a partitioned data set, and that a member of that data set is to be converted into a sequential data set.
NEW1	NUMBER	**NEW1**=*cccccccc* specifies the first sequence number assigned to new or replacement data, or specifies the first sequence number assigned in a renumbering operation. A value specified in NEW1 must be greater than a value specified in SEQ1 (unless SEQ1=ALL is specified, in which case this rule does not apply).
OUTHDR	ADD REPL CHANGE REPRO	**OUTHDR**=*cccccccc* specifies the symbolic name of the user routine that handles any user output (SYSUT2) header labels. OUTHDR is valid only when a sequential data set is being processed, but not when UPDATE is coded.
OUTTLR	ADD REPL CHANGE REPRO	**OUTTLR**=*cccccccc* specifies the symbolic name of the user routine that handles any user output (SYSUT2) trailer labels. OUTTLR is valid only when a sequential data set is being processed, but not when UPDATE is coded.

Operands	Applicable Control Statements	Description of Operands/Parameters
SEQ1	NUMBER DELETE	**SEQ1**={*cccccccc* | **ALL**} specifies records to be renumbered, deleted, or assigned sequence numbers. These values can be coded: *cccccccc* specifies the sequence number of the first logical record to be renumbered or deleted. This value is not coded in a NUMBER Detail statement that is used with an ADD or REPL Function statement. When this value is used in an insert operation, it specifies the existing logical record after which an insert is to be made. It must not equal the number of a statement just replaced or added. Refer to the INSERT parameter for additional discussion. **ALL** specifies a renumbering operation for the entire member or data set. ALL is used only when a CHANGE Function statement and a NUMBER Detail statement are used. ALL must be coded if sequence numbers are to be assigned to existing logical records having blank sequence numbers. If ALL is not coded, all existing logical records having blank sequence numbers. copied directly to the output master data set. When ALL is coded: (1) SEQ2 need not be coded and (2) one NUMBER Detail statement is permitted per Function statement. Refer to the INSERT parameter for additional discussion.
SEQ2	NUMBER DELETE	**SEQ2**=*cccccccc* specifies the sequence number of the last logical record to be renumbered or deleted. SEQ2 is required on all DELETE Detail statements. If only one record is to be deleted, the SEQ1 and SEQ2 specifications must be identical. SEQ2 is not coded in a NUMBER Detail statement that is used with an ADD or REPL Function statement.

Operands	Applicable Control Statements	Description of Operands/Parameters
SEQFLD	ADD REPL CHANGE REPRO	**SEQFLD=**{ *ddl* | (*ddl,ddl*)} *ddl* specifies, in decimal, the starting column (up to column 80) and length (8 or less) of sequence numbers within existing logical records and subsequent Data statements. Note that the starting column specification (*dd*) plus the length (*l*) cannot exceed the logical record length (LRECL) plus 1. Sequence numbers on incoming Data statements and existing logical records must be padded to the left with enough zeros to fill the length of the sequence field. (*ddl, ddl*) may be used when an alphameric sequence number generation is required. The first *ddl* specifies the sequence number columns as above. The second *ddl* specifies, in decimal, the starting column (up to column 80) and length (8 or less) of the numeric portion of the sequence numbers in subsequent NUMBER statements. This information is used to determine which portion of the sequence number specified by the NEW1 parameter may be incremented and which portion(s) should be copied to generate a new sequence number for inserted or renumbered records. **Note:** The numeric columns must fall within the sequence number columns specified (or defaulted) by the first *ddl*. Acceptable alphameric characters are A-Z, 1-9, @, #, $, *. **Default:** 738 is assumed, that is, an eight-byte sequence number beginning in column 73. Therefore, if existing logical records and subsequent Data statements have sequence numbers in columns 73 through 80, this keyword need not be coded.
SOURCE	ADD REPL CHANGE REPRO	**SOURCE=***x* specifies user modifications when the *x* value is 0, or IBM modifications when the *x* value is 1. The source is recorded in the directory entry of the output member. This parameter is valid only when a member of a partitioned data set is being processed. This parameter has no effect when SSI is specified.
SSI	ADD REPL CHANGE REPRO	**SSI=***hhhhhhhh* specifies eight hexadecimal characters of system status information (SSI) to be placed in the directory of the new master data set as four packed hexadecimal bytes of user data. This parameter is valid only when a member of a partitioned data set is being processed. SSI overrides any LEVEL or SOURCE data given on the same Function statement.

Operands	Applicable Control Statements	Description of Operands/Parameters
TOTAL	ADD REPL CHANGE REPRO	**TOTAL=**(*routinename,size*) specifies that exits to a user's routine are to be provided prior to writing each record. This parameter is valid only when a sequential data set is being processed. These values are coded: *routinename* specifies the name of the user's totaling routine. *size* specifies the number of bytes required for the user's data. The size should not exceed 32K, nor be less than 2 bytes. In addition, the keyword OPTCD=T must be specified for the SYSUT2 (output) DD statement. Refer to "Appendix A: Exit Routine Linkage" for a discussion of linkage conventions for user routines.
UPDATE	CHANGE	**UPDATE=INPLACE** specifies that the old master data set is to be updated within the space it actually occupies. The old master data set must reside on a direct access device. UPDATE is valid only when coded with CHANGE. No other function statements (ADD, REPL, REPRO) may be in the same job step.

Restrictions

- The output data set can have a blocking factor that is different from the input data set; however, if insufficient space is allocated for reblocked records, the update request is terminated.
- The message data set has a logical record length of 121 bytes, and consists of fixed length, blocked or unblocked records with an ASA control character in the first byte of each record. The input and output data sets have a logical record length of 80 bytes or less, and consist of standard fixed blocked (RECFM=FB) or unblocked records. The control data set contains 80-byte, blocked or unblocked records.
- The SYSIN DD statement is required for each use of IEBUPDTE.
- When UPDATE=INPLACE is specified, there must be no other function statements in the job step.
- Space must be allocated for an output data set (SYSUT2 DD statement) that is to reside on a direct access device, unless the data set is an existing data set.
- The SYSUT2 DD statement must not specify a DUMMY data set.
- When adding a member to an existing partitioned data set using an ADD Function statement, any DCB parameters specified on the SYSUT1 and SYSUT2 DD statements (or the SYSUT2 DD statement if that is the only one specified) must be the same as the DCB parameters already existing for the data set.
- If the SYSUT1 and SYSUT2 DD statements define the same sequential data set (direct access only), only those operations that add data to the end of the existing data set can be made. In these cases:
    1. The PARM parameter of the EXEC statement must imply or specify MOD. (See "PARM Information on the EXEC Statement" below.)
    2. The DISP parameter of the SYSUT1 DD statement must specify OLD.
    3. The DISP parameter of the SYSUT2 DD statement must specify MOD.
- The SYSIN DD statement is required for each use of IEBUPDTE.
- When UPDATE=INPLACE is specified, there must be no other function statements in the job step.

IEBUPDTE Examples

The following examples illustrate some of the uses of IEBUPDTE. Figure 13-5 can be used as a quick reference guide to IEBUPDTE examples. The numbers in the "Example" column point to examples that follow.

Operation	Data Set Organization	Device	Comments	Example
ADD and REPL	Partitioned	Disk	SYSUT1 and SYSUT2 DD statements define the same data set. A JCL procedure residing in the control data set is to be stored as a new member of a procedure library (PROCLIB). Another JCL procedure, also in the IEBUPDTE control data set, is to replace an existing member in PROCLIB.	1
CREATE a partitioned library	Partitioned	Disk	Input data is in the control data set. Output partitioned data set is to contain three members.	2
CREATE a partitioned data set	Partitioned	Disk	Input from control data set and from existing partitioned data set. Output partitioned data set is to contain three members.	3
UPDATE INPLACE and renumber	Partitioned	Disk	Input data set is considered to be the output data set as well; therefore, no SYSUT2 DD statement is required.	4
CREATE and DELETE	Partitioned, Sequential	Disk and Tape	Sequential master is to be created from partitioned disk input. Selected records are to be deleted. Blocked output.	5
CREATE, DELETE, and UPDATE	Sequential, Partitioned	Tape and Disk	Partitioned data set is to be created from sequential input. Records are to be deleted and updated. Sequence numbers in columns other than 73 through 80. One member is to be placed in the output data set.	6
INSERT	Partitioned	Disk	Block of logical records is to be inserted into an existing member. SYSUT1 and SYSUT2 DD statements define the same data set.	7
INSERT	Partitioned	Disk	Two blocks of logical records are to be inserted into an existing member. SYSUT1 and SYSUT2 DD statements define the same data set. Sequence numbers are alphanumeric.	8
CREATE	Sequential	Card Reader, and Disk	Sequential data set with user labels is to be created from card input.	9
COPY	Sequential	Disk	Sequential data set is to be copied from one direct access volume to another; user labels can be processed by exit routines.	10
CREATE	Partitioned	Disk	Create a new generation.	11

Figure 13-5. IEBUPDTE Example Directory

Note: Examples which use *disk* or *tape*, in place of actual device-ids, must be changed before use. See the Device Support section, in the Introduction to this manual, for valid device-id notation.

IEBUPDTE Example 1

In this example, two procedures are to be placed in the cataloged procedure library, SYS1.PROCLIB. The example assumes that the two procedures can be accommodated within the space originally allocated to the procedure library.

```
//UPDATE    JOB  09#660,SMITH
//          EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1    DD   DSNAME=SYS1.PROCLIB,DISP=OLD
//SYSUT2    DD   DSNAME=SYS1.PROCLIB,DISP=OLD
//SYSIN     DD   DATA
./         ADD   LIST=ALL,NAME=ERASE,LEVEL=01,SOURCE=0
./      NUMBER   NEW1=10,INCR=10
//ERASE     EXEC PGM=IEBUPDTE
//DD1       DD   UNIT=disk,DISP=(OLD,KEEP),VOLUME=SER=111111
//SYSPRINT DD   SYSOUT=A
./       REPL LIST=ALL,NAME=LISTPROC
./       NUMBER NEW1=10,INCR=10
//LIST      EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSUT1    DD   DISP=SHR,
//               DSN=SYS1.PROCLIB(&MEMBER)
//SYSUT2    DD   SYSOUT=A,
//               DCB=(RECFM=F,BLKSIZE=80)
//SYSIN     DD   DUMMY
./       ENDUP
/*
```

The control statements are discussed below:

- SYSUT1 and SYSUT2 DD define the SYS1.PROCLIB data set, which is assumed to be cataloged.
- SYSIN DD defines the control data set. The data set contains the utility control statements and the data to be placed in the procedure library.
- The ADD Function statement indicates that records (Data statements) in the control data set are to be placed in the output. The newly created procedure is to be listed in the message data set.

  The ADD function will not take place if a member, named ERASE, already exists in the new master data set referenced by SYSUT2.
- The REPL function statement indicates that records (Data statements) in the control data set are to replace an already existing member. The member is stored in the new master data set referenced by SYSUT2. The REPL function will only take place if a member named LISTPROC already exists in the old master data set referenced by SYSUT1.
- The NUMBER Detail statement indicates that the new and replacement procedures are to be assigned sequence numbers. The first record of each procedure is to be assigned sequence number 10; the next record is to be assigned sequence number 20, and so on.

IEBUPDTE Example 2

In this example, a three member, partitioned library is to be created. The input data is contained solely in the control data set.

```
//UPDATE    JOB  09#770,SMITH
//          EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD   SYSOUT=A
//SYSUT2    DD   DSNAME=OUTLIB,UNIT=disk,DISP=(NEW,KEEP),
// VOLUME=SER=111112,SPACE=(TRK,(50,,10)),DCB=(RECFM=F,
// LRECL=80,BLKSIZE=80)
//SYSIN     DD   DATA
./          ADD  NAME=MEMB1,LEVEL=00,SOURCE=0,LIST=ALL
```

(Data statements, sequence numbers in columns 73 through 80)

```
./          ADD  NAME=MEMB2,LEVEL=00,SOURCE=0,LIST=ALL
```

(Data statements, sequence numbers in columns 73 through 80)

```
./          ADD  NAME=MEMB3,LEVEL=00,SOURCE=0,LIST=ALL
```

(Data statements, sequence numbers in columns 73 through 80)

```
./        ENDUP
/*
```

The control statements are discussed below:

- SYSUT2 DD defines the new partitioned master OUTLIB. Enough space is allocated to allow for subsequent modifications without creating a new master data set.
- SYSIN DD defines the control data set. The data set contains the utility control statements and the data to be placed as three members in the output partitioned data set.
- The ADD Function statements indicate that subsequent Data statements are to be placed as members in the output partitioned data set. Each ADD Function statement specifies a member name for subsequent data and indicates that the member is to be listed in the message data set.
- The Data statements contain the data to be placed in the output partitioned data set.
- ENDUP signals the end of control data set input.

Note: Because sequence numbers (other than blank numbers) are included within the Data statements, no NUMBER Detail statements are included in the example.

IEBUPDTE Example 3

In this example, a three-member, partitioned data set (NEWMCLIB) is to be created. The data set is to contain:

- Two members (ATTACH and DETACH) copied from an existing partitioned data set (SYS1.MACLIB).
- A new member (EXIT), which is contained in the control data set.

```
//UPDATE    JOB  09#770,SMITH
//          EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1    DD   DSNAME=SYS1.MACLIB,DISP=SHR,UNIT=disk
//SYSUT2    DD   DSNAME=NEWMCLIB,VOLUME=SER=111112,UNIT=disk,
// DISP=(NEW,KEEP),SPACE=(TRK,(100,,10)),DCB=(RECFM=F,
// LRECL=80,BLKSIZE=80)
//SYSIN     DD   DATA
./       REPRO   NAME=ATTACH,LEVEL=00,SOURCE=1,LIST=ALL
./       REPRO   NAME=DETACH,LEVEL=00,SOURCE=1,LIST=ALL
./         ADD   NAME=EXIT,LEVEL=00,SOURCE=1,LIST=ALL
./      NUMBER   NEW1=10,INCR=100
```

(Data cards for EXIT member)

```
./       ENDUP
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input partitioned data set SYS1.MACLIB, which is assumed to be cataloged.
- SYSUT2 DD defines the output partitioned data set NEWMCLIB. Enough space is allocated to allow for subsequent modifications without creating a new master data set.
- SYSIN DD defines the control data set.
- The REPRO Function statements identify the existing input members to be copied onto the output data set. These members are also listed in the message data set.
- The ADD Function statement indicates that records (subsequent Data statements) are to be placed as a member in the output partitioned data set. The Data statements are to be listed in the message data set.
- The NUMBER Detail statement assigns sequence numbers to the Data statements. (The Data statements contain blank sequence numbers in columns 73 through 80.) The first record of the output member is assigned sequence number 10; subsequent records are incremented by 100.
- ENDUP signals the end of SYSIN data.

Note: The three named input members (ATTACH, DETACH, and BLDL) do not have to be specified in the order of their collating sequence in the old master.

IEBUPDTE Example 4

In this example, a member (MODMEMB) is to be updated within the space it actually occupies. Two existing logical records are to be replaced, and the entire member is to be renumbered.

```
//UPDATE    JOB  09#770,SMITH
//          EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1    DD   DSNAME=PDS,UNIT=disk,DISP=(OLD,KEEP),
// VOLUME=SER=111112
//SYSIN     DD   *
./      CHANGE   NAME=MODMEMB,LIST=ALL,UPDATE=INPLACE
./      NUMBER   SEQ1=ALL,NEW1=10,INCR=5
```

(Data statement 1, sequence number 00000020)
(Data statement 2, sequence number 00000035)

```
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the data set that is to be updated in place. (Note that the member name need not be specified in the DD statement.)
- SYSIN DD defines the control data set.
- The CHANGE Function statement indicates the name of the member to be updated and specifies the UPDATE=INPLACE operation. The entire member is to be listed in the message data set.
- The NUMBER Detail statement indicates that the entire member is to be renumbered, and specifies the first sequence number to be assigned and the increment value for successive sequence numbers.
- The Data statements replace existing logical records having sequence numbers of 20 and 35.

IEBUPDTE Example 5

In this example, a sequential master data set is to be created from partitioned input and selected logical records are to be deleted.

```
//UPDATE   JOB   09#770,SMITH
//         EXEC  PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD    SYSOUT=A
//SYSUT1   DD    DSNAME=PARTDS,UNIT=disk,DISP=(OLD,KEEP),
// VOLUME=SER=111112
//SYSUT2   DD    DSNAME=SEQDS,UNIT=tape,LABEL=(2,SL),
// DISP=(,KEEP),VOLUME=SER=001234,DCB=(RECFM=FB,
// LRECL=80,BLKSIZE=2000)
//SYSIN    DD    *
./      CHANGE   NEW=PS,NAME=OLDMEMB1
```

(Data statement 1, sequence number 00000123)

```
./      DELETE   SEQ1=223,SEQ2=246
```

(Data statement 2, sequence number 00000224)

```
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input partitioned data set PARTDS.
- SYSUT2 DD defines the output sequential data set. The data set is to be written as the second data set on a tape volume.
- SYSIN DD defines the control data set.
- CHANGE identifies the input member (OLDMEMB1) and indicates that the output is to be a sequential data set (NEW=PS).
- The first Data statement replaces the logical record whose sequence number is identical to the sequence number in the Data statement (00000123). If no such logical record exists, the Data statement is incorporated in the proper sequence within the output data set.
- The DELETE Detail statement deletes logical records having sequence numbers from 223 through 246.
- The second Data statement is inserted in the proper sequence in the output data set.

Note: Only one member can be used as input when converting to sequential organization.

IEBUPDTE Example 6

In this example, a member of a partitioned data set is to be created from sequential input and existing logical records are to be updated.

```
                                                                     72
//UPDATE    JOB  09#770,SMITH
//          EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1    DD   DSNAME=OLDSEQDS,UNIT=tape
// DISP=(OLD,KEEP),VOLUME=SER=001234
//SYSUT2    DD   DSNAME=NEWPART,UNIT=disk,DISP=(,KEEP),
// VOLUME=SER=111112,SPACE=(TRK,(10,5,5)),
// DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN     DD   *
./      CHANGE   NEW=PO,MEMBER=PARMEM1,LEVEL=01,                     C
./               SEQFLD=605,COLUMN=40,SOURCE=0
```

(Data statement 1, sequence number 00020)

```
./      DELETE   SEQ1=220,SEQ2=250
```

(Data statement 2, sequence number 00230)
(Data statement 3, sequence number 00260)

```
./       ALIAS   NAME=MEMB1
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the input sequential data set (OLDSEQDS). The data set resides on a tape volume.
- SYSUT2 DD defines the output partitioned data set. Enough space is allocated to provide for members that might be added in the future.
- SYSIN DD defines the control data set.
- The CHANGE Function statement identifies the output member and indicates that a conversion from sequential input to partitioned output is to be made. The SEQFLD parameter indicates that a five-byte sequence number is located in columns 60 through 64 of each Data statement. The COLUMN parameter specifies the starting column of a field (within subsequent Data statements) from which replacement information is obtained.
- The first Data statement is used as replacement data. Columns 40 through 80 of the statement replace columns 40 through 80 of the corresponding logical record. If no such logical record exists, the entire card image is inserted in the output member.
- The DELETE Detail statement deletes all of the logical records having sequence numbers from 220 through 250.
- The second Data statement, whose sequence number falls within the range specified in the DELETE Detail statement, is incorporated in its entirety in the output member.
- The third Data statement, which is beyond the range of the DELETE Detail statement, is treated in the same manner as the first Data statement.
- ALIAS assigns the alias MEMB1 to the output member PARMEM1.

IEBUPDTE Example 7

In this example, a block of three logical records is to be inserted into an existing member, and the updated member is to be placed in the existing partitioned data set.

Figure 13-6 shows existing sequence numbers, new sequence numbers, and Data statements to be inserted.

Sequence Numbers and Data Statements to be Inserted	New Sequence Numbers
10	10
15	15
Data statement 1	20
Data statement 2	25
Data statement 3	30
20	35
25	40
30	45

Figure 13-6. Sequence Numbers and Data Statements to be Inserted

```
//UPDATE    JOB  09#770,SMITH
//          EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD    SYSOUT=A
//SYSUT1    DD   DSNAME=PDS,UNIT=disk,DISP=(OLD,KEEP),
// VOLUME=SER=111112
//SYSUT2    DD   DSNAME=PDS,UNIT=disk,DISP=(OLD,KEEP),
// VOLUME=SER=111112
//SYSIN     DD   *
./      CHANGE   NAME=RENUM,LIST=ALL,LEVEL=01,SOURCE=0
./      NUMBER   SEQ1=15,NEW1=20,INCR=5,INSERT=YES
(Data statement 1)
(Data statement 2)
(Data statement 3)
/*
```

The control statements are discussed below:

- SYSUT1 and SYSUT2 DD define the partitioned data set (PDS).
- SYSIN DD defines the control data set.
- The CHANGE Function statement identifies the input member RENUM. The entire member is to be listed in the message data set.
- The NUMBER Detail statement specifies the insert operation and controls the renumbering operation.
- The Data statements are the logical records to be inserted. (Sequence numbers are assigned when the Data statements are inserted.)

In this example, the existing logical records have sequence numbers 10, 15, 20, 25, 30, etc. Sequence numbers are assigned by the NUMBER Detail statement, as follows:

1. Data statement 1 is assigned sequence number 20 (NEW1=20) and inserted after existing logical record 15 (SEQ1=15).
2. Data statements 2 and 3 are assigned sequence numbers 25 and 30 (INCR=5) and are inserted after Data statement 1.

3. Existing logical records 20, 25, and 30 are assigned sequence numbers 35, 40, and 45, respectively.
4. The remaining logical records in the member are renumbered.

IEBUPDTE Example 8

In this example, two blocks (three logical records per block) are to be inserted into an existing member, and the member is to be placed in the existing partitioned data set. A portion of the output member is to be renumbered.

```
//UPDATE    JOB  09#770,SMITH
//          EXEC PGM=IEBUPDTE,PARM=MOD                          72
//SYSPRINT DD   SYSOUT=A
//SYSUT1    DD   DSNAME=PDS,UNIT=disk,DISP=(OLD,KEEP),
// VOLUME=SER=111112
//SYSUT2    DD   DSNAME=PDS,UNIT=disk,DISP=(OLD,KEEP),
// VOLUME=SER=111112
//SYSIN     DD   *
./      CHANGE   NAME=RENUM,LIST=ALL,LEVEL=01,SOURCE=0,         C
./               SEQFLD=(765,783)
./      NUMBER   SEQ1=AA015,NEW1=AA020,INCR=5,INSERT=YES
(Data statement 1)
(Data statement 2)
(Data statement 3)
./      NUMBER   SEQ1=AA030,INSERT=YES
(Data statement 4)
(Data statement 5)
(Data statement 6)
(Data statement 7, sequence number AA035)
/*
```

Figure 13-7 shows existing sequence numbers, new sequence numbers, and Data statements to be inserted. It should be noted that the sequence numbers are alphameric.

Sequence Numbers and Data Statements to be Inserted	**New Sequence Numbers**
AA010	AA010
AA015	AA015
Data statement 1	AA020
Data statement 2	AA025
Data statement 3	AA030
AA020	AA035
AA025	AA040
AA030	AA045
Data statement 4	AA050
Data statement 5	AA055
Data statement 6	AA060
Data statement 7	AA065
AA040	AA070
AA050	AA075
BB010	BB010
BB015	BB015

Figure 13-7. Sequence Numbers and Seven Data Statements to be Inserted

The control statements are discussed below:

- SYSUT1 and SYSUT2 DD define the partitioned data set PDS.
- SYSIN DD defines the control data set.

- The CHANGE Function statement identifies the input member RENUM. The entire member is to be listed in the message data set.
- The NUMBER Detail statements specify the insert operations (INSERT=YES) and control the renumbering operation.
- Data statements 1, 2, 3, and 4, 5, 6 are the blocks of logical records to be inserted. Because they contain blank sequence numbers, sequence numbers are assigned when the Data statements are inserted.
- Data statement 7, since it contains a sequence number, terminates the insert operation. The sequence number is identical to the number on the next record in the old master data set; consequently, data statement 7 will replace the equally numbered old master record in the output data set.

The existing logical records in this example have sequence numbers AA010, AA015, AA020, AA025, AA030, AA035, AA040, AA045, AA050, BB010, BB015, etc. The insert and renumbering operations are performed as follows:

1. Data statement 1 is assigned sequence number AA020 (NEW1=AA020) and inserted after existing logical record AA015 (SEQ1=AA015).
2. Data statements 2 and 3 are assigned sequence numbers AA025 and AA030 (INCR=5) and are inserted after Data statement 1.
3. Existing logical records AA020, AA025, and AA030 are assigned sequence numbers AA035, AA040, and AA045, respectively.
4. Data statement 4 is assigned sequence number AA050 and inserted. (The SEQ1=AA030 specification in the second NUMBER statement places this Data statement after existing logical record AA030, which has become logical record AA045.)
5. Data statements 5 and 6 are assigned sequence numbers AA055 and AA060 and are inserted after Data statement 4.
6. Existing logical record AA035 is replaced by data statement 7, which is assigned sequence number AA065.
7. The remaining logical records in the member are renumbered until logical record BB010 is encountered. Because this record has a sequence number higher than the next number to be assigned, the renumbering operation is terminated.

IEBUPDTE Example 9

In this example, IEBUPDTE is used to create a sequential data set from card input. User header and trailer labels, also from the input stream, are placed on this sequential data set.

```
//LABEL     JOB   ,MSGLEVEL=1
//CREATION EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD   SYSOUT=A
//SYSUT2    DD   DSNAME=LABEL,VOLUME=SER=123456,UNIT=disk,
// DISP=(NEW,KEEP),LABEL=(,SUL),SPACE=(TRK,(15,3))
//SYSIN     DD   *
./        LABEL
```

(First header label)

(Last header label)

```
./          ADD    LIST=ALL,OUTHDR=ROUTINE1,OUTTLR=ROUTINE2
```

(First input data record)

(Last input data record)

```
./        LABEL
```

(First trailer label)
(Last trailer label)

```
./        ENDUP
/*
```

The control statements are discussed below:

- SYSUT2 DD defines and allocates space for the output sequential data set, which resides on a disk volume.
- SYSIN DD defines the control data set. (This control data set includes the sequential input data set and the user labels, which are on cards.)
- The first LABEL statement identifies the 80-byte card images in the input stream which will become user header labels. (They can be modified by the user's header-label processing routine specified on the ADD Function statement.)
- The ADD Function statement indicates that the Data statements that follow are to be placed in the output data set. The newly created data set is to be listed in the message data set. User output header and output trailer routines are to be given control prior to the writing of header and trailer labels.
- The second LABEL statement identifies the 80-byte card images in the input stream which will become user trailer labels. (They can be modified by the user's trailer-label processing routine specified on the ADD Function statement.)
- ENDUP signals the end of the control data set.

IEBUPDTE Example 10

In this example, IEBUPDTE is used to copy a sequential data set from one direct access volume to another. User labels are processed by user exit routines.

```
                                                                     72
//LABELS    JOB   ,MSGLEVEL=1
//          EXEC PGM=IEBUPDTE,PARM=( MOD, ,MMMMMM )
//SYSPRINT DD    SYSOUT=A
//SYSUT1    DD    DSNAME=OLDMAST,DISP=OLD,LABEL=( ,SUL ),
// VOLUME=SER=111111,UNIT=disk
//SYSUT2    DD    DSNAME=NEWMAST,DISP=( NEW,KEEP ),LABEL=( ,SUL ),
// UNIT=2314,VOLUME=SER=XB182,SPACE=( TRK,( 5,10 ) )
//SYSIN     DD    DSNAME=INPUT,DISP=OLD,LABEL=( ,SUL ),
// VOLUME=SER=222222,UNIT=disk
//
```

The control statements are discussed below:

- SYSUT1 DD defines the input sequential data set, which resides on a disk volume.
- SYSUT2 DD defines the output sequential data set, which will reside on a disk volume.
- SYSIN DD defines the control data set. The contents of this disk-resident data set in this example are:

```
./       REPRO    LIST=ALL,INHDR=SSSSSS,INTLR=TTTTTT,             C
./               OUTHDR=XXXXXX,OUTTLR=YYYYYY
./       ENDUP
```

- The REPRO Function statement indicates that the existing input sequential data set is to be copied to the output data set. This output data set is to be listed on the message data set. The user's label processing routines are to be given control when header or trailer labels are encountered on either the input or the output data set.
- ENDUP indicates the end of the control data set.

IEBUPDTE Example 11

In this example, a partitioned generation consisting of three members is to be used as source data in the creation of a new generation. IEBUPDTE is to be used to add a fourth member to the three source members and to number the new member. The resultant data set is to be cataloged as a new generation.

```
//          JOB
//          EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD    SYSOUT=A
//SYSUT1    DD    DSNAME=A.B.C( 0 ),DISP=OLD
//SYSUT2    DD    DSNAME=A.B.C( +1 ),DISP=( ,CATLG ),UNIT=disk,
// VOLUME=SER=111111,SPACE=( TRK,( 100,10,10 ) ),
// DCB=( RECFM=FB,LRECL=80,BLKSIZE=800 )
//SYSIN     DD    DATA
./  REPRO      NAME=MEM1,LEVEL=00,SOURCE=0,LIST=ALL
./  REPRO      NAME=MEM2,LEVEL=00,SOURCE=0,LIST=ALL
./  REPRO      NAME=MEM3,LEVEL=00,SOURCE=0,LIST=ALL
./  ADD        NAME=MEM4,LEVEL=00,SOURCE=0,LIST=ALL
./  NUMBER     NEW1=10,INCR=5
```

(data cards comprising MEM4)

```
./  ENDUP
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the latest generation, which is used as source data.
- SYSUT2 DD defines the new generation, which is created from the source generation and from an additional member included as input and data.
- The REPRO Function statements reproduce the named source members in the output generation.
- The ADD Function statement specifies that the data cards following the input stream be included as MEM4.
- The NUMBER Detail statement indicates that the new member is to have sequence numbers assigned in columns 73 through 80. The first record is assigned sequence number 10. The sequence number of each successive record is incremented by 5.
- ENDUP signals the end of input card data.

Note: This example assumes that a model DSCB exists on the catalog volume on which the index was built.

IEHATLAS PROGRAM

IEHATLAS is a system utility used with direct access devices when a defective track is indicated by a data check or missing address marker condition.

IEHATLAS can be used to locate and assign an alternate track to replace the defective track. Usable data records on the defective track are retrieved and transferred to the alternate track. A replacement for the bad record is created from data supplied by the user and placed on the alternate track.

In a simple application, IEHATLAS is used as a separate job after an abnormal termination of a problem program. Input data necessary for execution of IEHATLAS—the address of the defective track and replacement records—may be obtained from the dump and from backup data.

A more complex use of IEHATLAS may involve the preparation of a user's SYNAD routine, which reconstructs the necessary input data and invokes IEHATLAS dynamically.

When IEHATLAS is invoked, it attempts to write on the defective track. If the subsequent read-back check indicates that the attempt was successful, a message is issued on the SYSOUT device. If not, a supervisor call routine (SVC 86) is entered automatically.

The SVC routine locates and assigns an alternate track. (If a defective track already has an alternate and an error occurs on that alternate, the SVC routine assigns the next available alternate. All of the valid data records on the defective track are retrieved and transferred to the alternate track. The input record is written on the alternate track in the correct position to recover from the previous error.

When a READ error occurs and a complete recovery is desired, IEHDASDR can be used to produce a listing of error data on a track. Using this data, the input data record for IEHATLAS can be created. The *replace* function can then be performed by executing IEHATLAS.

IEHATLAS supports all current DASD, as listed in the Device Support section of this manual, except the MSS staging packs and virtual volumes.

Input and Output

IEHATLAS uses the following input: (1) a description of the defective track, specifying the cylinder, track, record, key, and data length (in hexadecimal notation), (2) an indication if WRITE Special is needed, and (3) a valid copy (in hexadecimal notation) of the bad record.

IEHATLAS produces as output: (1) a message, issued on the SYSOUT device, containing the user's control information, the input record, and diagnostics, (2) the input record, written on either the original (defective) track or on an alternate track containing the usable data taken from the defective track, and (3) the return parameter list (specifying a maximum of three error record numbers in hexadecimal when an unrecoverable error occurs).

Control

IEHATLAS is controlled by job control statements and utility control statements. The job control statements are used to execute or invoke IEHATLAS and to define the data sets used and produced by IEHATLAS.

A utility control statement is used to specify whether the bad record is a member of the volume table of contents or a member of some other data set. It is also used to indicate whether or not the WRITE Special CCW command is to be used for track overflow records.

Job Control Statements

Figure 14-1 shows the job control statements necessary for using IEHATLAS.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEHATLAS) or, if the job control statements reside in a procedure library, the procedure name.
SYSPRINT DD	Defines a sequential data set that contains the output messages issued by IEHATLAS.
SYSUT1 DD	Defines the data set that contains the bad record.
SYSIN DD	Defines the control data set, which contains the utility control statement and a copy of the bad record.

Figure 14-1.IEHATLAS Job Control Statements

Utility Control Statement

Figure 14-2 shows the utility control statements necessary for using IEHATLAS.

Statement	Use
TRACK	Specifies that an alternate track is to be assigned for a track that does not contain VTOC records.
VTOC	Specifies that an alternate track is to be assigned for a track that contains VTOC records.

Figure 14-2.IEHATLAS Utility Control Statements

TRACK or VTOC Statement

The TRACK or VTOC statement is used to identify the defective record.

Care should be taken to ensure that the input record data length does not exceed the track size. This is especially important when the WRITE Special command is specified because the error may not be recognized immediately by the system.

The TRACK or VTOC statement must not begin in column 1.

Input data (consisting of the hexadecimal replacement record) begins in column 1 immediately following the utility control data. Input data may continue through column 80. As many cards as necessary may be used to contain the replacement record. All columns (1 through 80) are used on the additional cards.

IEHATLAS is designed to replace an error record with a copy of that record. It cannot be used to replace a record with another of a different key and/or data length.

An end-of-file record cannot be changed; therefore, input for key and/or data fields are ignored.

The format of the TRACK or VTOC statement is:

{**TRACK**=*bbbbccccchhhhrrkkdddd*[S] }
{**VTOC**=*bbbbccccchhhhrrkkdddd* }

Return Codes

Return Code	**Meaning**
0	Successful completion; ATLAS has assigned an alternate track.
4	The device does not have software-assignable alternate tracks.
8	All the alternate tracks for the device have been assigned.
12	The requested main storage space is not available.
16	There was an I/O error in the alternate track assignment after N attempts at assignment (where N=10% of the assignable alternate tracks for this device).
20	The error is a condition other than a data check or missing address marker.
24	There is an error in the Format 4 DSCB that prevents ATLAS from reading it.
28	The user-specified error record is the Format 4 DSCB, which ATLAS cannot handle because the alternate track information is unreliable.
32	ATLAS cannot handle the error found in the count field of the last record on the track.
36	There are errors in the Home Address or in Record Zero.
40	ATLAS found one or more errors in record(s) and assigned an alternate track. 1) There was an error on an end-of-file record. 2) ATLAS encountered an error in the count field. 3) There were errors in more than three count fields.
48	ATLAS found no errors on the track and assigned no alternate track.
52	Because of an I/O error, ATLAS cannot reexecute the user's channel program successfully.
56	The system does not support track overflow.
60	The track address provided does not belong to the indicated data set.

Figure 14-3.Return Codes from ATLAS

Operands	Applicable Control Statements	Description of Operands/Parameters
bbbb	TRACK VTOC	*bbbb* This number must be zeros.
cccc	TRACK VTOC	*cccc* is the number of the cylinder in which the defective track was found.
dddd	TRACK VTOC	*dddd* is the data length of the bad record. (When a WRITE Special command is used, *dddd* is the length of the record segment.)
hhhh	TRACK VTOC	*hhhh* is the defective track number.
rrkk	TRACK VTOC	*rrkk* is the record number and key length for the bad record.
S	TRACK	S is an optional byte of EBCDIC information that specifies that the WRITE Special command is to be used (when the last record on the track overflows and must be completed elsewhere).

Restrictions

- The block size for the SYSPRINT data set must be a multiple of 121. The block size for the SYSIN data set must be a multiple of 80. Any blocking factor can be specified.
- DISP=SHR must not be coded on the SYSUT1 DD statement.

IEHATLAS Examples

The following examples illustrate some of the uses of IEHATLAS. Figure 14-4 can be used as a quick reference guide to IEHATLAS examples. The numbers in the "Example" column point to examples that follow.

Operation	Comments	Example
Get Alternate Track	Write Special is included because of a track overflow condition.	1
Get Alternate Track	Alternate track assigned for a bad end-of-file record.	2
Get Alternate Track	Alternate track assigned for a bad VTOC record.	3
Get Alternate Track	Replace defective record zero.	4

Figure 14-4.IEHATLAS Example Directory

Note: Examples which use *disk* in place of actual device-ids, must be changed before use. See the Device Support section, in the Introduction to this manual for valid device-id notation.

IEHATLAS Example 1

In this example, the data set defined by SYSUT1 contains the bad record. An alternate track on the specified unit and volume will be assigned to replace the defective track.

```
//JOBATLAS JOB  06#990,SMITH,MSGLEVEL=1
//STEP      EXEC PGM=IEHATLAS
//SYSPRINT DD   SYSOUT=A
//SYSUT1    DD   DSNAME=NEWSET,UNIT=disk,VOLUME=SER=333333,
// DISP=OLD
//SYSIN     DD   *
 TRACK=0000000200042202000 6S
F3F1C2C2F0F00000
/*
```

The control statements are discussed below:

- SYSPRINT DD defines the device to which the output messages can be written.
- SYSUT1 DD defines the data set that contains the bad record.
- SYSIN DD defines the control data set, which follows in the input stream.
- TRACK specifies the cylinder and track number for the defective track, and the record number, key length, and data length of the bad record. In this example, the input record is to be placed on cylinder two, track four, record 22; it has a key length of two with a logical record length of six. The WRITE Special (S) character is used because there is a track overflow condition.

The input record in this example is a typical hexadecimal record as defined by a TRACK statement. The input record contains eight bytes (data length = 6, key length = 2).

IEHATLAS Example 2

In this example, an alternate track is assigned for a bad end-of-file record.

```
//JOBATLAS JOB  06#990,SMITH,MSGLEVEL=1
//STEP     EXEC PGM=IEHATLAS
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=EOFSET,UNIT=disk,VOLUME=SER=333333,
// DISP=OLD
//SYSIN    DD   *
 TRACK=00000001000003000000
/*
```

The control statements are discussed below:

- SYSPRINT DD defines the device to which the output messages can be written.
- SYSUT1 DD defines the data set that contains the bad record.
- SYSIN DD defines the control data set, which follows in the input stream.
- TRACK defines an end-of-file record on cylinder one, track zero, record three. Input data other than the utility control statement is not required.

IEHATLAS Example 3

In this example, an alternate track is assigned for a bad volume table of contents record.

```
//JOBATLAS JOB  06#990,SMITH,MSGLEVEL=1
//STEP     EXEC PGM=IEHATLAS
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//SYSIN    DD   *
 VTOC=000000000005022C0060
D6C2D1C5C3E34040404040404040404040404040404040404040404040404040404040404040404040
40404040F1F2F3F1F1F0F000014401360000000100000000000000000000000000000000000000000
000040008000000F000F00033333333333333333310DDD00000100000A0000000A00000000000000
0000000000000000000000000000000000000000
/*
```

The control statements are discussed below:

- SYSPRINT DD defines the device to which the output messages can be written.
- SYSUT1 DD defines the data set that contains the bad record.
- SYSIN DD defines the control data set, which follows in the input stream.
- VTOC defines the location of the bad VTOC record as track five of cylinder zero. The record number is 2 with a key length of 44. Record length of the bad record is 96.

The input record in this example is a typical hexadecimal record as defined by the VTOC statement. The input record contains 140 bytes (data length = 96, key length = 44).

IEHATLAS Example 4

In this example, the replacement record is Record 0.

```
//JOBATLAS   JOB 06#990,SMITH,MSGLEVEL=1
//STEP       EXEC PGM=IEHATLAS
//SYSPRINT   DD SYSOUT=A
//SYSUT1     DD DSNAME=NEWSET,UNIT=disk,VOLUME=SER=333333,
// DISP=OLD
//SYSIN      DD *
     TRACK=000000020004000000008
0000000000000000
/*
```

The control statements are discussed below:

- SYSPRINT DD defines the device to which the output messages can be written.
- SYSUT1 DD defines the data set that contains the bad record.
- SYSIN DD defines the control data set, which follows in the input stream.
- TRACK specifies the bin, cylinder, and track number for the defective track, and the record number, key length, and data length of the bad record. In this example, the input record is to be placed on cylinder two, track four, record zero; it has a key length of zero with a logical record length of eight. The input record in this example is a typical hexadecimal record as defined by a TRACK statement. The input record contains eight bytes (data length=8, key length=0).

IEHDASDR PROGRAM

IEHDASDR is a system utility used to prepare direct access volumes for use and to assign alternate tracks on direct access volumes.

In addition, IEHDASDR can be used to dump the entire contents or portions of a direct access volume to a volume or volumes of the same direct access device type, to a tape volume or volumes, to a system output device. Data that is dumped to a tape volume is arranged so that it can subsequently be restored to its original organization by IEHDASDR or IBCDMPRS.

IEHDASDR can be used with volumes containing VSAM and non-VSAM data sets. Information about VSAM data sets can be found in *OS/VS2 Access Method Services*.

Only the special MSS initialize function is allowed on MSS staging packs.

The program can be used to:

- FORMAT: Assign alternate tracks for defective tracks. Write R0 and erase the rest of the track. List the alternate and defective tracks. Then QUICK DASDI to make the direct access volume suitable for operating system use.
- ANALYZE: Analyze tracks, assign alternate tracks for defective tracks, and perform QUICK DASDI functions to make 2314 or 2305 direct access volumes suitable for operating system use.
- LABEL: Change the volume serial number of a formatted direct access volume.
- GETALT: Assign alternate tracks.
- DUMP: Create a backup or transportable copy of a direct access volume, or list the contents on a system output device.
- RESTORE: Copy dumped data from a tape volume to a direct access volume.
- PUTIPL: Install a user-supplied IPL bootstrap and IPL text program on a nonsystem residence DASD volume.

Initializing a Direct Access Volume

IEHDASDR can be used to initialize a direct access volume by either of two methods:

A non-QUICK DASDI:

1. Unassign all alternate tracks.
2. Rewrite the home address and/or record zero (HA/R0) on all tracks.
3. Test flagged defective tracks and recover them if no errors are detected.
4. Assign defective tracks to new, alternate tracks.
5. Perform all other functions of QUICK DASDI.

A QUICK DASDI:

1. Write IPL records on track 0 (records 1 and 2).
2. Write volume labels on track 0 (record 3) and provide space for additional records, if requested (reads alternate tracks and decreases the total count of the alternates by one when an alternate is found defective or assigned).
3. Construct and write a volume table of contents (VTOC).

4. Write an IPL program, if requested, on track 0.
5. Optionally, check for tracks that have been previously designated as defective (flagged) and have had alternate tracks assigned.
6. Optionally, write a track descriptor record (record 0) and erase the remainder of each track. May also attempt to reclaim any track that has the defective bit on in the flag byte of the home address.

IEHDASDR can be used to format 3350 devices, a modified surface analysis will be defaulted for OFFLINE ANALYZE (PASSES=1). The analysis procedure will be:

- Unassign all alternates
- Rewrite the home-address and record-zero (HA/R0) on all tracks
- Perform surface analysis on previously flagged defective tracks and reclaim them if no errors are detected, otherwise, assign an alternate.
- Write a volume label, VTOC, and IPL text, if supplied.

Figure 15-1 shows a direct access volume after it has been prepared for use. A direct access volume can be initialized in this manner using IEHDASDR.

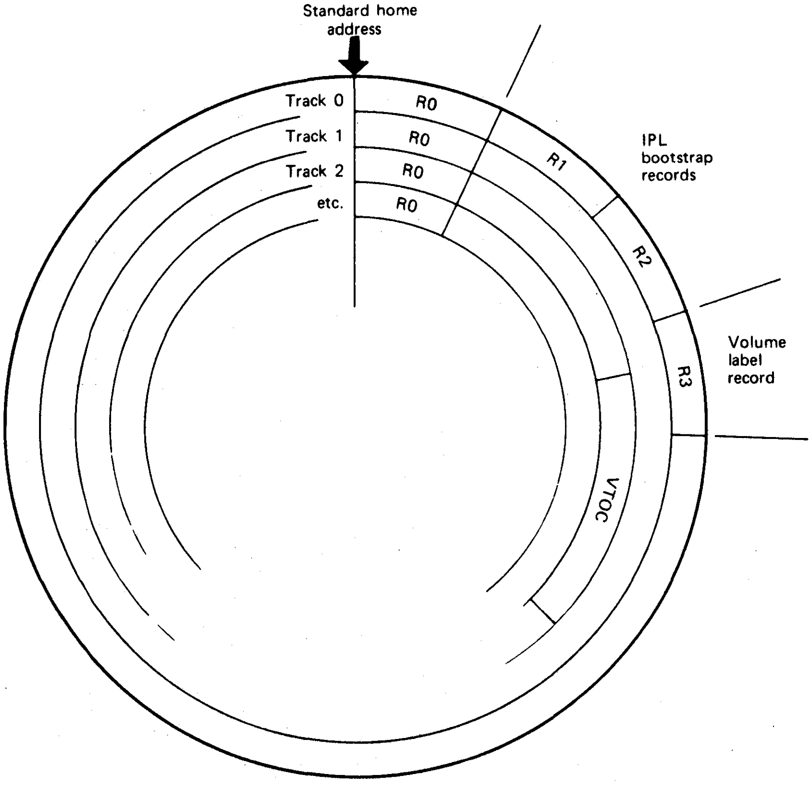


Figure 15-1. Direct Access Volume Initialized Using IEHDASDR

IEHDASDR can be used to attempt to reclaim tracks on a 3340/3344 device with the defective bit on in the home address (HA) flag byte. When the ANALYZE function is executed on these devices, with the FLAGTEST=NO option, the volume is scanned for flagged defective tracks.

- When a track is found flagged (defective), the track is surface analyzed.
- If no defect is indicated, the track is returned to service with the defective bit off in the home address flag byte and a standard R0 is written.
- A defective primary track is assigned the next available alternate.

If over a period of time the same track on a particular 3340/3344 shows a history of failures, or has been flagged by the manufacturer, the track is probably marginal and should be assigned an alternate even if no error occurs on the surface analysizs.

Initialize—MSS Staging Volumes

IEHDASDR can be used to prepare a 3330 or 3330-1 volume for use with MSS as a staging pack. The format of the staging pack is as follows:

	3330	**3330-1**
Primary tracks	0-408	0-808
Alternate tracks	409,410	809-814

Note: A 1-track VTOC will be written on track 2 of cylinder 0 with a format 5 DSCB that indicates no free tracks.

Changing the Volume Serial Number of a Direct Access Volume

IEHDASDR can be used to change the volume serial number of an initialized direct access volume. Optionally, a one- to ten-character owner name can be placed in the volume label record (record 3 of track 0). If an owner name already exists, it is overwritten with the new name.

Note: All cataloged data sets residing on a volume whose label is changed must be recataloged if the catalog reflects the old serial number.

Assigning Alternate Tracks for Specified Tracks

IEHDASDR can be used to assign an alternate track on a disk volume. An alternate track can be assigned for any track, whether it is defective or not. If the specified track is an alternate, a new alternate is assigned; if the specified track is an unassigned alternate, it is flagged to prevent its future use.

For 3350 volumes only, surface analysis will be performed to determine if the track is defective. Alternates will be assigned only if an error is detected.

Creating a Backup, Transportable, or Printed Copy

IEHDASDR can be used to dump a direct access volume or a portion of a volume to any number of tape volumes or volumes of the same direct access device type, or to a system output device. The program can dump a single track, a group of tracks, or an entire volume.

When an entire volume is dumped:

- All primary tracks (for which no alternate tracks are assigned) are dumped.
- When a defective primary track is found, the alternate track is dumped in place of the primary track.

Each track to be dumped will have all of its data except the home address (HA) and the count field of record zero (R0) copied to the receiving volume. The dump function of IEHDASDR is dependent on the validity of the Count field of every record on the track being dumped. The results of reading an erroneous R1 count field are unpredictable, while R2 through Rn will cause the dump function to terminate.

A receiving direct access volume retains its own serial number unless the user specifies that it is to be assigned the serial number of the direct access volume being dumped.

Except for a printing operation, only data that is owned is dumped; IEHDASDR checks the first or only Free Space (Format 5) data set control block (DSCB) in the volume table of contents. The Free Space (Format 5) DSCB identifies unowned (unused) space on the direct access volume. Whenever an unowned track is encountered, a dummy record, containing a home address and record zero, is written on the receiving volume. When data is dumped to a system output device, the entire range of specified tracks is dumped.

A printing operation prints each record in hexadecimal. In addition, all printable characters are also represented in EBCDIC.

Figure 15-2 shows the format of printed output. Each track is identified by its absolute track address (cccchhhh). The R0 data field is printed on the same line as the track address. Each printed record is preceded by a count field that identifies the applicable track address (cccchhhh), the record number of the record being printed (rr), and the key and data length (kk and dddd) of the record.

If an alternate track is printed in place of a primary track, it is identified in the printout by the primary track address.

Copying Dumped Data to a Direct Access Volume

When a direct access volume is dumped to a tape volume, the data is placed in a format that is specially suited for the tape volume. IEHDASDR can be used to restore the format of the dumped data and place the data on the same type of direct access volume as the original volume; that is, data originally dumped from a 2314 volume can be restored to a 2314 volume, etc.

Identical copies of dumped data can be restored to any number of volumes of the same direct access device type as the original volume during the execution of a single restore operation. In addition, data that was dumped by IBCDMPRS can be restored.

A receiving direct access volume retains its own serial number unless the user specifies that it is to be assigned the serial number of the direct access volume

```
*** TRACK  cccchhhh      RO DATA  xxxxxxxxxxxxxxxx

    COUNT  cccchhhhrrkkdddd
                        key and data fields                                        key and data fields

                          (hexadecimal)                                                 (EBCDIC)

000000   xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx   xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx  *... .................*
000032   xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx   xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx  *... .................*
                                etc.

    COUNT  cccchhhhrrkkdddd

000000   xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx   xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx  *... .................*
000032   xxxxxxxx xxxxxxxx xxxxxxxx etc.

*** TRACK  cccchhhh      RO DATA  xxxxxxxxxxxxxxxx

    COUNT  cccchhhhrrkkdddd

000000   xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx   xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx  *... .................*
000032   xxxxxxxx xxxxxxxx etc.
```

Figure 15-2. Format of a Direct Access Volume Dumped to a Printer Using IEHDASDR originally dumped. If multiple direct access volumes are to be dumped to, and the user specifies that the serial number of the dumped volume is to be propagated, all receiving volumes are assigned that serial number.

Dumping and Restoring Unlike Devices

With the 3330, 3330-1, and 3340, you have the capability of upward device migration. That is, a 3330 can be dumped or restored to a 3330-1 volume, but a 3330-1 cannot be dumped or restored to a 3330. Likewise, a 3340, 35-megabyte model can be dumped or restored to a 3340, 70-megabyte model, but the 70-megabyte model cannot be dumped or restored to the 35-megabyte model.

If the input volume contains a VSAM catalog or VSAM data sets, upward migration with IEHDASDR to a different device type should not be done due to device-dependent information in the VSAM catalog. The Access Method Services utility must be used to move the VSAM catalog or data set.

When any of these device migration functions are performed, the 'DOS' bit in the receiving volume's Format 4 DSCB is set to indicate the Format 5 DSCB is incorrect. It is recommended that a job step be executed to allocate a temporary data set for the receiving volume to cause the DADSM function to reset the DOS bit and correct the Format 5 DSCB.

Formatting a Direct Access Volume

IEHDASDR can be used to format a direct access volume. A volume can be formatted to:

- Check a direct access volume for previously flagged tracks. No formatting is performed on known defective tracks. The defective and the alternate tracks are printed.
- Format each track by writing R0 and erase the rest of the track.
- Assign alternate tracks for defective tracks.
- Construct IPL bootstrap records (records 1 and 2 of track 0), a volume label record (record 3 of track 0), and a volume table of contents (VTOC), whose size and placement are determined by the user.
- Optionally, write an IPL program record and provide owner information in the volume label record.

Writing IPL Records with the PUTIPL Function

IEHDASDR can be used, via the PUTIPL function, to write user-supplied IPL bootstrap records and an IPL program on cyliner 0, track 0, of any initialized DASD volume, other than the system residence volume. See Figure 15-1.

The contents of the IPL records and the contents of the records that make up the program are not checked by IEHDASDR. It is the user's responsibility to ensure that the IPL records can load an executable program.

The first IPL record must contain a PSW followed by two CCWs (channel command words). The CCWs must have the following hexadecimal formats:

First CCW:	06xxxxxx60000090
Second CCW:	08xxxxxx00000000

The first CCW is a command to read in the second IPL record at main storage address xxxxxx. The second CCW is a transfer-in-channel command (a branch) to the CCW that begins the second IPL record.

The second IPL record must be a 144-byte channel program. Bytes 32 to 42 of this record must contain zeros.

The program may consist of:

- One record, not longer than 3K (3072) bytes.
- Two records, neither longer than 3K (3072) bytes.
- Three records, none longer than 2K (2048) bytes.

Figure 15-3 shows an input data set with three program records.

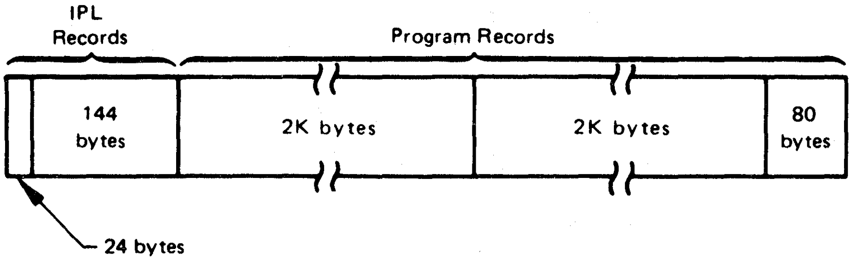


Figure 15-3. Input Data Set With Three Program Records

If the output volume does not contain user labels, IEHDASDR writes program records after the volume label record. Figure 15-4 shows where program records are written when the output volume does not contain user labels.

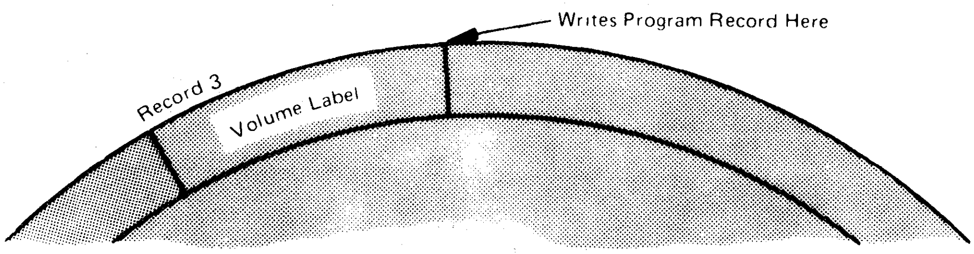


Figure 15-4. Cylinder 0, Track 0 Fragment Without User Labels

If user labels have been written after the volume label, the user can specify that IEHDASDR:

- Write over the user labels.
- Put the program records *after* the user labels when a non-2314 volume is used.

Figure 15-5 shows program records to be written after user labels.

The following errors are possible when using IEHDASDR PUTIPL function to write IPL records and a program on a direct access volume:

- A 2314 output volume contains user labels, but the user has not specified that the user labels are to be overwritten.
- The total input (IPL records and program) consists of fewer than three records.

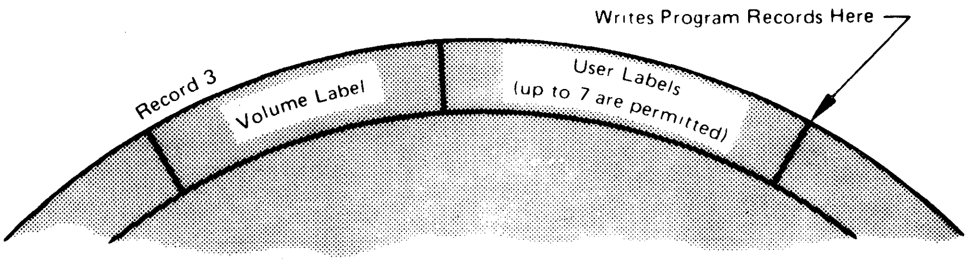


Figure 15-5. Cylinder 0, Track 0 Fragment With User Labels

- The first and second IPL records are not 24 bytes and 144 bytes in length, respectively.
- A third program record is longer than 2K bytes.
- The output device is not a direct access device.
- The output volume contains a VTOC on cylinder 0, track 0.
- The output volume is the system residence volume.

Input and Output

IEHDASDR uses as input a control data set containing utility control statements, and optionally, IPL text.

The primary output or result of executing IEHDASDR is determined by the application. A sequential message data set is created to list informational messages (for example, control statements used), dumped data (for a print operation), and any error messages.

IEHDASDR provides a return code to indicate the results of program execution. The return codes and their meanings are:

- 00, which indicates successful completion.
- 04, which indicates that an unusual condition was encountered; however, the overall result is successful. A warning message is issued.
- 08, which indicates that a specified operation did not complete successfully. An attempt is made to perform any additional operations.
- 16, which indicates that either an error occurred upon invoking IEHDASDR, or IEHDASDR was unable to open the input or message data set. The job step is terminated.

Control

IEHDASDR is controlled by job control statements and utility control statements. The job control statements are used to execute or invoke IEHDASDR and define the data sets used and produced by IEHDASDR. The utility control statements are used to control the functions of the program.

Job Control Statements

Figure 15-6 shows the job control statements necessary for using IEHDASDR.

Statement	**Use**
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEHDASDR) or, if the job control statements reside in a procedure library, the procedure name. Additional information can be entered in the PARM parameter of the EXEC statement; see "PARM Information on the EXEC Statement" below.
STEPCAT	Is required when a volume contains a VSAM data set which is not cataloged on the master catalog.
SYSPRINT DD	Defines a sequential message data set. The data set can be written to a system output device, a tape volume, or a direct access device.
anyname DD	Defines a direct access device type.
tapename DD	Defines a magnetic tape unit.
SYSIN DD	Defines the control data set. The control data set usually resides in the input stream; however, it can be defined as a blocked or unblocked sequential data set or as a member of a procedure library.

Figure 15-6. IEHDASDR Job Control Statements

The "anyname" DD statement can be entered:

```
//anyname DD UNIT=xxxx,VOLUME=SER=xxxxxx,DISP=OLD
```

If more than one volume is to be processed on a single mountable device, deferred mounting can be specified in the "anyname" DD statement by entering:

```
//anyname DD UNIT=(xxxx,,DEFER),VOLUME=(PRIVATE,...),
// DISP=(NEW,KEEP)
```

The "anyname" DD statement is not used for an operation that analyzes an offline direct access volume.

If the volume serial number of a volume to be processed online is not known, it may be possible to make a nonspecific, PRIVATE volume request on a specific unit; for example:

```
//anyname DD UNIT=(191,,DEFER),VOLUME=PRIVATE,DISP=(NEW,KEEP),
// SPACE=(TRK,(1,1))
```

In this case, the operator is asked to mount a scratch volume on that unit. See "Appendix C: DD Statements for Defining Mountable Devices" for the appropriate DD statement and for a discussion of how to make a nonspecific unit request.

If an IEHDASDR operation produces a volume serial number that is a duplicate of a volume serial number already allocated within the system, the volume to which the duplicate number is assigned is made unavailable to the system. The operator is asked to remove the applicable volume at the completion of the operation.

The "tapename" DD statement can be entered:

```
//tapename DD UNIT=xxxx,VOLUME=SER=xxxxxx,LABEL=(...,...),
// DISP=(....,KEEP),DCB=(TRTCH=C,DEN=x)
```

If more than one tape volume is to be processed on the same tape unit, deferred mounting can be specified by:

```
//tapename DD UNIT=(xxxx,,DEFER),VOLUME=(PRIVATE,...)
```

If standard labeled tapes are specified, the DSNAME should also be provided.

The “anyname” DD and “tapename” DD statements are referred to by utility control statements for program operation.

Both the SYSIN and the SYSPRINT data set can have a blocking factor of other than 1.

PARM Information on the EXEC Statement

The EXEC statement for IEHDASDR can contain PARM information that is used by the program to control line density on output listings and to indicate the maximum number of operations of the same type that can be performed concurrently in the job step.

The EXEC statement can be coded:

```
// EXEC PGM=IEHDASDR {,PARM=‘N=n’              }
                     {,PARM=‘LINECNT=xx’       }
                     {,PARM=‘LINECNT=xx,N=n’ }
```

The LINECNT value specifies the number of lines per page in the listing of the SYSPRINT data set. The number *xx* is a 2-digit decimal number ranging from 01 to 99. If LINECNT is omitted, the number of lines per page is 58.

The N value specifies a decimal number from one to six that represents the maximum number of like functions that can be performed concurrently by IEHDASDR, assuming that adequate system resources are available. If N is omitted, up to six ANALYZE, FORMAT, DUMP, or RESTORE operations are performed concurrently—according to the number of successive like statements in the input stream. (See “Utility Control Statements.”)

System resources permitting, multiple output copies can be specified in any or all of the concurrent operations. For example, if N=2 and four DUMP statements appear in succession, the first two dump operations are performed concurrently. As each dump operation is completed and system resources become available, a new dump operation begins.

Considerations

To reformat native 3330, 3330-1, or 3340 VM packs to OS/VS format, use the FORMAT function.

To reformat emulated 3330, 3330-1, or native 3350 VM packs to OS/VS format, use the ANALYZE (offline) function.

Data can be dumped from the system residence volume (the IPL volume); however, this is the only IEHDASDR operation that can be performed on that volume.

Because IEHDASDR can change serial numbers and existing data on a direct access volume, operating precautions must be followed by users who have two or more central processing units sharing the same direct access volume.

If IEHDASDR is run in a multiprogramming environment, you must choose a combination of DD statements (defining mountable devices) that will ensure that volume integrity is maintained. Refer to “Appendix C: DD Statements for Defining Mountable Devices.”

If non-VSAM password-protected data sets reside on volumes that are used by IEHDASDR, the following considerations must be made:

- When dumping from a volume containing read password-protected data sets, each data set must be described in a separate DD statement having a unique ddname. When the program is executed, the operator must supply the correct password (in answer to a console message) for each password-protected data set.

- When dumping to a tape volume from a direct access volume containing non-VSAM password-protected data sets, the DD statement defining the tape volume must include a DSNAME parameter. In addition, the LABEL parameter must define a standard labeled tape, include a PASSWORD subparameter, and specify or imply a file number of 1.
- When restoring from a tape volume, a DSNAME parameter must be included in the DD statement defining the tape volume.
- During the DUMP, RESTORE, ANALYZE, and FORMAT functions (see "Utility Control Statements"), the direct access "TO" volume is checked for password-protected data sets. At this time the operator must supply the correct password for each password-protected data set encountered.

Refer to *OS/VS2 System Programming Library: Data Management* for additional information on non-VSAM data set password protection.

If VSAM data sets reside on volumes that are used by IEHDASDR, the following considerations must be made:

- All VSAM data spaces are described by a Format-1 DSCB which indicates that the data set is password protected. Therefore, the catalog in which the data space is defined must be identified to IEHDASDR by a STEPCAT DD statement or defaulted to the master catalog, whether or not any VSAM data set is password protected.
- The catalog master password or the VSAM data set master password must be supplied by the operator for all VSAM password-protected data sets within each data space.
- A separate DD statement for each VSAM data set is not required as is the requirement for non-VSAM password-protected data sets.
- When no non-VSAM password-protected data sets reside on a volume, the restore tape(s) need not be password protected.
- PURGE=YES option must be specified on the RESTORE control card, if the receiving volume of a restore operation contains VSAM data spaces.

Refer to *OS/VS2 Access Method Services* for additional information on VSAM data set password protection.

If RACF-protected data sets reside on volumes that are used by IEHDASDR, the following considerations apply:

- The user must have ALTER access authorization to all RACF-defined data sets on the TODD volume of a FORMAT, ANALYZE, full RESTORE, and full DUMP operation.
- The user must have UPDATE access authorization to all RACF-defined data sets on the TODD volume of a partial DUMP or partial RESTORE.
- The user must have READ access authorization to all RACF-defined data sets on the FROMDD volume of a DUMP.
- The user may bypass expiration date, password, and RACF data set access authority checking for RACF-defined volumes if the user has the proper volume access authority. Figure 15-7 indicates the required DASD volume access authority for the various functions.

Function/Option	**Authorization Required for RACF-Defined DASD Volume IDs**	
	FROMDD	**TODD**
DUMP	READ	ALTER
RESTORE	READ[1]	ALTER
ANALYZE	—	ALTER
FORMAT	—	ALTER
GETALT	—	ALTER
PUTIPL	—	UPDATE
LABEL	ALTER	ALTER
DUMP with CPYVOLID = YES	ALTER	ALTER
RESTORE with CPYVOLID = YES	ALTER[1]	ALTER

[1] RACF authorization refers to the DASD volume ID used during the creation of the tape. RACF authorization for the tape volume ID is optional.

Figure 15-7. RACF Authorization Required for IEHDASDR Functions

IEHDASDR can perform up to six concurrent operations of ANALYZE, FORMAT, DUMP, or RESTORE operations (see "Utility Control Statements"). This feature, which can shorten the time required to execute the program, is controlled by (1) the number of devices defined for use and (2) the physical arrangement of utility control statements in the input stream. For example, assuming that the required devices are defined and available, a combination of six successive statements of the same type permits six concurrent operations to take place. However, if the utility control statements are arranged so that no operations of the same type appear in succession, no operations are performed concurrently, even though many devices might be defined for use.

Note: The number of concurrent operations allowed can be overridden by an EXEC statement PARM value.

Utility Control Statements

Figure 15-8 shows the utility control statements necessary for using IEHDASDR.

For most operations, multiple copies of a source volume can be made. The program can also perform from two to six ANALYZE, FORMAT, DUMP, or RESTORE operations concurrently, according to the number of successive like statements in the input stream; that is, up to six direct access volumes can be analyzed or formatted, or dumped simultaneously, or up to six magnetic tape (restore) volumes can be processed simultaneously.

Statement	Use
ANALYZE	Analyzes the recording surface to test for defective tracks (2314 and 2305 only), assigns alternates for any defective tracks found, and initializes the volume to make it ready for use. Format 3350 volumes in 3350 or 3330 or 3330-1 mode.
ANALYZE MSS	Analyzes the recording surface of a MSS device to test for defective tracks, assigns alternates for any defective tracks found, and formats the volume to make it ready for use.
FORMAT	Write R0 on each track and initialize the volume to make it ready for use.
LABEL	Changes the volume serial number of a direct access volume and, optionally, updates the owner field.
GETALT	1)Test a track and, if necessary, assign an alternate (2314 and 3350 only), or 2) bypass testing and assign an alternate.
DUMP	Dumps a single track, a group of tracks, or an entire direct access volume.
RESTORE	Restores a previously dumped direct access volume to a direct access device.
IPLTXT	Signals the beginning of IPL program text statements.
PUTIPL	Specifies that IPL records and a program are to be written on a direct access device.

Figure 15-8. IEHDASDR Utility Control Statements

ANALYZE Statement

The ANALYZE statement is used to analyze the recording surface of a 2314 or 2305. Bit patterns are written on a track, read, and tested for defects. If no defects are found, the track is formatted to make it ready for system use.

An IEHDASDR job to initialize a Buffered-log device will not perform a surface analysis. The ANALYZE option can also perform a "QUICK DASDI."

When the ANALYZE option is performed on a 3340 with FLAGTEST=NO, an attempt is made to reclaim any track that has the defective bit on in the flag byte of the home address.

Note: If the device is online the volume label and VTOC are read, and the information contained in them is used to initialize the volume. If the device is offline, the volume label and VTOC information is ignored.

The format of the ANALYZE statement is:

[*label*] **ANALYZE** **TODD=** {(*cuu*,...) | (*ddname*,...)}
,VTOC=*xxxxx*
,EXTENT=*xxxxx*
[**,NEWVOLID=***serial*]
[**,IPLDD=***ddname*]
[**,FLAGTEST=** {**YES** | **NO**}]
[**,PASSES=** {*n* | 0}]
[**,OWNERID=***name*]
[**,PURGE=** {**YES**{**NO**}]

ANALYZE MSS Statement

This statement is used to allow IEHDASDR to prepare a standard 3330 or 3330-1 volume for use as an MSS staging pack. Cylinders 409 and 410 for 3330 and 809 through 814 for 3330-1 will be assigned as alternates. Defective primary tracks will be reassigned to this alternate area. A one track VTOC will be written on track 2 with a format 5 DSCB that indicates no free tracks.

The volume must be offline and the NEWVOLID must be specified. If other ANALYZE parameters are specified, they will be ignored.

The format of the ANALYZE MSS statements is:

[*label*] **ANALYZE**	**TODD=**{(*cuu*,...)}
	,NEWVOLID=*serial*
	,MSS
	[**,OWNERID=***name*]

Note: To prepare an MSS staging pack for non-MSS use, an offline ANALYZE followed by a FORMAT should be performed.

FORMAT Statement

The FORMAT statement is used to prepare a volume for operating-system use. Except for flag testing, no analysis is made prior to formatting a track. Previously flagged disk tracks remain flagged; alternate tracks are assigned, where applicable.

The output includes a list of defective tracks and their assigned alternates.

Note: If a command reject is detected while a FORMAT operation is performed on an assigned alternate track on an IBM 2305 Fixed Head Storage volume, processing continues as if no alternate track existed. No action need be taken if message IEH400I is typed out on the operator's console in response to this condition.

If FORMAT cannot read a home address, it flags the track as being defective and assigns an alternate track.

The format of the FORMAT statement is:

[*label*] **FORMAT**	**TODD=(** *ddname*,...)
	,VTOC=*xxxxx*
	,EXTENT=*xxxxx*
	[**,NEWVOLID=***serial*]
	[**,IPLDD=***ddname*]
	[**,OWNERID=***name*]
	[**,PURGE=**{**YES** | **NO**}]

LABEL Statement

The LABEL statement is used to change the serial number of a direct access volume and, optionally, to update the owner field in record 3 of track 0. One LABEL statement must be included for each volume that is to have its label changed.

The format of the LABEL statement is:

[*label*] **LABEL** **TODD=** {*cuu* | *ddname*}
,NEWVOLID=*serial*
[**,OWNERID=***name*]

GETALT Statement

The GETALT statement is used to assign an alternate track for a specified disk track if the volume was previously initialized.

For 3350 volumes, alternate tracks will be assigned only if an error is detected during surface analysis.

Flags set by GETALT statement, for 3330 or 3330-1, tracks, cannot be removed by IEHDASDR.

The format of the GETALT statement is:

[*label*] **GETALT** **TODD=***ddname*
,TRACK=*cccchhhh*

DUMP Statement

The DUMP statement dumps a single track, a group of consecutive tracks, or an entire direct access volume to one or more direct access volumes of the same device type, to one or more tape volumes, or to a system output device (printer assumed). When dumping more than one file to the same tape volume, the tape is rewound to the load point at the end of each dump operation.

An extra input/output error (data check) message is generated at the console when the dump to SYSPRINT function encounters one of the following conditions:

- Missing address marker.
- Data check in count and key fields and/or data field.
- Input/output error on a search command.
- Missing address marker and no record found.

The additional data check message printed at the console is generated by the dump function's error recovery procedure. However, the additional message is not reflected by a SYNADAF message in the SYSPRINT data set. If a missing address marker is encountered during a space count command, the function terminates with a return code of 8.

Note: If multiple output volumes are specified in a DUMP statement and abnormal completion of the DUMP operation occurs, the operation is terminated on all output volumes.

Do not dump a volume and restore new data to that volume in the same job step. IEHDASDR does not *flush* the input stream if an operation is unsuccessful; that is, the program attempts to perform any remaining functions after encountering an error. Thus, if a dump operation is unsuccessful, data is lost if a subsequent restore operation places new data on the dumped volume.

Partial dumps of direct access volumes should be used with extreme caution. Because only those tracks that are dumped are placed on the receiving volume, the partially dumped data may not be usable. When partially dumped data is subsequently restored, it is placed on the same tracks as it originally occupied.

When using the DUMP statement, do not specify the same ddname in more than one TODD parameter in a single job step, except when the ddname is SYSPRINT.

When space permits, more than one direct access volume can be dumped to a restore tape. Each dumped volume will be handled as a separate file.

When dumping to or restoring from a tape, specified as standard label or "BLP", a disposition of KEEP should be specified in the DD statement for the tape. Unlabeled tapes may have other disposition parameters.

When restoring from a restore file on a tape, the same file sequence number and tape label format used in the dump operation must be used.

Intermixing of restore files with system data sets is not recommended because of the unique format of the restore file.

The format of the DUMP statement is:

[*label*] **DUMP** **FROMDD**=*ddname*
,TODD=(*ddname*,...)
[**,CPYVOLID**= {**YES** | **NO**}]
[**,BEGIN**=*cccchhhh*]
[**,END**=*cccchhhh*]
[**,PURGE**={**YES** | **NO**}]

RESTORE Statement

The RESTORE statement is used to restore a direct access volume or volumes from a tape volume on which a dumped copy was previously placed.

Note: When a standard labeled restore tape created by IBCDMPRS is restored by IEHDASDR, the DD card describing the tape for IEHDASDR can specify LABEL=(2,BLP).

The format of the RESTORE statement is:

[*label*] **RESTORE** **TODD**=(ddname,...)
,FROMDD=*ddname*
[**,CPYVOLID**= {**YES** | **NO**}]
[**,PURGE**={**YES** | **NO**}]

IPLTXT Statement

The IPLTXT statement is used to mark the beginning of IPL program text statements. An ANALYZE or FORMAT statement must precede this statement.

IPL text need be included only once in the input stream; that is, IEHDASDR refers to the first copy of IPL text encountered when performing multiple functions in a single job step.

The format for the IPLTXT statement is:

[*label*] **IPLTXT**

PUTIPL Statement

The PUTIPL statement specifies that IPL bootstrap records and a program are to be read from an input data set and written to cylinder 0, track 0 of a direct access volume. As a result, cylinder 0, track 0 of the output volume will contain a program that the user should be able to load from the console.

Note: If the PUTIPL function of IEHDASDR is used to write IPL records, the user must supply both the IPL bootstrap records and the IPL text program. The contents of the bootstrap records and the IPL text program are not checked by IEHDASDR. The IPL text on SAMPLIB cannot be used, unless the user also supplies the bootstrap records ahead of the IPL text.

The format of the PUTIPL statement is:

[*label*]	**PUTIPL**	**FROMDD=***ddname*
		,TODD=*ddname*
		[,PURGE={YES | <u>NO</u>}]

Operands	Applicable Control Statements	Description of Operands/Parameters
BEGIN	DUMP	**BEGIN=***cccchhhh* specifies in hexadecimal a cylinder number, *cccc* and head number, *hhhh,* that identify the first track to be dumped. If BEGIN is omitted, the dump operation begins with track 0. **Default:** Dump begins with track 0.
CPYVOLID	DUMP RESTORE	**CPYVOLID={YES | NO}** specifies whether receiving direct access volumes are to be assigned the serial number of the dumped volume. **YES** specifies that all receiving direct access volumes are to be assigned the serial number of the dumped volume. **NO** specifies that receiving volumes are to keep their own serial numbers.
END	DUMP	**END=***cccchhhh* specifies in hexadecimal a cylinder number, *cccc,* and head number, *hhhh,* that identify the last track to be dumped. If only one track is to be dumped, both BEGIN and END specify that track address. **Default**: The last primary track of the volume is the last track to be copied. (Alternate tracks are not dumped unless they are assigned as alternates.)
EXTENT	ANALYZE FORMAT	**EXTENT=***xxxxxx* specifies the decimal length of the VTOC in tracks. The VTOC cannot extend into the alternate track area or to a second volume.
FLAGTEST	ANALYZE	**FLAGTEST={YES | NO}** specifies whether a check is to be made for previously flagged tracks. The default changes to NO for OFFLINE initialization of 2314 or 2305 volumes. FLAGTEST is not applicable to 3330, 3330-1, or 3350 volumes. **YES** specifies that each track is to be checked to see whether it was previously flagged as defective. Alternate tracks are re-assigned. **NO** specifies that surface analysis will be performed without a check for previously flagged tracks on 2305 or 2314 volumes. On 3340 volumes, "PASSES=1" is forced and analysis is performed on each flagged (defective) track.

Operands	Applicable Control Statements	Description of Operands/Parameters
FROMDD	DUMP RESTORE PUTIPL	**FROMDD**=*ddname* • specifies the ddname of the DD statement defining the device containing the direct access volume from which a copy or copies are to be made (for DUMP). • specifies the ddname of the DD statement that defines the tape volume containing the data to be restored. If more than one tape volume is to be used as input, the DD statement for the tape must indicate multiple volume (for RESTORE). • specifies the ddname of the DD statement that identifies the input data set. The DD statement must contain the DSNAME and DISP parameters and, if the input data set is not cataloged or passed from an earlier step, the VOL and UNIT parameters (for PUTIPL).
IPLDD	ANALYZE FORMAT	**IPLDD**=*ddname* specifies the ddname of a DD statement defining the data set containing the IPL program. The IPL program can be included in the SYSIN (input stream) data set, or it can be defined as a sequential data set or a member of a partitioned data set. If IPL text is included in the input stream, an IPLTXT statement is used to separate the ANALYZE statement from the IPL program text statements. Maximum IPL record size is restricted to 6,496 bytes.
MSS	ANALYZE MSS	**MSS** specifies an MSS staging pack is to be prepared.
NEWVOLID	ANALYZE ANALYZE MSS FORMAT LABEL	**NEWVOLID**=*serial* specifies a one- to six-character serial number. The serial number is assigned to all direct access volumes processed through the use of this control statement. This parameter is required for the analysis of a volume offline. **Default:** The direct access volumes retain their own serial numbers.
OWNERID	ANALYZE ANALYZE MSS FORMAT LABEL	**OWNERID**=*name* specifies a one- to ten-character name or other identifying information to be placed in the volume label record. OWNERID is specified as a character string of any alphameric, national character, hyphen (-), slash (/), or period (.).

Operands	Applicable Control Statements	Description of Operands/Parameters
PASSES	ANALYZE	**PASSES=** {n | 0} For 2305 or 2314: specifies the number of passes to be made in analyzing a recording surface. These values can be coded: *n* specifies the number of times a bit pattern test is to be performed. The *n* value is a decimal number from 1 to 255. 0 specifies that the ANALYZE function is to perform a QUICK DASDI. **Default:** The bit pattern test is performed once on each track. For 3330: PASSES=1 is not applicable; PASSES=0 indicates do a QUICK DASDI. For 3340: PASSES is not applicable. For 3350: PASSES=1 (ONLINE) is not applicable; PASSES=1 (OFFLINE) write-HA and R0 on each track to convert volume format to 3330, 3330-1, or 3350 mode. Test all defective (flagged) tracks and recover (unflag) those that pass the surface analysis test. PASSES=0—do a QUICK DASDI.

Operands	Applicable Control Statements	Description of Operands/Parameters
PURGE	ANALYZE FORMAT DUMP RESTORE PUTIPL	**PURGE={YES | NO}** specifies whether the ANALYZE, FORMAT, DUMP, or RESTORE operations are to be terminated when an unexpired data set is encountered, or, for PUTIPL only, specifies whether user labels are to be overwritten. PURGE does not apply when dumping to a restore tape. PURGE=YES must be specified if the receiving volume of a restore operation contains VSAM data spaces. If PURGE is omitted and an unexpired data set is encountered, the ANALYZE, FORMAT, DUMP, or RESTORE operations are terminated. These values can be coded: **YES** indicates that all unexpired data sets on the volume can be overwritten provided that the operator signals his concurrence when the first unexpired data set is encountered, or, for PUTIPL only, specifies that the program may be written over any user labels or over any data that follows the volume label record. If PURGE=YES is coded and an unexpired data set is encountered, the operation is prompted. The operator replies are: • U, which indicates that all unexpired data sets on this volume can be overwritten. • T, which indicates that this volume contains unexpired data sets that must not be overwritten. **NO** specifies that the various operations are to be terminated if an unexpired data set is encountered, or, for PUTIPL only, specifies that the program may not be written over standard user labels. If the output device is a 2305 or Buffered-log DASD, the program is written following any standard user labels. If the output volume contains user labels and the device is a 2314, there may not be enough space on the track for the IPL program; in that case, the write operation is terminated. The PURGE parameter does not apply to password-protected data sets; that is,, the operator must always respond with the proper password for each password-protected data set encountered. If he is unable to do so, the operation is terminated.

<table>
<thead>
<tr><th>Operands</th><th>Applicable Control Statements</th><th>Description of Operands/Parameters</th></tr>
</thead>
<tbody>
<tr>
<td>TODD</td>
<td>ANALYZE
ANALYZE MSS
FORMAT
LABEL
GETALT
DUMP
RESTORE
PUTIPL</td>
<td>For ANALYZE and LABEL:
TODD={(<i>cuu</i>,...) | (<i>ddname</i>,...)}
For ANALYZE MSS:
TODD={<i>cuu</i>,...}
For FORMAT, DUMP, and RESTORE:
TODD=(<i>ddname</i>),...)
For GETALT and PUTIPL:
TODD=<i>ddname</i>
specifies the ddname for the volume to be processed. These values can be coded:
(<i>ccu,...</i>) or <i>cuu</i>
specifies the channel and unit address of a direct access device containing a volume to be initialized or labeled. This value is used only if the volume is offline, which includes the first analysis of a volume or for labeling an offline volume. If this value is coded, a DD statement defining the device must not be provided. The specified devices must be varied offline (by use of the VARY OFFLINE command) prior to the execution of the job step.
(<i>ddname</i>,...) or <i>ddname</i>
specifies (1) the ddname of the system output device (SYSPRINT); (2) the ddnames of the DD statements defining the devices containing the direct access or tape volumes on which copies are to be made; (3) the ddname of the DD statement that identifies the volume serial number of the output volume.
Note 1: If TODD=SYSPRINT is coded, the direct access volume described by FROMDD is dumped to the system output device. If a permanent data check or missing address marker is encountered while reading the direct access volume, the defective records are identified and printed. The output may exceed the expected data size due to a data check in the count field of the error record.
Note 2: If multiple output volumes are specified in a FORMAT, ANALYZE, or RESTORE statement and an abnormal completion of the format or restore operation occurs, the operation is terminated on all output volumes.</td>
</tr>
</tbody>
</table>

Operands	Applicable Control Statements	Description of Operands/Parameters
TRACK	GETALT	TRACK=*cccchhhh* specifies in hexadecimal the cylinder number, *cccc*, and head number, *hhhh*, on a track for which an alternate track is requested. TRACK cannot specify track 0 or the first track occupied by the VTOC.
VTOC	FORMAT ANALYZE	VTOC=*xxxxxx* specifies a one- to five-byte decimal relative track address representing a primary track on which the volume table of contents is to begin. The VTOC cannot occupy track 0. To improve performance when reading from and writing to the VTOC, it is recommended that every VTOC end on the last track of a cylinder (a cylinder boundary). This means that you should determine the starting address for the VTOC by subtracting the number of tracks allocated to the VTOC from the nearest larger track that ends on a cylinder boundary. For example, if the VTOC requires 5 tracks on a 3336 disk pack, which has 19 tracks per cylinder, the starting track should be specified as track 14, so that the VTOC will end on track 18 (the last track of the first cylinder).

Restrictions

- If an error is detected in the VTOC, IEHDASDR may terminate this control function and continue with the next control card.
- If IEHDASDR is used to change a volume serial number and a subsequent operation is performed on the newly labeled volume in the same job step, two "anyname" DD statements are required. The VOLUME parameter in the first statement includes the old volume serial number; the VOLUME parameter in the second statement specifies the new volume serial number. In addition, the second statement specifies unit affinity with the first.
- One "anyname" DD statement is required for each device to be used in the job step unless the device is to be processed offline.
- The "tapename" DD statement must be included if a data set is dumped to tape or if a previously dumped data set is to be restored to a direct access volume.
- A tape created with the IEHDASDR DUMP function cannot be copied or transmitted by other programs. Such attempts will yield unpredictable results due to the physical layout of the tape. IEHDASDR allows copies to be produced as required.
- If BLKSIZE is specified on the SYSIN DD statement, it must be a multiple of 80. If BLKSIZE is omitted from the statement, a block size of 80 bytes is assumed.
- If BLKSIZE is specified on the SYSPRINT DD statement, it must be a multiple of 121. If BLKSIZE is omitted or incorrectly specified, a block size of 121 bytes is assumed.
- SYSIN attributes must be identical if SYSIN data sets are to be concatenated.
- If the PUTIPL function of IEHDASDR is used to write IPL records, the user must supply both the IPL bootstrap records and the IPL (TXT) program. The contents of the bootstrap records and the IPL (TXT) program is not checked by IEHDASDR. The IPL TXT in SYS1.SAMPLIB can not be used, unless the user also supplies the bootstrap records, with the PUTIPL function.
- The format 4 DSCB must be placed as record one (R1) on a track to conform to IBM standards.
- IEHDASDR does not support volumes with indexed VTOC, the IBM 3375, or the IBM 3380. Refer to *Device Support Facilities* for information on initialization and maintenance of such DASD volumes. Also, refer to *Data Facility/Data Set Services: User's Guide and Reference* for information on dumping or restoring such DASD volumes.

IEHDASDR Examples

The following examples illustrate some of the uses of IEHDASDR. Figure 15-9 can be used as a quick reference guide to IEHDASDR examples. The numbers in the "Example" column point to examples that follow.

Operation	Device	Comments	Example
INITIALIZE	Disk	QUICK DASDI to build a VTOC and change the volume serial number.	1
INITIALIZE	Disk	FORMAT will verify HA and write a standard R0 on each track. IPL text is included in the input stream. Volume serial id is changed.	2
INITIALIZE	Disk	Three previously initialized volumes are to be reinitialized; their volids are to be changed.	3
INITIALIZE	3350	Change volume format to match hardware mode (3330, 3330-1 or 3350).	4
INITIALIZE	MSS Staging Volume	A staging volume for MSS is initialized.	5
WRITE PROGRAM	Disk	Write IPL bookstrap records and a program on track 0 of a direct access volume.	6
GETALT and LABEL	Disk	Get alternate tracks for a previously initialized volume and change its volume serial number.	7
DUMP	Disk	Dump a copy of one volume to three other volumes.	8
DUMP	Disk and system output device	Dump a group of tracks to the system output device, which is assumed to be a printer.	9
DUMP	Disk and Tape	Dump a disk volume to magnetic tape. Only one tape volume is required.	10
RESTORE	Disk and 7-track Tape	A 3330 disk volume, previously dumped to tape, is to be restored to direct access.	11
DUMP and RESTORE	Disks and Tape	Dump operations are to be performed concurrently to minimize input/output time. Restore operations are to be performed concurrently to minimize input/output time.	12
RESTORE	Disk and Tape	A 2314 volume, previously dumped to two tape volumes, is to be restored to disk.	13
DUMP and RESTORE	Disk and Tape	VSAM and non-VSAM password-protected data sets are dumped and then restored. The receiving volume does not contain a VSAM user catalog.	14
DUMP and RESTORE	Disk and Tape	VSAM and non-VSAM password-protected data sets are dumped and then restored. The receiving volume contains a VSAM user catalog.	15

Figure 15-9. IEHDASDR Example Directory

Note: Examples which use *disk* or *tape*, in place of actual device-ids, must be changed before use. See the Device Support section, in the Introduction to this manual, for valid device-id notation.

IEHDASDR Example 1

In this example, a disk volume is initialized with a VTOC and volume serial number—a "QUICK DASDI" is performed.

```
//DASDR13   JOB
//          EXEC PGM=IEHDASDR
//SYSPRINT  DD SYSOUT=A
//DISK      DD UNIT=disk,DISP=OLD,VOL=(PRIVATE,,SER=(111111))
//SYSIN     DD *
 ANALYZE   TODD=DISK,VTOC=00019,EXTENT=00019,NEWVOLID=333333
/*
```

The control statements are discussed below:

- DISK DD defines a buffered-log DASD, volume (111111).
- ANALYZE defines the starting location and extent of the volume table of contents.

IEHDASDR Example 2

In this example, a disk volume is formatted and assigned a new serial number.

```
                                                                  72
//DASDR11    JOB
//           EXEC PGM=IEHDASDR
//SYSPRINT   DD   SYSOUT=A
//DISK       DD   UNIT=disk,DISP=OLD,VOL=(PRIVATE,
// SER=(111111))
//SYSIN      DD   *
    FORMAT        TODD=disk,VTOC=00006,EXTENT=00005,                   C
                  NEWVOLID=333333,PURGE=YES,IPLDD=SYSIN
    IPLTXT
IPL TXT (text) statements
   .
   .
   .
/*
```

The control statements are discussed below:

- DISK DD defines the disk device on which the volume (111111) is mounted.
- SYSIN DD defines the control data set which follows in the input stream.
- FORMAT defines a starting location and extent of a volume table of contents, specifies a new serial number, and indicates that the IPL text is included in the input stream. Record 0 (R0) of each track is rewritten and the rest of the track is erased. Assigns alternate tracks for flagged (defective) tracks.
- IPLTXT signals the start of IPL text.

IEHDASDR Example 3

In this example, three previously initialized disk volumes are to be initialized and assigned new serial numbers.

```
                                                                       72
//DASDR2   JOB
//         EXEC PGM=IEHDASDR
//SYSPRINT DD   SYSOUT=A
//VOL1     DD   UNIT=(disk,,DEFER),DISP=OLD,
// VOLUME=(PRIVATE,,SER=(111111))
//VOL2     DD   UNIT=(disk,,DEFER),DISP=OLD,
// VOLUME=(PRIVATE,,SER=(222222))
//VOL3     DD   UNIT=(disk,,DEFER),DISP=OLD,
// VOLUME=(PRIVATE,,SER=(333333))
//SYSIN    DD   *
  ANALYZE   TODD=VOL1,VTOC=00003,EXTENT=00010,                         C
               OWNERID=SMITH,NEWVOLID=DISK01,FLAGTEST=NO
  ANALYZE   TODD=VOL2,VTOC=00006,EXTENT=00010,                         C
               OWNERID=SMITH,NEWVOLID=DISK02,FLAGTEST=NO
  ANALYZE   TODD=VOL3,VTOC=00004,EXTENT=00010,                         C
               OWNERID=SMITH,NEWVOLID=DISK03,FLAGTEST=NO
/*
```

The control statements are discussed below:

- VOL1, VOL2, and VOL3 DD define three disk devices on which the volumes to be initialized are mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- The ANALYZE statements indicate the ddnames of DD statements defining devices on which the three disk volumes (111111, 222222, and 333333) are to be mounted. The ANALYZE statements also define starting locations and extents of the three VTOCs, specify new owner names and serial numbers (DISK01, DISK02, and DISK03), and indicate that no flag testing is to be performed on these volumes.

IEHDASDR Example 4

In this example an OFFLINE 3350 volume (in 3350 or 3330 format) will be reformatted to 3330-1 format. HA and R0 fields will be rewritten. Each flagged (defective) track encountered will be tested and recovered (unflagged) if no errors are found.

```
//DASDR4     JOB
//S1         EXEC PGM=IEHDASDR
//SYSPRINT   DD   SYSOUT=A
//SYSIN      DD    *
             ANALYZE TODD=130,VTOC=7675,EXTEND=19,                     C
                  NEWVOLID=222222,IPLDD=SYSIN,                         C
                  PASSES=1
                  IPLTXT
.
.  (IPL TEXT STATEMENTS)
.
/*
```

The control statements are discussed below:

- ANALYZE specifies that an OFFLINE 3350 is to be reformatted and initialized.
- VTOC specifies a one cylinder VTOC in the center of the 3330-1 volume.
- PASSES=1 causes HA and R0 to be written on each track, to conform with the device type defined for address 130.

IEHDASDR Example 5

In this example a staging volume for MSS is initialized.

```
//DASDR16   JOB
//          EXEC  PGM=IEHDASDR
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
            ANALYZE TODD=350,NEWVOLID=SSID00,MSS
/*
```

The control statements are discussed below:

- ANALYZE defines the staging device which is to be initialized.
- NEWVOLID specifies the new volume identification for the pack and MSS specifies that it is to be formatted as an MSS staging volume.

IEHDASDR Example 6

In this example, IPL bootstrap records and a program are to be written on track 0 of a direct access volume.

```
//DASDR10   JOB
//          EXEC PGM=IEHDASDR
//SYSPRINT  DD   SYSOUT=A
//INPUT     DD   DSNAME=IPLPROG,UNIT=disk,
// VOL=SER=222222,DISP=OLD
//OUTPUT    DD   UNIT=disk,VOL=SER=111111,DISP=OLD
//SYSIN     DD   *
        PUTIPL   FROMDD=INPUT,TODD=OUTPUT,PURGE=YES
/*
```

The control statements are discussed below:

- INPUT DD defines the input data set, which contains the IPL records and program to be written. The input data set resides on a disk volume (222222).
- OUTPUT DD defines the output data set, which is to reside on a disk volume (111111).
- SYSIN DD defines the control data set, which follows in the input stream.
- PUTIPL identifies the DD statements (INPUT and OUTPUT) that define the input and output data sets and specifies that the program to be written on the disk A volume can be written over any data after the volume label record.

IEHDASDR Example 7

In this example, alternate tracks are to be assigned for three suspected defective tracks on a 3330 volume.

```
//DASDR3    JOB
//          EXEC PGM=IEHDASDR
//SYSPRINT DD   SYSOUT=A
//VOLUME1  DD   UNIT=(3330,,DEFER),DISP=OLD,
// VOLUME=(PRIVATE,,SER=(333000))
//SYSIN    DD   *
       GETALT   TODD=VOLUME1,TRACK=00050011
       GETALT   TODD=VOLUME1,TRACK=00A00007
       GETALT   TODD=VOLUME1,TRACK=01010002
        LABEL   TODD=VOLUME1,NEWVOLID=DISK00,OWNERID=SMITH
/*
```

The control statements are discussed below:

- VOLUME1 DD defines a device that is to contain the 3330 volume (333000).
- SYSIN DD defines the control data set, which follows in the input stream.
- The GETALT statements specify the ddname of the DD statement defining the device on which the 3330 volume is mounted. The GETALT statements specify the relative track addresses of the tracks for which alternates are to be assigned.
- LABEL specifies the ddname of the DD statement defining the device on which the 3330 volume is mounted. The LABEL statement changes the serial number of the 3330 volume from 333000 to DISK00.

IEHDASDR Example 8

In this example, a copy of an entire volume (111111) is to be dumped to three volumes (222222, 333333, and 444444).

The example follows:

```
                                                                       72
//DASDR4    JOB
//          EXEC PGM=IEHDASDR
//SYSPRINT DD   SYSOUT=A
//DUMPFROM DD   UNIT=(disk,,DEFER),DISP=OLD,
// VOLUME=(PRIVATE,,SER=(111111))
//DUMPTO1  DD   UNIT=(disk,,DEFER),DISP=OLD,
// VOLUME=(PRIVATE,,SER=(222222))
//DUMPTO2  DD   UNIT=(disk,,DEFER),DISP=OLD,
// VOLUME=(PRIVATE,,SER=(333333))
//DUMPTO3  DD   UNIT=(disk,,DEFER),DISP=OLD,
// VOLUME=(PRIVATE,,SER=(444444))
//SYSIN    DD   *
 DUMP   FROMDD=DUMPFROM,TODD=(DUMPTO1,DUMPTO2,DUMPTO3),                 C
               PURGE=YES
/*
```

The control statements are discussed below:

- DUMPFROM DD defines a mountable device that is to contain a source volume.
- DUMPTO1, DUMPTO2, and DUMPTO3 DD define mountable devices that are to contain the three receiving volumes.
- DUMP specifies the dump operation and identifies the DD statements defining the applicable devices. All receiving volumes are to retain their own serial numbers.

IEHDASDR Example 9

In this example, a copy of tracks 0 through 62 (of a 3330 DASD) is to be dumped to a system output device.

The example follows:

```
//DASDR5    JOB
//          EXEC PGM=IEHDASDR
//SYSPRINT DD    SYSOUT=A
//DEV       DD    UNIT=disk,DISP=OLD,
// VOLUME=(PRIVATE,,SER=(111111))
//SYSIN     DD    *
 DUMP    FROMDD=DEV,TODD=SYSPRINT,BEGIN=00000000,END=00030004
/*
```

The control statements are discussed below:

- DEV DD defines a device that is to contain the source volume.
- DUMP specifies the dump operation, identifies the DD statements defining the source and receiving devices, and identifies the tracks that are to printed.

IEHDASDR Example 10

In this example, a disk volume (111111) is to be dumped to a tape volume (222222).

```
//DASDR6    JOB
//          EXEC PGM=IEHDASDR
//SYSPRINT DD    SYSOUT=A
//SOURCE    DD    UNIT=(disk,,DEFER),DISP=OLD,
// VOLUME=(PRIVATE,,SER=(111111))
//RECEIVE   DD    UNIT=(tape,,DEFER),DISP=NEW,DSNAME=TAPE1,
// VOLUME=(PRIVATE,,SER=(222222))
//SYSIN     DD    *
          DUMP    FROMDD=SOURCE,TODD=RECEIVE
/*
```

Note: This example assumes that only one tape volume is required. If more than one is required, code the volume serial numbers of the additional volumes in the VOLUME parameter of the DD statement that defines the magnetic tape device. For unlabeled tapes, include a volume count in the DD statement.

The control statements are discussed below:

- SOURCE DD defines a mountable device that is to contain the source volume.
- RECEIVE DD defines a tape drive that is to contain the receiving tape volume.
- DUMP specifies the dump operation and identifies the DD statements defining the source and receiving devices.

IEHDASDR Example 11

In this example, three disk volumes (222222, 333333, 444444) are to be restored from a 7-track, 556 bits per inch, standard-labeled tape volume.

```
//DASDR7    JOB
//          EXEC PGM=IEHDASDR
//SYSPRINT DD   SYSOUT=A
//TAPE      DD   UNIT=(2400-2,,DEFER),DISP=OLD,
// DCB=(TRTCH=C,DEN=1),DSNAME=TAPE1,
// VOLUME=(PRIVATE,,SER=(111111))
//DIRACC1   DD   UNIT=(disk,,DEFER),DISP=OLD,
// VOLUME=(PRIVATE,,SER=(222222))
//DIRACC2   DD   UNIT=(disk,,DEFER),DISP=OLD,
// VOLUME=(PRIVATE,,SER=(333333))
//DIRACC3   DD   UNIT=(disk,,DEFER),DISP=OLD,
// VOLUME=(PRIVATE,,SER=(444444)
//SYSIN     DD   *
     RESTORE    TODD=(DIRACC1,DIRACC2,DIRACC3),FROMDD=TAPE
/*
```

The control statements are discussed below:

- TAPE DD defines a 7-track tape unit that is to contain the source tape volume.
- DIRACC1, DIRACC2, and DIRACC3 DD define mountable devices that are to contain the three receiving volumes.
- RESTORE specifies the restore operation and identifies the DD statements defining the source and receiving devices. The receiving volumes retain their own serial numbers.

IEHDASDR Example 12

In this example, two direct access volumes are to be dumped concurrently to two receiving volumes in one operation; two direct access volumes are to be restored concurrently from two standard-labeled tape volumes in another operation.

```
//DASDR8    JOB
//          EXEC PGM=IEHDASDR
//SYSPRINT DD   SYSOUT=A
//SOURCE1   DD   UNIT=(disk,,DEFER),DISP=OLD,
// VOLUME=(PRIVATE,,SER=(111111))
//SOURCE2   DD   UNIT=(disk,,DEFER),DISP=OLD,
// VOLUME=(PRIVATE,,SER=(222222))
//TO1       DD   UNIT=disk,VOLUME=SER=333333,DISP=OLD
//TO2       DD   UNIT=disk,VOLUME=SER=444444,DISP=OLD
//SOURCE3   DD   UNIT=(tape,,DEFER),DISP=OLD,LABEL=(,NL),
// VOLUME=(PRIVATE,,SER=(555555))
//SOURCE4   DD   UNIT=(tape,,DEFER),DISP=OLD,LABEL=(,NL),
// VOLUME=(PRIVATE,,SER=(666666))
//TO3       DD   UNIT=AFF=TO1,VOLUME=SER=777777,DISP=OLD
//TO4       DD   UNIT=AFF=TO2,VOLUME=SER=888888,DISP=OLD
//SYSIN     DD   *
        DUMP    FROMDD=SOURCE1,TODD=TO1
        DUMP    FROMDD=SOURCE2,TODD=TO2
     RESTORE    TODD=TO3,FROMDD=SOURCE3
     RESTORE    TODD=TO4,FROMDD=SOURCE4
/*
```

The control statements are discussed below:

- SOURCE1 and SOURCE2 DD define devices on which the source volumes for the dump operation are to be mounted.
- TO1 and TO2 DD define devices on which the receiving volumes for the dump operation are to be mounted.

- SOURCE3 and SOURCE4 DD define devices on which the source tape volumes for the restore operation are to be mounted.
- TO3 and TO4 DD define devices on which the receiving direct access volumes for the restore operation are to be mounted. The receiving volumes for the restore operation are to be mounted on the same devices as the receiving volumes for the dump operation were mounted.

IEHDASDR Example 13

In this example, a disk volume previously dumped to tape is to be restored. Because a completely filled disk volume requires more space than is available on a single reel of tape, two tape volumes were used in the dump operation.

```
//DASDR9    JOB  00#990,SMITH
//          EXEC PGM=IEHDASDR
//SYSPRINT  DD   SYSOUT=A
//TAPE      DD   UNIT=tape,VOL=(,,,2,SER=(111111,222222)),
// DISP=OLD
//DISK      DD   UNIT=disk,VOL=SER=333333,DISP=OLD
//SYSIN     DD   *
   RESTORE  FROMDD=TAPE,TODD=DISK
/*
```

The control statements are discussed below:

- TAPE DD defines the tape volumes that contain the data to be restored to disk.
- DISK DD defines the disk volume to which data is to be restored.
- RESTORE specifies that data is to be restored from the tape volumes defined in the TAPE DD statement to the disk volume defined in the DISK DD statement.

Note: For unlabeled tapes, use the external volume identification and the LABEL=(,NL) parameter on the associated tape DD card. Also, be sure the serial numbers are entered in the same order as during the previous disk-to-tape dump.

IEHDASDR Example 14

In this example a disk volume containing VSAM and non-VSAM password-protected data sets is dumped to a standard labeled tape and then restored. The receiving volume does not contain a VSAM user catalog.

```
//DASDR14   JOB
//D14STEP1  EXEC   PGM=IEHDASDR
//STEPCAT   DD     DSNAME=VSAMCAT1,DISP=OLD
//SYSPRINT  DD     SYSOUT=A
//DISK1     DD     UNIT=disk,VOL=SER=111111,DISP=OLD
//TAPE1     DD     UNIT=tape,DISP=NEW,DSNAME=TAPE1,
// VOL=(,,,2,SER=(222222,333333)),
// LABEL=(,SL,PASSWORD),DCB=DEN=4
//DISKA     DD     UNIT=disk,VOL=SER=111111,DISP=OLD,
// DSNAME=DATASET1
//SYSIN     DD     *
     DUMP  FROMDD=DISK1,TODD=TAPE1
/*
//D14STEP2  EXEC   PGM=IEHDASDR
//STEPCAT   DD     DSNAME=VSAMCAT1,DISP=OLD
//SYSPRINT  DD     SYSOUT=A
//DISK2     DD     UNIT=disk,VOL=SER=111111,DISP=OLD
//TAPE2     DD     UNIT=tape,DISP=OLD,DSNAME=TAPE1,
// VOL=(,,,2,SER=(222222,333333)),LABEL=(,SL)
//SYSIN     DD     *
     RESTORE  TODD=DISK2,FROMDD=TAPE2,PURGE=YES
/*
```

The control statements are discussed below:

- The STEPCAT DD statements define a VSAM user catalog in which VSAM data sets on the volume are cataloged. The data sets are not cataloged in the master catalog. This must be provided even when no data sets are VSAM password protected.
- TAPE1 defines a tape unit upon which a three-volume data set resides. This data set must be password protected when any non-VSAM data sets on the volume are password protected. If no non-VSAM password data sets reside on the volume, the tape need not be password protected.
- DISK1 defines the device that is to contain the source volume.
- DISKA defines a non-VSAM password-protected data set which resides on the source volume. This is necessary since password prompting is by DDname and would cause confusion if more than one non-VSAM password-protected data set resided on the volume.
- DISK2 defines the device that is to contain the receiving volume.
- TAPE2 defines the tape unit that is to contain the source tape volumes.
- D14STEP1, during this job step the operator will be required to provide the password for each non-VSAM password-protected data set identified by the unique DDname provided in the JCL. The operator will also be required to provide the catalog master password or the data set master password for each VSAM password-protected data set.
- D14STEP2, during this job step the operator will be required to provide the password for each non-VSAM password-protected data set residing on the receiving volume. A DD statement need not be provided for those non-VSAM data sets. The operator will also be required to provide the catalog master password or the data set master password for each VSAM password-protected data set.

  If the tape data set is password protected, its password must also be supplied.

  PURGE=YES is specified since the receiving volume contains VSAM data spaces.

IEHDASDR Example 15

In this example a disk volume containing VSAM and non-VSAM password-protected data sets is dumped to a standard labeled tape and then restored.

This example is intended to illustrate the specific situation where:

STEP1 The volume dumped to tape contains VSAM and non-VSAM password-protected data sets.

and

STEP2 The receiving volume of the restore operation contains a VSAM user catalog which describes the VSAM data sets to be overlayed.

```
//DASDR15   JOB
//D15STEP1  EXEC    PGM=IEHDASDR
//STEPCAT   DD      DSNAME=VSAMCAT1,DISP=OLD
//SYSPRINT  DD      SYSOUT=A
//DISK1     DD      UNIT=disk,VOL=SER=333333,DISP=OLD
//TAPE1     DD      UNIT=tape,DISP=NEW,DSNAME=TAPE1,
// VOL=(,,,2,SER=(111111,222222)),
// LABEL=(,SL,PASSWORD),DCB=DEN=4
//DISKA     DD      UNIT=disk,VOL=SER=333333,DISP=OLD,
// DSNAME=DATASET1
//SYSIN     DD      *
     DUMP   FROMDD=DISK1,TODD=TAPE1
/*
//D15STEP2  EXEC    PGM=IEHDASDR
//STEPCAT   DD      DSNAME=VSAMCAT1,DISP=OLD
//SYSPRINT  DD      SYSOUT=A
//TAPE2     DD      UNIT=TAPE,DISP=OLD,DSNAME=TAPE1,
// VOL=(,,,2,SER=(111111,222222)),LABEL=(,SL)
//SYSIN     DD
     RESTORE        TODD=STEPCAT,FROMDD=TAPE2,PURGE=YES
/*
```

The control statements are discussed below:

- The STEPCAT DD statements define a VSAM user catalog in which VSAM data sets on the volume are cataloged. The data sets are not cataloged in the master catalog. This must be provided even when no data sets are VSAM password protected. DISP=OLD must be specified to ensure the integrity of the dumped data sets.
- TAPE1 defines a tape unit upon which a three-volume data set resides. This data set must be password protected when any non-VSAM data sets on the volume are password protected. If no non-VSAM password data sets reside on the volume, the tape need not be password protected.
- DISK1 defines the device that is to contain the source volume.
- DISKA defines a non-VSAM password-protected data set which resides on the source volume. This is necessary since password prompting is by DDname and would cause confusion if more than one non-VSAM password-protected data set resided on the volume.
- TAPE2 defines the tape unit that is to contain the source tape volumes.
- D15STEP1, during this job step the operator will be required to provide the password for each non-VSAM password-protected data set identified by the unique DDname provided in the JCL. The operator will also be required to provide the catalog master password or the data set master password for each VSAM password-protected data set.

- D15STEP2, during this job step the operator will be required to provide the password for each non-VSAM password-protected data set residing on the receiving volume. A DD statement need not be provided for those non-VSAM data sets.

  The operator will also be required to provide the catalog master password or the data set master password for each VSAM password-protected data set.

- The STEPCAT DD statement is required to allow the VSAM user catalog on the receiving volume to be opened so that a check for password protection can be made. Since IEHDASDR does not allow two DD statements to reference the same volume for one control operation, the STEPCAT DD statement is also used to describe the receiving volume. DISP=OLD must be specified to ensure that the integrity of the volume is maintained during the restore operation.

  If the tape data set is password-protected, its password must also be supplied.

IEHINITT PROGRAM

IEHINITT is a system utility used to place IBM volume label sets written in EBCDIC, in BCD, or in ASCII (American Standard Code for Information Interchange) onto any number of magnetic tapes mounted on one or more tape units. Because IEHINITT can overwrite previously labeled tapes regardless of expiration date and security protection, it is suggested that IEHINITT be moved and deleted from SYS1.LINKLIB into an authorized password protected private library. Each volume label set created by the program contains:

- A standard volume label with user-specified serial number and owner identification.
- An 80-byte dummy header label. For IBM standard labels, this record consists of HDR1 followed by zeros. For labels written in ASCII, this record consists of HDR1 followed by zeros in the remaining positions, with the exception of position 54, which contains an ASCII space.
- A tapemark.

Note: When a labeled tape is subsequently used as a receiving volume: (1) the tape mark created by IEHINITT is overwritten, (2) the dummy HDR1 record created by IEHINITT is filled in with operating system data and device-dependent information, (3) a HDR2 record, containing data set characteristics, is created, (4) user header labels are written if exits to user label routines are provided, (5) a tapemark is written, and (6) data is placed on the receiving volume.

Figure 16-1 shows an IBM standard label group after a volume is used to receive data. Refer to *OS/VS2 MVS Data Management Services Guide* for a discussion of volume labels.

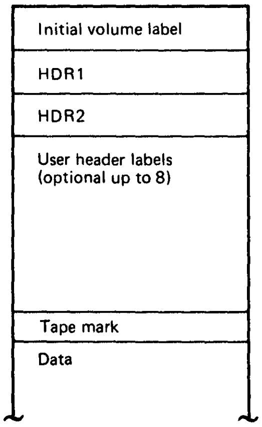


Figure 16-1.IBM Standard Label Group After Volume Receives Data

Placing a Standard Label Set on Magnetic Tape

IEHINITT can be used to write BCD labels on 7-track tape volumes and EBCDIC or ASCII labels on 9-track tape volumes. Any number of 7-track and/or 9-track tape volumes can be labeled in a single execution of IEHINITT.

Tape volumes are labeled in sequential order by specifying a serial number to be written on the first tape volume. The serial number is incremented by 1 for each successive tape volume. If only one tape volume is to be labeled, the specified serial number can be either numeric or alphameric. If more than one volume is to be labeled, the serial numbers must be specified as six numeric characters.

- If any errors are encountered while attempting to label a tape, the tape is left unlabeled. IEHINITT attempts to label any tapes remaining to be processed.
- The user can provide additional information, such as owner name, rewind or unload specifications, and whether the label is to be written in ASCII.
- The user must supply all tapes to be labeled, and must include with each job request explicit instructions to the operator about where each tape is to be mounted.
- IEHINITT writes 7-track tape labels in even parity (translator on, converter off).
- Previously labeled tapes can be overwritten with new labels regardless of expiration date and security protection.

For information on creating routines to write standard or non-standard labels, refer to *OS/VS Tape Labels.*

Input and Output

IEHINITT uses as input a control data set that contains the utility control statements.

IEHINITT produces an output data set that contains: (1) utility program identification, (2) initial volume label information for each successfully labeled tape volume, (3) contents of utility control statements, and (4) any error messages.

IEHINITT produces a return code to indicate the results of program execution. The return codes and their meanings are:

- 00, which indicates successful completion. A message data set was created.
- 04, which indicates successful completion. No message data set was defined by the user.
- 08, which indicates that the program completed its operation, but error conditions were encountered during processing. A message data set was created.
- 12, which indicates that the program completed its operation, but error conditions were encountered during processing. No message data set was defined by the user.
- 16, which indicates that the program terminated operation because of error conditions encountered while attempting to read the control data set. A message data set was created if defined by the user.

Control

IEHINITT is controlled by job control statements and utility control statements. The job control statements are used to execute or invoke IEHINITT and to define data sets used and produced by IEHINITT. Utility control statements are used to specify applicable label information.

Job Control Statements

Figure 16-2 shows the job control statements necessary for using IEHINITT.

Statement	**Use**
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEHINITT) or, if the job control statements reside in a procedure library, the procedure name. The EXEC statement can include additional PARM information; see "PARM Information on the EXEC Statement."
SYSPRINT DD	Defines a sequential output data set.
anyname DD	Defines a tape unit to be used in a labeling operation; more than one tape unit can be identified.
SYSIN DD	Defines the control data set. The control data set normally resides in the input stream; however, it can be defined as a member of a partitioned data set or as a sequential data set outside the input stream.

Figure 16-2.IEHINITT Job Control Statements

The "anyname" DD statement is entered:

```
//anyname DD DCB=DEN=x,UNIT=(xxxx,n,DEFER)
```

The DEN parameter specifies the density at which the labels are written. The UNIT parameter specifies the device type, number of units to be used for the labeling operation, and deferred mounting. The name "anyname" must be identical to a name specified in a utility control statement to relate the specified unit(s) to the appropriate utility control statement.

PARM Information on the EXEC Statement

The EXEC statement can include PARM information that specifies the number of lines to be printed between headings in the message data set, as follows:

```
PARM='LINECNT=nn'
```

If PARM is omitted, 60 lines are printed between headings.

If IEHINITT is invoked, the line count option can be passed in a parameter list that is referred to by the "optionaddr" subparameter of the LINK or ATTACH macro instruction. In addition, a page count can be passed in a six-byte parameter list that is referred to by the "hdingaddr" subparameter of the LINK or ATTACH macro instruction. For a discussion of linkage conventions, refer to "Appendix B: Invoking Utility Programs from a Problem Program."

Utility Control Statement

IEHINITT uses a utility control statement to provide control information for a labeling operation.

INITT Statement

The INITT statement provides control information for the IEHINITT program.

Any number of INITT utility control statements can be included for a given execution of the program. An identically named DD statement must exist for a utility control statement in the job step.

Figure 16-3 shows a printout of a message data set including the INITT statement and initial volume label information. In this example, one INITT statement was used to place serial numbers 001122 and 001123 on two tape volumes. VOL10011220 and VOL10011230 are interpreted, as follows:

- VOL1 indicates that an initial volume label was successfully written to a tape volume.
- 001122 and 001123 are the serial numbers that were written onto the volumes.
- 0 is the Volume Security field.

No errors occurred during processing.

```
                                                       72
SYSTEM SUPPORT UTILITIES   IEHINITT                     C

ALL  INITT  SER=001122,NUMBTAPE=2,OWNER='P.T.BROWN',
            DISP=REWIND

VOL10011220                P.T. BROWN
VOL10011230                P.T.BROWN
```

Figure 16-3.Printout of INITT Statement Specifications and Initial Volume Label Information

The format of the INITT statement is:

ddname **INITT** **SER=***xxxxxx*
[,OWNER='*cccccccccc* [*cccc*]']
[,NUMBTAPE={*n* | 1}**]**
,DISP= {**REWIND** | **UNLOAD**}
[,LABTYPE=AL]

Operands	Applicable Control Statements	Description of Operands/Parameters
DISP	INITT	DISP={REWIND | UNLOAD} specifies whether a tape is to be rewound or unloaded. These values can be coded: REWIND specifies that a tape is to be rewound (but not unloaded) after the label has been written. If DISP=REWIND is not specified, the tape volume is rewound and unloaded. UNLOAD specifies that a tape is to be unloaded after the label has been written.
LABTYPE	INITT	LABTYPE=AL specifies that a volume label written in ASCII is to be created. **Default**: The tape is written in EBCDIC for 9-track tape volumes and in BCD for 7-track tape volumes.
ddname	INITT	*ddname* specifies a name that is identical to a ddname in the name field of a DD statement defining a tape unit(s). This name must begin in column 1.
NUMBTAPE	INITT	NUMBTAPE={*n* | 1} specifies the number of tapes to be labeled according to the specifications made in this control statement. The value *n* represents a number from 1 to 255. If more than one tape is specified, the serial number must be numeric.
OWNER	INITT	OWNER=*'cccccccccc*[*cccc*]' specifies the owner's name or similar identification. The information is specified as character constants, and can be up to 10 bytes in length for EBCDIC and BCD volume labels, or up to 14 bytes in length for volume labels written in ASCII. The delimiting apostrophes can be omitted if no blanks, commas, apostrophes, equal signs, or other special characters (except periods or hyphens) are included. If an apostrophe is included within the OWNER name field, it must be written as two consecutive apostrophes.
SER	INITT	SER=*xxxxxx* specifies the volume serial number of the first or only tape to be labeled. The serial number cannot contain blanks, commas, apostrophes, equal signs, or special characters other than periods or hyphens. A specified serial number is incremented by one for each additional tape to be labeled. (Serial number 999999 is incremented to 000000.) When processing multiple tapes, the volume serial number must be all numeric.

Restrictions

- The SYSPRINT data set must have a logical record length of 121 bytes. It must consist of fixed length records with an ASA control character in the first byte of each record. Any blocking factor can be specified.
- The SYSIN data set must have a logical record length of 80. Any blocking factor can be specified.
- Labels written in ASCII cannot be put on 7-track tape volumes.

IEHINITT Examples

The following examples illustrate some of the uses of IEHINITT. Figure 16-4 can be used as a quick reference guide to IEHINITT examples. The numbers in the "Example" column point to examples that follow.

Operation	Comments	Example
LABEL	Three 9-track tapes are to be labeled.	1
LABEL	A 9-track tape is to be labeled.	2
LABEL	Two groups of 9-track tape volumes are to be labeled.	3
LABEL	9-track tape volumes are to be labeled. Sequence numbers are to be incremented by 10.	4
LABEL	Three 9-track tape volumes are to be labeled. An alphameric label is to be placed on a 2400 volume; numeric labels are placed on the 2400-4 volumes.	5
LABEL	Two 9-track tape volumes are to be labeled. The first volume is labeled at a density of 6250 bpi; the second at a density of 1600 bpi.	6

Figure 16-4.IEHINITT Example Directory

Note: Examples which use *tape* in place of actual device-ids, must be changed before use. See the Device Support section, in the Introduction to this manual, for valid device-id notation.

IEHINITT Example 1

In this example, serial numbers 001234, 001235, and 001236 are to be placed on three tape volumes; the labels are to be written in EBCDIC at 800 bits per inch. Each volume to be labeled is mounted, when it is required, on a single 9-track tape unit.

```
//LABEL1    JOB  09#990,BROWN,MSGLEVEL=(1,1)
//          EXEC PGM=IEHINITT
//SYSPRINT DD    SYSOUT=A
//LABEL    DD    DCB=DEN=2,UNIT=(tape,1,DEFER)
//SYSIN    DD    *
LABEL   INITT    SER=001234,NUMBTAPE=3
/*
```

IEHINITT Example 2

In this example, serial number 001001 is to be placed on one ASCII tape volume; the label is to be written at 800 bits per inch. The volume to be labeled is mounted, when it is required, on a 9-track tape unit.

```
//LABEL2    JOB   09#990,BROWN,MSGLEVEL=(1,1)
//          EXEC  PGM=IEHINITT
//SYSPRINT  DD    SYSOUT=A
//ASCIILAB  DD    DCB=DEN=2,UNIT=(tape,1,DEFER)
//SYSIN     DD    *
ASCIILAB INITT    SER=001001,OWNER='SAM A. BROWN',LABTYPE=AL
/*
```

IEHINITT Example 3

In this example, two groups of serial numbers (001234, 001235, 001236, and 001334, 001335, 001336) are placed on six tape volumes. The labels are to be written in EBCDIC at 800 bits per inch. Each volume to be labeled is mounted, when it is required, on a single 9-track tape unit.

```
//LABEL3   JOB   09#990,BROWN,MSGLEVEL=(1,1)
//         EXEC  PGM=IEHINITT
//SYSPRINT DD    SYSOUT=A
//LABEL    DD    DCB=DEN=2,UNIT=(tape,1,DEFER)
//SYSIN    DD    *
LABEL   INITT    SER=001234,NUMBTAPE=3
LABEL   INITT    SER=001334,NUMBTAPE=3
/*
```

IEHINITT Example 4

In this example, serial numbers 001234, 001244, 001254, 001264, 001274, etc., are to be placed on eight tape volumes. The labels are to be written in EBCDIC at 800 bits per inch. Each volume to be labeled is mounted, when it is required, on one of four 9-track tape units.

```
//LABEL4   JOB   09#990,BROWN,MSGLEVEL=(1,1)
//         EXEC  PGM=IEHINITT
//SYSPRINT DD    SYSOUT=A
//LABEL    DD    DCB=DEN=2,UNIT=(tape,4,DEFER)
//SYSIN    DD    *
LABEL   INITT    SER=001234
LABEL   INITT    SER=001244
LABEL   INITT    SER=001254
LABEL   INITT    SER=001264
LABEL   INITT    SER=001274
LABEL   INITT    SER=001284
LABEL   INITT    SER=001294
LABEL   INITT    SER=001304
/*
```

IEHINITT Example 5

In this example, serial number TAPE1 is to be placed on a tape volume, and serial numbers 001234 and 001235 are to be placed on two tape volumes. The labels are to be written in EBCDIC at 800 and 1600 bits per inch, respectively.

```
//LABEL5    JOB  09#990,BROWN,MSGLEVEL=(1,1)
//          EXEC PGM=IEHINITT
//SYSPRINT DD    SYSOUT=A
//LABEL1    DD   DCB=DEN=2,UNIT=(tape,1,DEFER)
//LABEL2    DD   DCB=DEN=3,UNIT=(tape,1,DEFER)
//SYSIN     DD   *
LABEL1  INITT    SER=TAPE1
LABEL2  INITT    SER=001234,NUMBTAPE=2
/*
```

IEHINITT Example 6

In this example, the serial number 006250 is to be written in EBCDIC on a tape volume at a density of 6250 bpi, and the serial number 001600 is to be written in EBCDIC on a second volume at a density of 1600 bpi.

```
//LABEL6    JOB  09#990,BROWN,MSGLEVEL=(1,1)
//          EXEC PGM=IEHINITT
//SYSPRINT DD    SYSOUT=A
//DDFIRST   DD   DCB=DEN=4,UNIT=(tape,1,DEFER)
//DDSECOND DD    DCB=DEN=3,UNIT=(tape,1,DEFER)
//SYSIN     DD   *
DDFIRST  INITT   SER=006250
DDSECOND INITT   SER=001600
/*
```

IEHLIST PROGRAM

IEHLIST is a system utility used to list entries in a catalog, entries in the directory of one or more partitioned data sets, or entries in a volume table of contents. Any number of listings can be requested in a single execution of the program.

Listing Catalog Entries

IEHLIST lists all OS catalog entries that are part of the structure of a fully qualified, data set name. Figure 17-1 shows an index structure for which IEHLIST lists fully qualified names A.B.D.W, A.B.D.X, A.B.E.Y, and A.B.E.Z. Because A.C.F does not represent a cataloged data set (that is, the lowest level of qualification has been deleted), it is not a fully qualified name, and it is not listed.

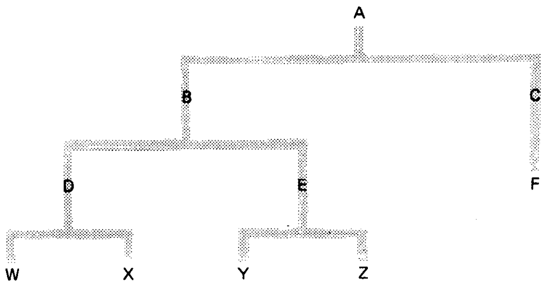


Figure 17-1.Index Structure—Listed by IEHLIST

Note: IEHLIST will list only OS catalogs (SYSCTLG data sets). To list VSAM catalogs, use Access Method Services.

Listing a Partitioned Data Set Directory

IEHLIST can list up to ten partitioned data set directories in a single application of the program. A partitioned directory is composed of variable length records blocked into 256-byte blocks. Each directory block can contain one or more entries which reflect member (and/or alias) names and other attributes of the partitioned members in edited and unedited format.

Figure 17-2 shows a directory block as it exists in storage.

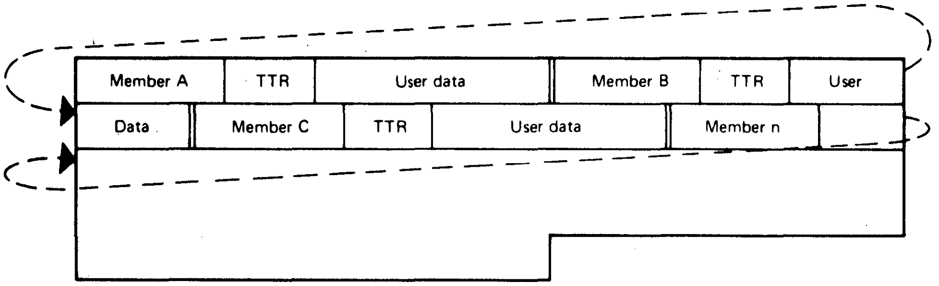


Figure 17-2.Sample Directory Block

Edited Format

IEHLIST optionally provides the following information, which is obtained from the applicable partitioned data set directory, when an edited format is requested:

- Member name
- Entry point
- Relative address of start of member
- Relative address of start of text
- Contiguous virtual storage requirements
- Length of first block of text
- Origin of first block of text
- System status indicators
- Linkage editor attributes
- APF authorization required
- Other information

Before printing the directory entries on the first page, an index is printed explaining the asterisk (*), if any, following a member name, the *attributes* (fields 3 and 10), and *other information* (field 12). Under the OTHER INFORMATION INDEX, scatter and overlay format data is described positionally as it appears in the listing; under the ATTRIBUTE INDEX, the meaning of each attribute bit is explained.

Each directory entry occupies one printed line, except when the member name is an alias and the main member name and associated entry point appear in the user data field. When this occurs, two lines are used and every alias is followed by an asterisk. If the main member is renamed, the old member name will still be in the alias directory entry and consequently printed on the second line.

Note: The FORMAT option applies only to a partitioned data set whose members have been created by the linkage editor (that is, the directory entries are at least 34 bytes long). If a directory entry is less than 34 bytes, a message is issued and the entry is printed in unedited format; if the entry is longer than 34 bytes, it is assumed that it is created by the linkage editor.

Figure 17-3 shows an edited entry for a partitioned member (IEANUC01). The entry is shown as it is listed by the IEHLIST program.

```
                                                            OTHER INFORMATION INDEX
    SCATTER FORMAT    SCTR=SCATTER/TRANSLATION TABLE TTR IN HEX, LEN OF SCRT LIST IN DEC, LEN OF TRANS TABLE IN DEC,
                            ESDID OF FIRST TEXT RCD IN DEC, ESDID OF CSECT CONTAINING ENTRY POINT IN DEC

    OVERLAY FORMAT    ONLY=NOTE LIST RCD TTR IN HEX, NUMBER OF ENTRIES IN NOTE LIST RCD IN DEC

    ALIAS NAMES       ALIAS MEMBER NAMES WILL BE FOLLOWED BY AN ASTERISK IN THE PDS FORMAT LISTING

                                                              ATTRIBUTE INDEX

    BIT    ON      OFF          BIT   ON       OFF            BIT    ON          OFF          BIT    ON          OFF

     0     RENT    NOT RENT      4    OL       NOT OL           8    NOT DC      DC           12     NOT EDIT    EDIT
     1     REUS    NOT REUS      5    SCTR     BLOCK            9    ZERO ORG    NOT ZERO     13     SYMS        NO SYMS
     2     ONLY    NOT ONLY      6    EXEC     NOT EXEC        10    EP ZERO     NOT ZERO     14     F LEVEL     E LEVEL
     3     TEST    NOT TEST      7    1 TXT    MULTI RCD       11    NO RLD      RLD          15     REFR        NOT REFER
MEMBER      ENTRY     ATTR    REL        ADDR-HEX    CONTIG        LEN 1ST    ORB 1ST    SSI       VS        AUTH OTHER
NAME        PT-HEX    HEX     BEGIN      1ST TXT     STOR-DEC      TXT-DEC    TXT-HEX    INFO      ATTR      REQ  INFORMATION

IEANUC01    000000    06E2    000004     00020F      000166248     0927                  ABSENT    880000    NO   SCTR=000000,
                                                                                                                  00484,01084,32,32

OF THE 00002 DIRECTORY BLOCKS ALLOCATED TO THIS PDS, 00001 ARE(IS) COMPLETELY UNUSED
```

Figure 17-3.Edited Partitioned Directory Entry

Unedited (Dump) Format

The user may choose the unedited format. If this is the case, IEHLIST lists each member separately.

Figure 17-4 shows how the information in Figure 17-2 is listed.

Note: A listing organized as shown in Figure 17-4 can also be obtained by using IEBPTPCH (see "IEBPTPCH Program ").

```
MEMB A      TTR          USER DATA
MEMB B      TTR          USER DATA
MEMB C      TTR          USER DATA
MEMB n      TTR          USER DATA
```

Figure 17-4.Sample Partitioned Directory Listing

To correctly interpret user data information, the user must know the format of the partitioned entry. The formats of directory entries are discussed in *OS/VS2 Data Areas.*

Listing a Volume Table of Contents

IEHLIST can be used to list, partially or completely, entries in a specified volume table of contents (VTOC). The program lists the contents of selected data set control blocks (DSCBs) in edited or unedited form.

Note:VSAM data spaces are identified only; not the data sets within them.

Edited Format

Two edited formats are available. One is a comprehensive listing of the DSCBs in the VTOC. It provides the status and attributes of the volume, and describes in depth the data sets residing on the volume. This listing includes:

- Logical record length and block size
- Initial and secondary allocations
- Upper and lower limits of extents
- Alternate track information
- Available space information, in detail
- Option codes (printed as two hexadecimal digits)
- Record formats

A VTOC consists of as many as seven types of DSCBs which contain information about the data sets residing on the volume:

- Identifier DSCB—Format 1
- Index DSCB—Format 2
- Extension DSCB—Format 3
- VTOC DSCB—Format 4
- Free Space DSCB—Format 5
- Shared Extent DSCB—Format 6
- Free VTOC DSCB—Format 0

The first DSCB in a VTOC (and on your listing) is always a VTOC (Format 4) DSCB. It defines the scope of the VTOC itself; that is, it contains information about the VTOC and the volume rather than the data sets referenced by the VTOC.

The VTOC (Format 4) DSCB is followed, when necessary, by the Free Space (Format 5) DSCB, which describes the space available on the volume for allocation to other data sets. More than one Format 5 DSCB may be required to describe the available space on a volume because each Format 5 DSCB describes only 26 extents.

The Format 4 and Format 5 DSCBs are followed, in any order, by Format 1, 2, 3, or 6 DSCBs.

Each Identifier (Format 1) DSCB contains information about a particular data set residing on the volume. This type of DSCB describes the characteristics and up to three extents of the data set.

For data sets having indexed sequential organization, additional characteristics are specified in an Index (Format 2) DSCB pointed to by the Identifier (Format 1) DSCB.

Additional extents are described in an Extension (Format 3) DSCB pointed to by the Identifier (Format 1) DSCB or in the Index (Format 2) DSCB for an indexed-sequential data set.

A Shared Extent (Format 6) DSCB is used for shared-cylinder allocation. It describes the extent of space (one or more contiguous cylinders) that is being shared by two or more data sets. The Shared Extent (Format 6) DSCB is pointed to by the VTOC (Format 4) DSCB. Subsequent Format 6 DSCBs are pointed to by the previous Format 6 DSCB.

A Free VTOC Record (Format 0) DSCB, which indicates space available for another DSCB, is not listed by IEHLIST. They are 140-byte records, consisting of binary zeros, that are overwritten with Format 1, 2, 3, and 6 DSCBs when a new data set is allocated, and with Format 5 DSCBs when space is released.

Figure 17-5 shows a sample listing of the edited format. This sample illustrates how each DSCB will appear on a listing, although in many cases the VTOC may not contain all possible types. The information is in columns, with the values or numbers appearing underneath each item's heading.

The second edited format is an abbreviated description of the data sets. It is provided by default when no format is requested specifically. It provides the following information:

- Data set name
- Creation date (dddyy)
- Expiration date (dddyy)
- Password indication
- Organization of the data set
- Extent(s)
- Volume serial number

The last line in the listing indicates how much space remains in the VTOC.

```
                             SYSTEMS SUPPORT UTILITIES---IEHLIST                               PAGE 1
               CONTENTS OF VTOC ON VOL EXAMPL
FORMAT 4 DSCB NO AVAIL/MAX DSCB /MAX DIRECT  NO AVAIL NEXT ALT    FORMAT 6    LAST FMT 1     VTOC EXTENT      THIS DSCB
          VI   DSCBS   PER TRK  BLK PER TRK  ALT TRK  TRK(C-H)    (C-H-R)     DSCB(C-H-R)/LOW(C-H) HIGH(C-H)  (C-H-R)
          00     154       16         10         30    200                      5   0   5    5   0   5   9    5   0   1

FORMAT 5 DSCB     A = NUMBER OF TRKS IN ADDITION TO FULL CYLS IN THE EXTENT
      TRK  FULL          TRK  FULL          TRK  FULL          TRK  FULL          TRK  FULL          TRK  FULL
     ADDR  CYLS   A     ADDR  CYLS   A     ADDR  CYLS   A     ADDR  CYLS   A     ADDR  CYLS   A     ADDR  CYLS   A
       17     3    3     110   189    0

                DSCB(C-H-R)   5   0   2

---------------DATA SET NAME---------------  ID  SER NO  SEQ NO  CREDT  EXPDT  REFDT  NO EXT  DSORG  RECFM  OPTCD  BLKSIZE
EXAMPLE.OF.COMBINED.FORMATS.ONE.AND.TWO       1  EXAMPL      1   36699  27469  00000     1     IS    F                100
            LRECL  KEYLEN  INITIAL ALLOC  2ND ALLOC/LAST BLK PTR(T-R-L)  USED PDS BYTES  FMT 2 OR 3(C-H-R)/DSCB(C-H-R)
              100     4     ABSTR                0                                              5   0   3     5   0   4

            EXTENTS  NO  LOW(C-H)   HIGH(C-H)
                      0     6   0      10   9

     2MIND(M-B-C-H)/3MIND(M-B-C-H)/L2MFN(C-H-R)/L3MIN(C-H-R)/CYLAD(M-B-C-H)/ADLIN(M-B-C-H)/ADHIN(M-B-C-H)/NOBYT/ NOTRK
       0 0   0   0    0 0   0   0    0   0   0    0   0   0    1 0  10   9    0 0   0   0    1 0  10   9     70     0

     LTRAD(C-H-R)/LCYAD(C-H-R)/LMSAD(C-H-R)/LPRAD(M-B-C-H-R) /NOLFV /CYLOV/ TAGDT/   PRCTR  / OVRCT/ RORG1/PTRDS(C-H-R)
       6   0   3   10   9   1    0   0   0    1 0   6   1  12    1      0             20          0      0
                                ----UNABLE TO CALCULATE EMPTY SPACE.

---------------DATA SET NAME---------------  ID  SER NO  SEQ NO  CREDT  EXPDT  REFDT  NO EXT  DSORG  RECFM  OPTCD  BLKSIZE
EXAMPLE.OF.COMBINED.FORMATS.ONE.AND.THREE     1  EXAMPL      1   36699  27069  00000    16     PS    V               3504
            LRECL  KEYLEN  INITIAL ALLOC  2ND ALLOC/LAST BLK PTR(T-R-L)   USED PDS BYTES  FMT 2 OR 3(C-H-R)/DSCB(C-H-R)
             3500           TRKS               1         15   1  1723                           5   0   6     5   0   5

            EXTENTS  NO  LOW(C-H)   HIGH(C-H)   NO  LOW(C-H)  HIGH(C-H)      NO  LOW(C-H)   HIGH(C-H)
                      0     0   1       0   1    1     0   2      0   2       2     0   3       0   3
                      3     0   4       0   4    4     0   5      0   5       5     0   6       0   6
                      6     0   7       0   7    7     0   8      0   8       8     0   9       0   9
                      9     1   0       1   0   10     1   1      1   1      11     1   2       1   2
                     12     1   3       1   3   13     1   4      1   4      14     1   5       1   5
                     15     1   6       1   6
                                ----ON THE ABOVE DATA SET,THERE ARE  0 EMPTY TRACK(S).

THERE ARE  192 EMPTY CYLINDERS PLUS    3 EMPTY TRACKS ON THIS VOLUME
THERE ARE  154 BLANK DSCBS IN THE VTOC ON THIS VOLUME
```

Figure 17-5.Sample Printout of a Volume Table of Contents

Unedited (Dump) Format

This option produces a complete hexadecimal listing of the DSCBs in the VTOC. The listing is in an unedited *dump* form, requiring the user to know the various formats of applicable DSCBs. The VTOC overlay for IEHLIST listings of VTOCs in dump format is useful in identifying the fields of the DSCBs.

Refer to *OS/VS2 Data Areas* for a discussion of the various formats that data set control blocks can assume.

Input and Output

IEHLIST uses the following input:

- One or more source data sets that contain the data to be listed. The input data set(s) can be: (1) a VTOC data set, (2) a partitioned data set, or (3) an OS catalog data set (SYSCTLG).
- A control data set, which contains utility control statements that are used to control the functions of IEHLIST.

IEHLIST produces as output a message data set which contains the result of the IEHLIST operations. The message data set includes the listed data and any error messages.

IEHLIST produces a return code to indicate the results of program execution. The return codes and their meanings are:

- 00, which indicates successful completion.

- 08, which indicates that an error condition caused a specified request to be ignored. Processing continues.
- 12, which indicates that a permanent input/output error occurred. The job is terminated.
- 16, which indicates that an unrecoverable error occurred while reading the data set. The job is terminated.

Control

IEHLIST is controlled by job control statements and utility control statements. The job control statements are used to execute or invoke IEHLIST and to define the data sets used and produced by IEHLIST.

Utility control statements are used to control the functions of the program and to define those data sets or volumes to be modified.

Job Control Statements

Figure 17-6 shows the job control statements necessary for using IEHLIST.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEHLIST) or, if the job control statements reside in a procedure library, the procedure name. Additional PARM information can be specified to control the number of lines printed per page. See "PARM Information on the EXEC Statement" below.
SYSPRINT DD	Defines a sequential message data set.
anyname1 DD	Defines a permanently mounted volume.
anyname2 DD	Defines a mountable device type.
SYSIN DD	Defines the control data set. The control data set normally follows the job control language in the input stream; however, it can be defined as an unblocked sequential data set or member of a procedure library.

Figure 17-6.IEHLIST Job Control Statements

The "anyname1" DD statement can be entered:

```
//anyname1 DD UNIT=xxxx,VOLUME=SER=xxxxxx,DISP=OLD
```

The UNIT and VOLUME parameters define the device type and volume serial number. The DISP=OLD specification prevents the inadvertent deletion of the data set. This statement is arbitrarily assigned the ddname DD1 in the IEHLIST examples.

When deferred mounting is required, the "anyname2" DD statement can be entered:

```
//anyname2 DD UNIT=(xxxx,,DEFER),VOLUME=(PRIVATE,...),DISP=OLD
```

See "Appendix C: DD Statements for Defining Mountable Devices" for information on defining mountable devices. This statement is arbitrarily assigned the ddname DD2 in the IEHLIST examples. Statements defining additional mountable devices are assigned ddnames DD3, DD4, etc.

With the exception of the SYSIN and SYSPRINT DD statements, all DD statements in this table are used as device allocation statements, rather than as true data definition statements.

PARM Information on the EXEC Statement

Additional information can be specified in the PARM parameter of the EXEC statement to control the number of lines printed per page. The PARM parameter can be coded:

PARM='LINECNT=xx'

The LINECNT parameter specifies the number of lines, xx, to be printed per page; xx is a decimal number from 01 through 99. If LINECNT is not specified, 58 lines are printed per page. The PARM field cannot contain embedded blanks, zeros, or any other PARM keywords, or the default of 58 is used.

Utility Control Statements

Statement	Use
LISTCTLG	Requests a listing of all or part of an OS catalog (SYSCTLG).
LISTPDS	Requests a directory listing of one or more partitioned data sets.
LISTVTOC	Requests a listing of all or part of a volume table of contents.

Figure 17-7.IEHLIST Utility Control Statements

LISTCTLG Statement

The LISTCTLG statement is used to request a listing of either the entire catalog or a specified portion of the catalog (SYSCTLG data set). The listing includes the fully qualified name of each applicable cataloged data set and the serial number of the volume on which it resides. *Empty index levels are not listed.*

The format of the LISTCTLG statement is:

[*label*] **LISTCTLG** [**VOL=***device* **=***serial*]

[**,NODE=***name*]

LISTPDS Statement

The LISTPDS statement is used to request a directory listing of one or more partitioned data sets that reside on the same volume.

Before printing the directory entries on the first page, an index is printed explaining the *attributes* (fields 3 and 10) and *other information* (field 12). OTHER INFORMATION INDEX explains scatter and overlay format data as it appears in the listing; ATTRIBUTE INDEX explains each attribute bit.

Note: The Format option of the LISTPDS statement may be used only on a partitioned data set whose members have been created by the linkage editor. Members that have not been created by the linkage editor cause their directory entries to be listed in unedited (DUMP) format.

The format of the LISTPDS statement is:

[*label*] **LISTPDS** **DSNAME=**{*dsname1* | (*dsname1* [,*dsname2* [],...])}

[**,VOL=***device* **=***serial*]

[{**,DUMP** | **,FORMAT**}]

LISTVTOC Statement

The LISTVTOC statement is used to request a partial or complete listing of the entries in a specified volume table of contents.

The format of the LISTVTOC statement is:

```
[label] LISTVTOC  [{DUMP | FORMAT} [,INDEXDSN=SYS1.VTOCIX.nnn]]
                  [,DATE=dddyy]
                  [,VOL=device=serial]
                  [,DSNAME=(name[,name]...)]
```

Operands	Applicable Control Statements	Description of Operands/Parameters
DATE	LISTVTOC	DATE=*dddyy* specifies that each entry that expires before this date is to be flagged with an asterisk (*) in the listing. This parameter applies only to the abbreviated edited format. The date is represented by *ddd,* the day of the year, and *yy,* the last two digits of the year. **Default:** No asterisks appear in the listing..
DSNAME		**DSNAME=(***name[,name]...* **)**
	LISTPDS	specifies the fully qualified names of the partitioned data sets whose directories are to be listed. A maximum of ten names is allowed. If the list consists of a single name, the parentheses can be omitted.
	LISTVTOC	specifies the fully qualified names of the data sets whose entries are to be listed. A maximum of ten names is allowed. If the list consists of a single name, the parentheses can be omitted.
DUMP	LISTPDS LISTVTOC	**DUMP** specifies that the listing is to be in unedited, hexadecimal form. **Default:** If both DUMP and FORMAT are omitted, an abbreviated edited format is generated for LISTVTOC. For LISTPDS, DUMP is the default used.
FORMAT		FORMAT
	LISTPDS	specifies that the listing is to be edited for each directory entry.
	LISTVTOC	specifies that a comprehensive edited listing is to be generated. **Default:** If both FORMAT and DUMP are omitted, an abbreviated edited format is generated for LISTVTOC. For LISTPDS, DUMP is the default used.
INDEXDSN	LISTVTOC	INDEXDSN=SYS1.VTOCIX.nnnn specifies that index information is to be listed, in addition to the VTOC. nnn is any third level qualifier. DUMP or FORMAT must be specified if INDEXDSN is specified. For more information, refer to *Data Facility/ Device Support: User's Guide and Reference.*
NODE	LISTCTLG	NODE=*name* specifies a qualified name. All data set entries whose names are qualified by this name are listed. The CVOL must be defined in the VSAM Master Catalog as: SYSTCTLG.VYYYYYY, where YYYYYY is the serial number of the CVOL, see *Using OS Catalog Management with the Master Catalog: CVOL Processor* and *OS/VS2 MVS CVOL Processor*. **Default:** All data set entries are listed.
VOL	LISTCTLG LISTPDS LISTVTOC	VOL=*device=serial* specifies the device type and volume serial number of the volume on which the catalog, PDS directory, or VTOC resides. **Default:** For LISTCTLG, the catalog is assumed to reside on the system residence volume.

Restrictions

- The block size for SYSIN and SYSPRINT must be a multiple of 80 and 121, respectively. Any blocking factor can be specified for these block sizes.
- Concatenated DD statements are allowed only for SYSIN.
- An "anyname1" DD statement must be included for each permanently mounted volume referred to in the job step. (The system residence volume is considered to be a permanently mounted volume.)
- An "anyname2" DD statement must be included for each mountable device to be used in the job step.
- Because IEHLIST modifies the internal control blocks created by device allocation DD statements, IEHLIST job control statements must not include the DSNAME parameter. (All data sets are defined explicitly or implicitly by utility control statements.)
- When IEHLIST is dynamically invoked in a job step by another program, the DD statements defining mountable devices for IEHLIST must precede DD statements required by the other program.
- IEHLIST cannot support empty space calculations for data sets allocated in blocks when the block sizes are approximately the same or larger than the track size. The empty block calculation gives only approximate indications of available space. When IEHLIST cannot supply an approximate number, the "Unable to Calculate" message is issued.
- IEHLIST specifications do not allow for protection of the object being listed. If another program updates a block of the data set just prior to IEHLIST reading the data set, a message (IEH105I or IEH108I) may be issued and the output produced by IEHLIST may be incorrect. If this happens, rerun the job.
- If you are using IEHLIST to list both the VTOC and the index data set of an indexed VTOC, refer to *Data Facility/Device Support: User's Guide and Reference.*

IEHLIST Examples

The following examples illustrate some of the uses of IEHLIST. Figure 17-8 can be used as a quick reference guide to IEHLIST examples. The numbers in the "Example" column point to examples that follow.

Operation	Devices	Comments	Example
LIST	Disk and system output device	Source catalog is to be listed on the system output device.	1
LIST	Disk system residence device and system output device	Three catalogs and part of a fourth are to be listed on the system output device.	2
LIST	Disk and system output device	Three partitioned directories are to be listed on the system output device.	3
LIST	Disk and system output device	Volume table of contents is to be listed in edited form; selected data set control blocks are listed in unedited form.	4

Figure 17-8. IEHLIST Example Directory

Note: Examples which use *disk*, in place of actual device-ids, must be changed

Note: Examples which use *disk*, in place of actual device-ids, must be changed before use. See the Device Support section, in the Introduction to this manual, for valid device-id notation.

Note: In the IEHLIST examples, the EXEC statement and the SYSPRINT DD statement can be replaced with the following job control statement:

```
// EXEC PROC=LIST
```

The EXEC statement invokes the following IBM-supplied cataloged procedure:

```
//LIST EXEC      PGM=IEHLIST,REGION=44K
//DDSRV DD       VOLUME=REF=SYS1.SVCLIB,DISP=OLD
//SYSPRINT DD    SYSOUT=A
```

IEHLIST Example 1

In this example, an OS catalog data set named SYSCTLG, residing on a disk volume (111111), is to be listed.

The example follows:

```
//LISTCAT   JOB  09#550,BLUE
//          EXEC PGM=IEHLIST
//SYSPRINT  DD   SYSOUT=A
//DD2       DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//SYSIN     DD   *
     LISTCTLG    VOL=disk=111111
/*
```

The control statements are discussed below:

- DD2 DD defines a mountable device on which the volume containing the source catalog is mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- LISTCTLG defines the source volume and specifies the list operation.

IEHLIST Example 2

In this example, a catalog residing on the system residence volume, two catalogs residing on disk volumes, and a portion of a catalog residing on another volume, are to be listed.

```
//LISTCATS JOB  09#550,BLUE
//          EXEC PGM=IEHLIST
//SYSPRINT  DD   SYSOUT=A
//DD1       DD   UNIT=diskB,VOLUME=SER=111111,DISP=OLD
//DD2       DD   UNIT=(diskA,,DEFER),DISP=OLD,
// VOLUME=(PRIVATE,,SER=(222222))
//SYSIN     DD   *
     LISTCTLG
     LISTCTLG    VOL=diskA=333333
     LISTCTLG    VOL=diskA=444444
     LISTCTLG    VOL=diskA=555555,NODE=A.B.C
/*
```

The control statements are discussed below:

- DD1 DD defines a system residence device. (The first catalog to be listed resides on the system residence volume.)
- DD2 DD defines a mountable device on which each *diskA* volume is mounted as it is required by the program.
- SYSIN DD defines the control data set, which follows in the input stream.

- The first LISTCTLG statement indicates that the catalog residing on the system residence volume is to be listed.
- The second and third LISTCTLG statements identify two *diskA* disk volumes containing catalogs to be listed.
- The fourth LISTCTLG statement identifies a *diskA* volume containing a catalog that is to be partially listed. All data set entries whose beginning qualifiers are "A.B.C" are listed.

IEHLIST Example 3

In this example, a partitioned directory existing on the system residence volume is to be listed. In addition, two partitioned directories existing on another disk volume are to be listed.

```
//LISTPDIR JOB  09#550,BLUE
//          EXEC PGM=IEHLIST
//SYSPRINT DD   SYSOUT=A
//DD1       DD   UNIT=diskB,VOLUME=SER=111111,DISP=OLD
//DD2       DD   UNIT=diskA,VOLUME=SER=222222,DISP=OLD
//SYSIN     DD   *
       LISTPDS   DSNAME=PARSET1
       LISTPDS   DSNAME=(PART1,PART2),VOL=diskA=222222
/*
```

The control statements are discussed below:

- DD1 DD defines the system residence device.
- DD2 DD defines a mountable device on which a disk volume (222222) is to be mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- The first LISTPDS statement indicates that the partitioned data set directory belonging to data set PARSET1 is to be listed. This data set exists on the system residence volume.
- The second LISTPDS statement indicates that partitioned directories belonging to data sets PART1 and PART2 are to be listed. These data sets exist on a disk volume (222222).

IEHLIST Example 4

In this example, a volume table of contents in edited form, is to be listed. The edited listing is supplemented by an unedited listing of selected data set control blocks.

```
//LISTVTOC JOB  09#550,BLUE
//          EXEC PGM=IEHLIST
//SYSPRINT DD   SYSOUT=A
//DD2       DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//SYSIN     DD   *
      LISTVTOC   FORMAT,VOL=disk=111111
      LISTVTOC   DUMP,VOL=disk=111111,DSNAME=(SET1,SET2,SET3)
/*
```

The control statements are discussed below:

- DD2 DD defines a mountable device on which the volume containing the specified volume table of contents is to be mounted.
- SYSIN DD defines the control data set which follows in the input stream.

- The first LISTVTOC statement indicates that the volume table of contents on the specified disk volume is to be listed in edited form.
- The second LISTVTOC statement indicates that the data set control blocks representing data sets SET1, SET2, and SET3 are to be listed in unedited form.

IEHMOVE PROGRAM

IEHMOVE is a system utility used to move or copy logical collections of operating system data.

IEHMOVE can be used to move or copy:

- A data set residing on from one to five volumes, with the exception of ISAM data sets, and VSAM data spaces.
- A group of cataloged data sets.
- An OS catalog (CVOL) or portions of a CVOL.
- A volume of data sets.

The scope of a basic move or copy operation can be enlarged by:

- Including or excluding data sets from a move or copy operation.
- Merging members from two or more partitioned data sets.
- Including or excluding selected members.
- Renaming moved or copied members.
- Replacing selected members.

If, for some reason, IEHMOVE is unable to successfully move or copy specified data, an attempt is made to reorganize the data and place it on the specified output device. The reorganized data—called an *unloaded data set*—is a sequential data set consisting of 80-byte blocked records that contain the source data and control information for subsequently reconstructing the source data as it originally existed.

When an unloaded data set is moved or copied to a device that will support the data in its true form, the data is automatically reconstructed. For example, if the user attempts to move a partitioned data set to a tape volume, the data is unloaded to that volume. The user can re-create the data set simply by moving the unloaded data set to a direct access (DASD) volume.

A move operation differs from a copy operation in that a move operation scratches source data if the data set resides on a direct access source volume and the expiration date has occurred. while a copy operation leaves source data intact. In addition, for cataloged data sets, a move operation updates the catalog to refer to the moved version (unless otherwise specified), while a copy operation leaves the catalog unchanged.

Space can be allocated for a data set on a receiving volume either by the user (through the use of DD statements in a prior job step) or by IEHMOVE in the IEHMOVE job step. If the source data is unmovable (that is, if it contains location dependent code), the user should allocate space on the receiving volume using absolute track allocation to ensure that the data set is placed in the same relative location on the receiving volume as it was on the source volume. Unmovable data can be moved or copied if space is allocated by IEHMOVE, but the data will not be in the same location on the receiving volume as it was on the source volume. When data sets are to be moved or copied between unlike DASD devices, a secondary allocation should be made to ensure that ample space is available on the receiving volume.

Space for a new data set should not be allocated by the user when a direct data set is to be moved or copied, not unloaded, because IEHMOVE cannot determine if the new data set is empty.

If IEHMOVE performs the space allocation for a new data set, the space requirement information of the old data set (if available) is used. This space requirement information is obtained from the DSCB of the source data set, if it is on a DASD volume, or the control information in the case of an unloaded data set.

If space requirement information is available, IEHMOVE uses this information to derive an allocation of space for the receiving volume, taking into account the differences in device characteristics, such as track capacity and overhead factors. However, when data sets with variable or undefined record formats are being moved or copied between unlike DASD devices, no assumption can be made about the space that each individual record needs on the receiving device.

In general, when variable or undefined record formats are to be moved or copied, IEHMOVE attempts to allocate sufficient space. This might cause too much space to be allocated under the following circumstances:

- When moving or copying from a device with a relatively large block overhead to a device with a smaller block overhead, the blocks being small in relation to the block size.
- When moving or copying from a device with a relatively small block overhead to a device with a larger block overhead, the blocks being large in relation to the block size.

Direct data sets with variable or undefined record formats always have the same amount of space allocated by IEHMOVE. This practice preserves any relative track addressing system that might exist within the data sets.

If the Resource Access Control Facility (RACF) is active, the following considerations apply:

- The user must have valid RACF authorization to access any RACF-defined data sets with IEHMOVE. ALTER authorization is required to the source data set for a MOVE function, as the source data set is scratched. When moving a volume or group of data sets, the user must have adequate access authorization to all of the RACF-protected data sets on the volume or in the group.
- If the user has the ADSP attribute and IEHMOVE is to allocate space for the receiving data set, that data set will be automatically defined to RACF. If the data set does not have the userid of this user as the first level qualifier, at least one of the following conditions must be met:
  - the user specifies MOVE or COPY with RENAME so that the first level qualifier is the correct userid
  - the data set being moved or copied is a group data set and the user is connected to the group with CREATE authority
  - the user has the OPERATION attribute
- If COPYAUTH is specified and the input data set is RACF-protected (whether or not the user has the ADSP attribute), the receiving data set of a MOVE or COPY operation is given a copy of the input data set's RACF protection and access list during allocation, governed by the same restrictions described above for defining a data set for a user with the

ADSP attribute. The user must have ALTER access authorization to the input data set to either MOVE or COPY using COPYAUTH.

A move or copy operation results in: (1) a moved or copied data set, (2) no action, or (3) an unloaded version of the source data set. These results depend upon the compatibility of the source and receiving volumes with respect to:

- Size of the volumes.
- Data set organization (sequential, partitioned, or direct).
- Movability of the source data set.
- Allocation of space on the receiving volume.

Two volumes are compatible with respect to size if (1) the source record size does not exceed the receiving track size, or (2) the receiving volume supports the track overflow feature and the output is to be written with track overflow. (Refer to "Job Control Statements" for notes on the track overflow feature.) When using direct data set organization, two volumes are compatible with respect to size if the source track capacity does not exceed the receiving track capacity. Direct data sets moved or copied to a smaller device type or tape are unloaded. If the user wishes to load an unloaded data set, it must be loaded to the same device type from which it was originally unloaded.

Figure 18-1 shows the results of move and copy operations when the receiving volume is a DASD volume that is compatible in size with the source volume. The organization of the source data set is shown along with the characteristics of the receiving volume.

Receiving Volume Characteristics	Sequential	Partitioned	Direct
Space allocated by IEHMOVE (movable data)	moved or copied	moved or copied	moved or copied
Space allocated by IEHMOVE (unmovable data)	moved or copied	moved or copied	no action
Space previously allocated, as yet unused	moved or copied	moved or copied	no action
Space previously allocated, partially used	no action	moved or copied (merged)	no action

Figure 18-1.Move and Copy Operations—DASD Receiving Volume with Size Compatible with Source Volume

Figure 18-2 shows the results of move and copy operations when the receiving volume is a DASD volume that is not compatible in size with the source volume. The organization of the source data set is shown along with the characteristics of the receiving volume.

Figure 18-3 shows the results of move and copy operations when the receiving volume is not a DASD volume. The organization of the source data set is shown along with the characteristics of the receiving volume.

Space cannot be previously allocated for a partitioned data set that is to be unloaded unless the SPACE parameter in the DD statement making the allocation implies sequential organization. Direct data sets should not be

Receiving Volume Characteristics	**Sequential**	**Partitioned**	**Direct**
Space allocated by IEHMOVE	unloaded	unloaded	unloaded
Space previously allocated, as yet unused	unloaded	unloaded	no action
Space previously allocated, partially used	no action	no action	no action

Figure 18-2.Move and Copy Operations—DASD Receiving Volume with Size Incompatible with Source Volume

Receiving Volume Characteristics	**Sequential**	**Partitioned**	**Direct**
Movable data	moved or copied	unloaded	unloaded
Unmovable data	unloaded	unloaded	no action

Figure 18-3.Move and Copy Operations—Non-DASD Receiving Volume

previously allocated because IEHMOVE cannot determine whether they are empty or not.

If a move or copy operation is unsuccessful, the source data remains intact.

If a move or copy operation is unsuccessful and space was allocated by IEHMOVE, all data associated with that operation is scratched from the receiving direct access volume. If the receiving volume was tape, it will contain a partial data set.

If a move or copy operation is unsuccessful and space was previously allocated, no data is scratched from the receiving volume. If, for example, IEHMOVE moved 104 members of a 105-member partitioned data set and encountered an input/output error while moving the 105th member:

- The entire partitioned data set is scratched from the receiving volume if space was allocated by IEHMOVE.
- No data is scratched from the receiving volume if space was previously allocated. In this case, after determining the nature of the error, the user need move only the 105th member into the receiving partitioned data set.

If a sequential data set, which is not an unloaded data set, on a non-DASD volume is to be moved or copied to a DASD volume, and space attributes are not available through a previous allocation, IEHMOVE makes a default space allocation. The default allocation consists of a primary allocation of 72,500 bytes of storage (data and gaps) and up to 15 secondary allocations of 36,250 bytes each.

When moving or copying a data set group or a volume containing password-protected data sets, the user must provide the password each time a data set is opened or scratched.

IEHMOVE always moves or copies any user labels associated with an input data set. IEHMOVE does not take exits to a user's label processing routines.

Note: If a data set that has only user trailer labels is to be moved from a tape volume to a direct access volume, space must be previously allocated on the direct access volume to ensure that a track is reserved to receive the user labels.

Reblocking

Data sets with fixed or variable records can be reblocked to a different block size by previously allocating the desired block size on the receiving volume. No reblocking can be performed when loading or unloading. Also, no reblocking can be performed on data sets with variable-spanned or variable-blocked-spanned records.

When moving or copying data sets with undefined record format and reblocking to a smaller block size (that is, transferring records to a device with a track capacity smaller than the track capacity of the original device), the user must make the block size for the receiving volume equal to or larger than the size of the largest record in the data set being moved or copied.

Moving or Copying a Data Set

IEHMOVE can be used to move or copy sequential, partitioned, and direct access data sets, as follows:

- A sequential data set can be: (1) *moved* from one DASD volume or non-DASD volume to another (or to the same volume provided that it is a direct access volume), or (2) *copied* from one volume to another (or to the same volume provided that the data set name is changed and the receiving volume is a DASD volume).
- A direct data set can be *moved* or *copied* from one DASD volume to another provided that the receiving device type is the same device type or larger, and that the record size does not exceed 32K.
- A partitioned data set can be: (1) *moved* from one DASD volume to another (or to the same volume) or, (2) *copied* from one direct access volume to another (or to the same volume provided that the data set name is changed).

When IEHMOVE uncatalogs a data set, cataloged in a VSAM catalog, which has one or more aliases, the aliases are also removed. An alias can be replaced by means of the Access Method Services program when the data set is cataloged later. If an alias name is specified in the DSNAME or PDS keyword, the true name only is moved or copied. Cataloging is done with the true name. The alias is removed if any uncataloging is done in a VSAM catalog.

IEHMOVE can be used to move or copy multivolume data sets. To move or copy a multivolume data set, specify the complete volume list in the VOL=SER parameter on the DD statement. To move or copy a data set that resides on more than one tape volume, specify the volume serial numbers of all the tape volumes and the sequence numbers of the data set on the tape volumes in the utility control statement. (You can specify the sequence number even if the data set to be moved or copied is the only data set on a volume.) To move or copy a data set to more than one tape volume, specify the volume serial numbers of all the receiving volumes in the utility control statement.

A data set with the unmovable attribute can be moved or copied from one DASD volume to another or to the same volume provided that space has been previously allocated on the receiving volume. Change the name of the data set if move or copy is to be done to the same volume. SVCLIB can be moved or copied to another location on the system residence volume, provided that space has been previously allocated on that volume. IEHPROGM must be used immediately after such a move operation to rename the moved version SYS1.SVCLIB. After such a copy operation, IEHPROGM must be used to scratch the old version and to rename the copied version.

When moving or copying a direct data set from one device to another device of the same type, relative track and relative block integrity are maintained.

When moving or copying a direct data set to a larger device, relative track integrity is maintained for data sets with variable or undefined record formats; relative block integrity is maintained for data sets with fixed record formats.

When moving or copying a direct data set to a smaller device or a tape, the data set is unloaded. An unloaded data set is loaded only when it is moved or copied to the same device type from which it was unloaded.

Figure 18-4 shows basic and optional move and copy operations for sequential and partitioned data sets.

Operation	**Basic Actions**	**Optional Actions**
Move Sequential	Move the data set. For DASD, scratch the source data. For cataloged data sets, update the catalog to refer to the moved data set.	Prevent automatic cataloging of the moved data set. Rename the moved data set.
Move Partitioned	Move the data set. Scratch the source data. For cataloged data sets, update the catalog to refer to the moved data set.	Prevent automatic cataloging of the moved data set. Rename the moved data set. Re-allocate directory space. (Not possible if the space was not allocated by IEHMOVE during this move function.) Perform a merge operation using members from two or more data sets. Move only selected members. Replace members. Unload the data set.
Copy Sequential	Copy the data set. The source data set is not scratched. The catalog is not updated to refer to the copied data set.	Uncatalog the source data set. Catalog the copied data set on the receiving volume. Rename the copied data set.
Copy Partitioned	Copy the data set. The source data is not scratched. The catalog is not updated to refer to the copied data set.	Uncatalog the source data set. Catalog the copied data set. Rename the copied data set. Re-allocate directory space. (Not possible if the space previously allocated is partially used.) Perform a merge operation using members from two or more data sets. Copy only selected members. Replace members. Unload the data set.

Figure 18-4. Moving and Copying Sequential and Partitioned Data Sets

IEHMOVE moves or copies partitioned members in the order in which they appear in the partitioned directory. That is, moved or copied members are placed in collating sequence on the receiving volume.

Figure 18-5 shows a copied partitioned data set. Note that the members are copied in the order in which they appear in the partitioned directory. IEBCOPY can be used to copy data sets whose members are not to be collated.

Members that are merged into an existing data set are placed, in collating sequence, after the last member in the existing data set. If the target data set contains a member with the same name as the from dataset, the member will not be moved/copied.

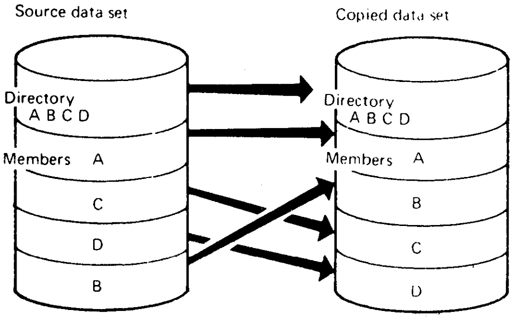


Figure 18-5. Partitioned Data Set Before and After an IEHMOVE Copy Operation

Figure 18-6 shows members from one data set merged into an existing data set. Members B and F are copied in collating sequence.

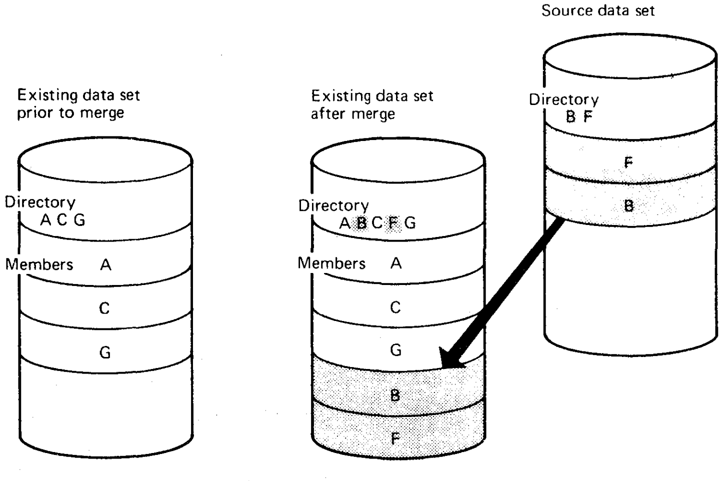
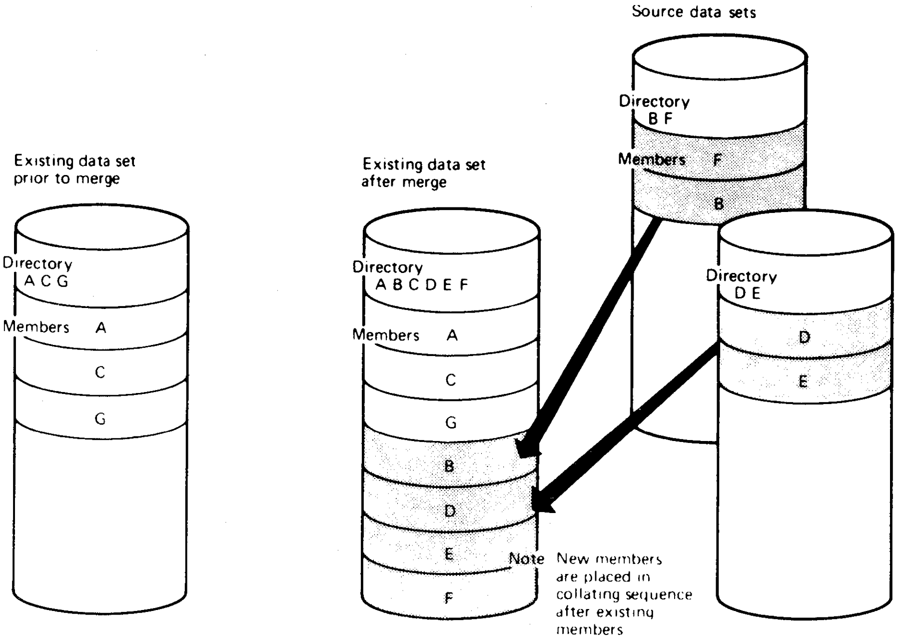


Figure 18-6. Merging Two Data Sets Using IEHMOVE

Figure 18-7 shows how members from two data sets are merged into an existing data set. Members from additional data sets can be merged in a like manner. Members F, B, D, and E from the source data sets are copied in collating sequence.



Note New members are placed in collating sequence after existing members

Figure 18-7.Merging Three Data Sets Using IEHMOVE

Moving or Copying a Group of Cataloged Data Sets

IEHMOVE can be used to move or copy a group of data sets that are cataloged in VSAM catalogs and whose names are qualified by one or more identical names. For example, a group of data sets qualified by the name A.B can include data sets named A.B.D and A.B.E, but could not include data sets named A.C.D or A.D.F.

If the user specifies that the data set group is cataloged in a CVOL, two additional options are available. First, additional data sets not belonging to the specified data set group can be included in the move or copy operation. Second, data sets belonging to the group can be excluded from the requested operation.

Additional data sets not belonging to the specified data set group can be included in the move or copy operation; data sets belonging to the group can be excluded.

Before copying/moving a DSGROUP that is cataloged in an OS catalog, the volume containing the catalog must be defined in the VSAM mater catalog. See *Using OS Catalog Management with the Master Catalog: CVOL Processor* for details on how this is done.

If a group of data sets is moved or copied to magnetic tape, the data sets must be retrieved one by one by data set name and file-sequence number, or by file-sequence number for unlabeled or non-standard labeled tapes.

Access Method Services can be used to determine the structure of the catalog.

Figure 18-8 shows basic and optional move and copy operations for a group of cataloged data sets.

Operation	Basic Actions	Optional Actions
Move group of cataloged data sets	Move the data set group (excluding password-protected data sets) to the specified volumes. Scratch the source data sets (direct access only). Merging is not done.	Prevent updating of the catalog. Include password-protected data sets in the operation, Unload data sets. If a data set group is cataloged in a CVOL: you may INCLUDE or EXCLUDE data sets during the operation.
Copy group of cataloged data sets	Copy the data set group (excluding password-protected data sets). Source data sets are not scratched. Merging is not done.	Include password-protected data sets in the operation. Uncatalog the source data sets. Catalog the copied data sets on the receiving volumes. Unload a data set or sets. If a data set group is cataloged in a CVOL: you may INCLUDE or EXCLUDE data sets during the operation.

Figure 18-8.Moving and Copying a Group of Cataloged Data Sets

Moving or Copying a Catalog

IEHMOVE can be used to move or copy an OS catalog (CVOL) or portions of an OS catalog without copying the data sets represented by the cataloged entries. If the catalog is in an unloaded form, all entries are moved or copied. The SYSCTLG (system catalog) data set need not be defined on the receiving volume before the operation. If, however, SYSCTLG was defined before the operation, the data set organization must not have been specified in the DCB field. Moved or copied entries are merged with any existing entries on the receiving volume. Note that the receiving volume must be a DASD volume unless the catalog is to be unloaded.

Figure 18-9 shows basic and optional move and copy operations for the catalog.

Operation	Basic Actions	Optional Actions
Move catalog	Move entries from the catalog to the specified direct access volume. Scratch the last index of all entries in the source catalog.	Exclude selected entries from operation. Move an unloaded version of the OS catalog. Unload the OS catalog.
Copy catalog	Copy entries from the catalog to the specified direct access device. The source catalog is not scratched.	Exclude selected entries from the operation. Copy an unloaded version of the OS catalog. Unload the OS catalog.

Figure 18-9. Moving and Copying the Catalog

Note: Before copying/moving an OS catalog both the volume containing the catalog and the volume to which the catalog is to be moved must be defined in the VSAM Master catalog.

Moving or Copying a Volume of Data Sets

IEHMOVE can be used to move or copy the data sets of an entire direct access volume to another volume or volumes. A move operation differs from a copy operation in that the move operation scratches source data sets, while the copy operation does not. For both operations, any cataloged entries associated with the source data sets remain unchanged. IEHPROGM can be used to uncatalog all of the cataloged data sets and recatalog them according to their new location.

If the source volume contains a SYSCTLG data set, that data set is the last to be moved or copied onto the receiving volume.

If a volume of data sets is moved or copied to tape, sequential data sets are 'moved' while partitioned and direct data sets are 'unloaded'. The data sets must be retrieved one by one by data set name and file-sequence number, or by file-sequence number for unlabeled or non-standard labeled tapes.

When copying a volume of data sets, the user has the option of cataloging all source data sets in a SYSCTLG data set on a receiving volume. However, if a SYSCTLG data set exists on the source volume, error messages indicating that an inconsistent index structure exists are generated when the source SYSCTLG entries are merged into the SYSCTLG data set on the receiving volume.

IEHMOVE ignores VSAM data spaces and system catalog data sets.

The move-volume feature does not merge partitioned data sets. If a data set on the volume to be moved has a name identical to a data set name on the receiving volume, the data set is not moved, or merged onto the receiving volume.

The copy-volume feature does merge partitioned data sets. If a data set on the volume to be copied has a name identical to a data set name on the receiving volume, the data set is copied and merged onto the receiving volume.

Figure 18-10 shows basic and optional move and copy operations for a volume of data sets.

Operation	**Basic Actions**	**Optional Actions**
Move a volume of data sets	Move all data sets not protected by a password to the specified direct access volumes. Scratch the source data sets for direct access volumes. The catalog is not updated.	Include password-protected data sets in the operation. Unload the data sets.
COPY a volume of data sets	Copy all data sets not protected by a password to the specified direct access volume. The source data sets are not scratched.	Include password-protected data sets in the operation. Catalog all copied data sets. Unload the data sets.

Figure 18-10.Moving and Copying a Volume of Data Sets

Moving or Copying Direct Data Sets with Variable Spanned Records

IEHMOVE can be used to move or copy direct data sets with variable spanned records from one direct access volume to a compatible direct access volume, provided that the record size does not exceed 32K.

Because a direct access data set can reside on one to five volumes (all of which must be mounted during any move or copy operation), it is possible for the data set to span volumes. However, single variable spanned records are contained on one volume.

Relative track integrity is preserved in a move or copy operation for spanned records. Moved or copied direct access data sets occupy the same relative number of tracks that they occupied on the source device.

If a direct data set is unloaded (moved or copied to a smaller device or tape), it must be loaded back to the same device type from which it was originally unloaded.

When moving or copying variable spanned records to a larger device, record segments are combined and re-spanned if necessary. Because the remaining track space is available for new records, variable spanned records are unloaded before being moved or copied back to a smaller device.

If a user wishes to create a direct data set without using data management BDAM macros, all data management specifications must be followed. Special attention must be given to data management specifications for R0 track capacity record content, segment descriptor words, and the BFTEK=R parameter.

When moving or copying a multivolume data set, the secondary allocation for direct data sets should be at least two tracks. (See the "WRITE SZ" macro in *OS/VS2 MVS Data Management Macro Instructions.*)

Input and Output

IEHMOVE uses the following input:

- One or more data sets, which contain the data to be moved, copied, or merged into an output data set.
- A control data set, which contains utility control statements that are used to control the functions of the program.
- A work data set, which is a work area used by IEHMOVE.

IEHMOVE produces the following output:

- An output data set, which is the result of the move, copy, or merge operation.
- A message data set, which contains informational messages (for example, the names of moved or copied data sets) and error messages, if applicable.

IEHMOVE produces a return code to indicate the results of program execution. The return codes and their meanings are:

- 00, which indicates successful completion.
- 04, which indicates that a specified function was not completely successful. Processing continues.
- 08, which indicates a condition from which recovery is possible. Processing continues.
- 12, which indicates an unrecoverable error. The job step is terminated.
- 16, which indicates that is is impossible to OPEN the SYSIN or SYSPRINT data set.

Control

IEHMOVE is controlled by job control statements and utility control statements. The job control statements are used to execute or invoke the program, define the devices and volumes used and produced by IEHMOVE, and prevent data sets from being deleted inadvertently.

Utility control statements are used to control the functions of the program and to define those data sets or volumes that are to be used.

Job Control Statements

Figure 18-11 shows the job control statements necessary for using IEHMOVE.

The SYSUT1 DD statement must be coded:

```
//SYSUT1 DD UNIT=xxxx,VOLUME=SER=xxxxxx,DISP=OLD
```

At least 3 utility work areas of 13, 13, and 26 contiguous tracks, respectively, must be available for work space on the volume defined by the SYSUT1 DD statement. (This figure is based on a 2314 being the work volume. If a direct access device other than a 2314 is used, an equivalent amount of space must be available.)

The anyname1 DD statement can be coded:

```
//anyname1 DD UNIT=xxxx,VOLUME=SER=xxxxxx,DISP=OLD
```

In the anyname1 DD statement, the UNIT and VOLUME parameters define the device type and volume serial number. The DISP=OLD specification prevents the inadvertent deletion of a data set. The anyname1 DD statement is arbitrarily assigned the ddname DD1 in the IEHMOVE examples.

The anyname2 DD statement can be coded:

```
//anyname2 DD UNIT=xxxx,VOLUME=SER=xxxxxx,DISP=OLD
```

When the number of volumes to be processed is greater than the number of devices defined by DD statements, there must be an indication (in the applicable DD statements) that multiple volumes are to be processed. This indication can be in the form of deferred mounting, as follows:

```
//anyname2 DD  UNIT=(xxxx,,DEFER),VOLUME=(PRIVATE,...),
//             DISP=(...,KEEP)
```

See "Appendix C: DD Statements for Defining Mountable Devices" for information on defining mountable devices. The anyname2 DD statement is arbitrarily assigned the ddname DD2 in the IEHMOVE examples. DD statements defining additional mountable device types are assigned names DD3, DD4, etc., when 7-track tape is to be used.

The tape DD statement can be coded:

```
//tape DD   DSNAME=xxxxxxxx,UNIT=xxxx,VOLUME=SER=xxxxxx,
//          DISP=(...,KEEP),LABEL=(...,...),DCB=(TRTCH=C,DEN=x)
```

A utility control statement parameter refers to the tape DD statement for label and mode information.

The date on which a data set is moved or copied to a magnetic tape volume is automatically recorded in the HDR1 record of a standard tape label if a TODD parameter is specified in a utility control statement. An expiration date can be specified by including the EXPDT or RETPD subparameters of the LABEL keyword in the DD statement referred to by a TODD parameter.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEHMOVE) or, if the job control statements reside in a procedure library, the procedure name. This statement can include optional PARM information; see "PARM Information on the EXEC Statement" below.
SYSPRINT DD	Defines a sequential message data set. The data set can be written onto a system output device, a magnetic tape volume, or a direct access volume.
SYSUT1 DD	Defines a volume on which 3 work data sets required by IEHMOVE are allocated.
anyname1 DD	Defines a permanently mounted DASD volume. (The system residence volume is considered to be a permanently mounted volume.)
anyname2 DD	Defines a mountable device type.
tape DD	Defines a mountable tape device.
SYSIN DD	Defines the control data set. The data set, which contains utility control statements, usually follows the job control statements in the input stream; however, it can be defined either as a sequential data set or as a member of a procedure library.

Figure 18-11.IEHMOVE Job Control Statements

A sequence number, for a data set on a tape volume, or a specific device address (for example, unit address 190), must be specified on a utility control statement instead of a DD statement. To move or copy a data set from or to a tape volume containing more than one data set, specify the sequence number of the data set in the utility control statement. To move or copy a data set from or to a specific device, specify the unit address (rather than a group name or device type) in the utility control statement. To copy to a unit record or unlabeled tape volume, specify any standard name or number in the utility control statement.

The tape DD statement can be used to communicate DCB attributes, of data sets residing on tape volumes that do not have standard labels, to IEHMOVE. If no DCB attributes are specified, an undefined record format and a block size of 2560 are assumed. However, in order to recognize unloaded data sets on an unlabeled tape volume, the DCB attributes must be specified as follows:

DCB=(RECFM=FB,LRECL=80,BLKSIZE=800).

IEHMOVE automatically calculates and allocates the amount of space needed for the work areas. No SPACE parameter, therefore, should be coded in the SYSUT1 DD statement. If, in the EXEC statement, POWER=3 is specified, the work space requirement is three times the basic requirements, etc.

With the exception of the SYSIN and SYSPRINT DD statements, all DD statements shown in Figure 18-11 are used as device allocation statements, rather than as true data definition statements. Because IEHMOVE modifies the internal control blocks created by device allocation DD statements, these statements must not include the DSNAME parameter. (All data sets are defined explicitly or implicitly by utility control statements.)

A merge operation requires that one DD statement defining a mountable device be present for each source volume containing data to be included in the merge operation.

Prior space allocations can be made by specifying a dummy execution of IEHPROGM before the execution of IEHMOVE.

Blocked format data sets that do not contain user data TTRNs or keys can be reblocked or unblocked by including the proper keyword subparameters in the DCB operand of the DD statement used to previously allocate space for the data set. The new blocking factor must be a multiple of the logical record length originally assigned to the data set. For a discussion of user data TTRNs, refer to *OS/VS2 MVS Data Management Services Guide.*

PARM Information on the EXEC Statement

The EXEC statement for IEHMOVE can contain PARM information that is used by the program to allocate additional work space and/or control line density on output listings. The EXEC statement can be coded, as follows:

```
// EXEC PGM=IEHMOVE[,PARM=    {'POWER=nnn'}
                              {'POWER=nnn,LINECNT=xx'}
                              {'LINECNT=xx'}]
```

The POWER=n parameter is used to request that the normal amount of space allocated for work areas be increased n times (1 to 999). The POWER parameter is used when 750 or more members are being moved or copied. The progression for the value of n is:

- POWER=2 when 750 to 1,500 members are to be moved or copied.
- POWER=3 when 1,501 to 2,250 members are to be moved or copied.
- POWER=4 when 2,251 to 3,000 members are to be moved or copied.

If POWER=2, the work space requirement on the SYSUT1 volume is two times the basic requirement; if POWER=3, work space requirement is three times the basic requirement, etc. For example, if POWER=2, three areas of 26, 26, and 52 contiguous tracks on a 2314 must be available.

When moving or copying an OS catalog, the value of the POWER parameter can be calculated, as follows:

$$n=(10D + V + 20G)/4000$$

where D is the total number of data sets, aliases, and generation data set entries (which is the number of data set names printed by IEHLIST when LISTCTLG is specified); V is the total number of volumes used by these data sets (which is the number of lines printed by IEHLIST when LISTCTLG is specified); and G is the number of generation data sets. Approximate values can be used:

- POWER=2 when 350 to 700 data sets are cataloged.
- POWER=3 when 701 to 1,050 data sets are cataloged.
- POWER=4 when 1,051 to 1,400 data sets are cataloged.

The LINECNT=xx parameter specifies the number of lines per page in the listing of the SYSPRINT data set; xx is a two-digit number in the range 04 through 99.

Job Control Language for the Track Overflow Feature

A data set containing track overflow records can be moved or copied if the source volume and the receiving volume are mounted on DASD that support the track overflow feature. (For direct data sets, the source and receiving devices must be the same device type.)

A data set that was written without track overflow can be moved or copied with or without track overflow or vice versa if the following conditions are met:

- Space was allocated for the data set prior to the request for a move or copy operation.
- The DD statement used for that allocation included the subparameter to specify the changed track overflow value and all other desired values. (The RECFM specifications assigned when the data set was originally created are overridden by the RECFM subparameter in this DD statement.)

If space has not been allocated, or if RECFM was not specified when space was allocated, the data set is moved or copied in accordance with RECFM specifications that were made when the data set was originally created.

The track overflow attribute is not retained for a sequential data set that is moved or copied to a device other than a DASD.

Utility Control Statements

IEHMOVE is controlled by the following utility control statements (see figure 18-12).

Statement	**Use**
MOVE DSNAME	Moves a data set.
COPY DSNAME	Copies a data set.
MOVE DSGROUP	Moves a group of cataloged data sets.
COPY DSGROUP	Copies a group of cataloged data sets.
MOVE PDS	Moves a partitioned data set.
COPY PDS	Copies a partitioned data set.
MOVE VOLUME	Moves a volume of data sets.
COPY VOLUME	Copies a volume of data sets.
MOVE CATALOG	Moves CVOL entries.
COPY CATALOG	Copies CVOL entries.

Figure 18-12. IEHMOVE Job Control Statements

In addition, there are four *subordinate* control statements that can be used to modify the effect of a MOVE DSGROUP, COPY DSGROUP, MOVE PDS, COPY PDS, MOVE CATALOG, or COPY CATALOG operation. The subordinate control statements are:

- INCLUDE statement, which is used to enlarge the scope of a MOVE DSGROUP (with CVOL), COPY DSGROUP (with CVOL), MOVE PDS, or COPY PDS statement by including a member or data set not explicitly included by the statement it modifies.
- EXCLUDE statement, which is used with a MOVE DSGROUP (with CVOL), COPY DSGROUP (with CVOL), MOVE PDS, COPY PDS, MOVE CATALOG, or COPY CATALOG statement to exclude data set(s), a member or catalog entry(ies) from a move or copy operation.
- REPLACE statement, which is used with a MOVE PDS or COPY PDS statement to exclude a member from a move or copy operation and to replace it with a member from another partitioned data set.

- SELECT statement, which is used with MOVE PDS or COPY PDS statements to select members to be moved or copied and, optionally, to rename the specified members.

FROM and CVOL should never appear in the same IEHMOVE utility control statement. FROMDD must be specified in the control statement when no data set label information is available. TODD must be specified in the control statement when an expiration data (EXPDT) or retention period (RETPD) is to be created or changed.

MOVE DSNAME Statement

The MOVE DSNAME statement is used to move a data set. The source data set is scratched.

If the data set is cataloged, the catalog is automatically updated unless UNCATLG/FROM is specified.

The format of the MOVE DSNAME statement is:

[*label*] **MOVE** **DSNAME**=*name*
,TO=*device* =*list*
[{**,FROM**=*device* =*list* | **,CVOL**=*device* =*serial*}]
[,UNCATLG]
[**,RENAME**=*name*]
[**,FROMDD**=*ddname*]
[**,TODD**=*ddname*]
[,UNLOAD]
[,COPYAUTH]

COPY DSNAME Statement

The COPY DSNAME statement is used to copy a data set.

The source data set, if cataloged, remains cataloged unless UNCATLG or CATLG without CVOL is specified.

The format of the COPY DSNAME statement is:

[*label*] **COPY** **DSNAME**=*name*
,TO=*device* =*list*
[{**,FROM**=*device* =*list* | **,CVOL**=*device* =*serial*}]
[,UNCATLG]
[,CATLG]
[**,RENAME**=*name*]
[**,FROMDD**=*ddname*]
[**,TODD**=*ddname*]
[,UNLOAD]
[,COPYAUTH]

MOVE DSGROUP Statement

The MOVE DSGROUP statement is used to move groups of data sets whose names are partially qualified by one or more identical names. The data sets may be cataloged on several catalogs. Source data sets are scratched. Data set groups to be moved must reside on direct access volumes. Only data sets that could be moved by MOVE DSNAME or MOVE PDS can be moved by MOVE DSGROUP. Alias entries in VSAM catalogs for the data sets are lost and can be replaced with Access Method Services.

INCLUDE and EXCLUDE statements, discussed later in this chapter, can be used to add to or delete data sets from the group, if CVOL is specified.

MOVE DSGROUP operations cause the catalog to be updated automatically unless UNCATLG is specified.

The format of the MOVE DSGROUP statement is:

[*label*] **MOVE** **DSGROUP[**=*name*]
,TO=*device* =*list*
[**,CVOL**=*device* =*serial*]
[,PASSWORD]
[,UNCATLG]
[**,TODD**=*ddname*]
[,UNLOAD]
[,COPYAUTH]

COPY DSGROUP Statement

The COPY DSGROUP statement is used to copy groups of data sets whose names are partially qualified by one or more identical names. The data sets may be cataloged on several catalogs. Only data sets that can be copied with COPY DSNAME or COPY PDS can be copied with COPY DSGROUP. Data set groups to be copied must reside on DASD volumes.

INCLUDE and EXCLUDE statements, discussed later in this chapter, can be used to add to or delete data sets from the group, if CVOL is specified.

The source data sets remain cataloged unless UNCATLG or CATLG without CVOL is specified.

The format of the COPY DSGROUP statement is:

[*label*] **COPY** **DSGROUP[**=*name*]
,TO=*device* =*list*
[**,CVOL**=*device* =*serial*]
[,PASSWORD]
[,UNCATLG]
[,CATLG]
[**,TODD**=*ddname*]
[,UNLOAD]
[,COPYAUTH]

MOVE PDS Statement

The MOVE PDS statement is used to move partitioned data sets. When used in conjunction with INCLUDE, EXCLUDE, REPLACE, or SELECT statements, the MOVE PDS statement can be used to merge selected members of several partitioned data sets or to delete members.

If IEHMOVE is used to allocate space for an output partitioned data set, the MOVE PDS statement can be used to expand a partitioned directory.

If the receiving volume contains a partitioned data set with the same name, the two data sets are merged. The source data set is scratched.

MOVE PDS causes the specified catalog to be updated automatically unless UNCATLG/FROM is specified.

The format of the MOVE PDS statement is:

[*label*] **MOVE** **PDS=***name*
,TO=*device =serial*
[{**,FROM=***device =serial* | **,CVOL=***device =serial* }]
[**,EXPAND=***nn*]
[**,UNCATLG**]
[**,RENAME=***name*]
[**,FROMDD=***ddname*]
[**,TODD=***ddname*]
[**,UNLOAD**]
[**,COPYAUTH**]

COPY PDS Statement

The COPY PDS statement is used to copy partitioned data sets. When used in conjunction with INCLUDE, EXCLUDE, REPLACE, or SELECT statements, the COPY PDS statement can be used to merge selected members of several partitioned data sets or to delete members.

If IEHMOVE is used to allocate space for an output partitioned data set, the COPY PDS statement can be used to expand a partitioned directory.

If the receiving volume already contains a partitioned data set with the same name, the two are merged.

The source partitioned data set remains cataloged unless UNCATLG or CATLG without CVOL is specified. The format of the COPY PDS statement is:

[*label*] **COPY** **PDS=***name*
,TO=*device =serial*
[{**,FROM=***device =serial* | **,CVOL=***device =serial*}]
[**,EXPAND=***nn*]
[**,UNCATLG**]
[**,CATLG**]
[**,RENAME=***name*]

[,FROMDD=*ddname***]**

[,TODD=*ddname***]**

[,UNLOAD]

[,COPYAUTH]

MOVE CATALOG Statement

The MOVE CATALOG statement is used to move the entries of a CVOL catalog (SYSCTLG data set) without moving the data sets associated with those entries. Certain entries can be excluded from the operation by means of the EXCLUDE statement. If the receiving volume contains a CVOL, the source CVOL entries are merged with it.

The format of the MOVE CATALOG statement is:

[*label*] **MOVE** **CATALOG[=***name***]**

TO=*device =serial*

[{**,CVOL=** *device= serial* | **,FROM=** *device = serial*}]

[,FROMDD=*ddname***]**

[,TODD=*ddname***]**

[,UNLOAD]

[,COPYAUTH]

COPY CATALOG Statement

The COPY CATALOG statement is used to copy the entries of a CVOL cata (SYSCTLG data set) without copying the data sets associated with these entri Certain entries can be excluded from a copy operation with the EXCLUDE statement. If the receiving volume contains a CVOL, the source CVOL is merged with it.

The format of the COPY CATALOG statement is:

[*label*] **COPY** **CATALOG[=***name***]**

,TO=*device =serial*

[{**,CVOL=** *device =serial* | **,FROM=** *device = serial*}]

[,FROMDD=*ddname***]**

[,TODD=*ddname***]**

[,UNLOAD]

[,COPYAUTH]

MOVE VOLUME Statement

The MOVE VOLUME statement is used to move all the data sets residing on a specified volume. Catalog entries associated with the data sets remain unchanged. Data sets to be moved must reside on direct access volumes.

The format of the MOVE VOLUME statement is:

[*label*] **MOVE** **VOLUME=***device =serial*
,TO=*device =list*
[,PASSWORD]
[,TODD=*ddname*]
[,UNLOAD]
[,COPYAUTH]

COPY VOLUME Statement

The COPY VOLUME statement is used to copy all the data sets residing on a specified volume. Catalog entries associated with the data sets remain unchanged. Data sets to be copied must reside on direct access volumes.

If CATLG is specified, error messages indicating that an inconsistent index structure exists are issued when the source SYSCTLG data set entries are merged into the CVOL catalog on the receiving volume. (Because the SYSCTLG data set is the last to be copied, only those entries representing cataloged data sets not residing on the source volume are copied into a receiving volume's SYSCTLG data set; entries representing all data sets residing on the source volume have already been made in the receiving SYSCTLG data set.)

The format of the COPY VOLUME statement is:

[*label*] **COPY** **VOLUME=***device =serial*
,TO=*device =list*
[,PASSWORD]
[,CATLG]
[,TODD=*ddname*]
[,UNLOAD]
[,COPYAUTH]

INCLUDE Statement

The INCLUDE statement is used to enlarge the scope of MOVE DSGROUP, COPY DSGROUP, MOVE PDS, or COPY PDS statements by including a member or a data set not explicitly defined in those statements. The INCLUDE statement follows the MOVE or COPY statement whose function it modifies. The record characteristics of the included partitioned data sets must be compatible with those of the other partitioned data sets being moved or copied. Any number of INCLUDE statements can modify a MOVE or COPY statement. For a PDS, the INCLUDE statement is invalid when data is unloaded or when unloaded data is moved or copied. For DSGROUP operations, INCLUDE is invalid unless CVOL has been specified on the MOVE/COPY DSGROUP control statement.

The format of the INCLUDE statement is:

[*label*] **INCLUDE** **DSNAME**=*name*
[**,MEMBER**=*membername*]
[{**,FROM**=*device =list* | **,CVOL**=*device =serial*}]

EXCLUDE Statement

The EXCLUDE statement is used to restrict the scope of MOVE DSGROUP, COPY DSGROUP, MOVE PDS, COPY PDS, MOVE CATALOG, or COPY CATALOG statements by excluding a specific portion of data defined in those statements.

Partitioned data set members excluded from a MOVE PDS operation cannot be recovered (the source data set is scratched). Any number of EXCLUDE statements can modify a MOVE PDS or COPY PDS statement.

Source data sets or catalog entries excluded from a MOVE DSGROUP or MOVE CATALOG operation remain available. Only one EXCLUDE statement can modify a MOVE DSGROUP, COPY DSGROUP, MOVE CATALOG, or COPY CATALOG statement. The EXCLUDE statement is invalid when data is unloaded or when unloaded data is moved or copied. The EXCLUDE statement is invalid for a DSGROUP operation unless CVOL is specified on the MOVE/COPY DSGROUP control statement.

The format of the EXCLUDE statement is:

[*label*] **EXCLUDE** {**DSGROUP**=*name* | **MEMBER**=*membername*}

SELECT Statement

The SELECT statement is used with the MOVE PDS or COPY PDS statement to select members to be moved or copied, and to optionally rename these members. The SELECT statement cannot be used with either the EXCLUDE or REPLACE statement to modify the same MOVE PDS or COPY PDS statement. The SELECT statement is invalid when data is unloaded or when unloaded data is moved or copied. Members not selected in a MOVE PDS operation cannot be recovered since the source data set is scratched.

The format of the SELECT statement is:

[*label*] **SELECT** {**MEMBER**=(*name* [, *name*]...) |
MEMBER=((*name*, *newname*)[,(*name*, *newname*)]. . .)}

REPLACE Statement

The REPLACE statement is used with a MOVE PDS or COPY PDS statement to exclude a member from the operation and replace it with a member from another partitioned data set. The *new* member must have the same name as the *old* member and must possess compatible record characteristics. Any number of REPLACE statements can modify a MOVE PDS or COPY PDS statement. The REPLACE statement is invalid when data is unloaded or when unloaded data is moved or copied.

The format of the REPLACE statement is:

[*label*] **REPLACE** **DSNAME**=*name*
,MEMBER=*name*
[{**,FROM**=*device =serial* | **,CVOL**=*device =serial*}]

Operands	Applicable Control Statements	Description of Operands/Parameters
CATALOG	MOVE CATALOG COPY CATALOG	**CATALOG[=*name*]** specifies the CVOL catalog entries to be moved or copied. If *name,* which is a fully qualified name, is not coded, all entries in the catalog are to be moved or copied. If *name* is coded, all catalog entries whose names are qualified by this name are moved or copied. If the name is a fully qualified data set name, only the catalog entry that corresponds to that data set is moved or copied.
CATLG	COPY DSNAME COPY DSGROUP COPY PDS COPY VOLUME	**CATLG** specifies that the copied data set(s) is to be cataloged. If the CVOL operand is omitted, the cataloging is done in the VSAM master/JOBCAT/STEPCAT catalog. If the RENAME and FROM operands are omitted, the source data set(s) is uncataloged to permit the copied data set(s) to be cataloged. If CVOL operand is specified, the cataloging is done in the OS catalog on the receiving DASD volume. If an OS catalog does not exist on the receiving DASD volume, one is created.
COPYAUTH	MOVE DSNAME COPY DSNAME MOVE DSGROUP COPY DSGROUP MOVE PDS COPY PDS MOVE CATALOG COPY CATALOG MOVE VOLUME COPY VOLUME	COPYAUTH specifies that the receiving data set is to be given the same access list as the input data set, if the input data set is RACF-protected.
CVOL	MOVE DSNAME COPY DSNAME MOVE PDS COPY PDS INCLUDE REPLACE	CVOL =*device*=*serial* specifies the device type and serial number of the CVOL on which the catalog search for the data set is to begin. If the CVOL and FROM operands are omitted, the data set is assumed to be cataloged in the VSAM master/JOBCAT/STEPCAT catalog.
	MOVE DSGROUP COPY DSGROUP	**CVOL=*device*=*serial*** specifies the device type and serial number of the CVOL on which the catalog search for the data set(s) is to begin. If the CVOL operand is omitted, the data set(s) is assumed to be cataloged in the VSAM master/JOBCAT/STEPCAT catalog.
	MOVE CATALOG COPY CATALOG	**CVOL=*device*=*serial*** specifies the device type and serial number of the CVOL from which the SYSCTLG data set is to be moved or copied. If the CVOL and FROM operands are omitted, the SYSCTLG data set to be moved or copied is assumed to reside on the system residence volume.

Operands	Applicable Control Statements	Description of Operands/Parameters
DSGROUP	MOVE DSGROUP COPY DSGROUP	**DSGROUP=***name* specifies the cataloged data set(s) to be moved or copied. If *name* is a fully qualified data set name, that data set is not moved or copied. If *name* is one or more qualifiers, all data sets whose names are qualified by *name* are moved or copied. If *name* is omitted, all data sets whose names are found in the searched catalog are moved or copied.
	EXCLUDE	**DSGROUP=***name* Specifies the cataloged data set(s) or the catalog entry(ies) to be excluded in a MOVE/COPY DSGROUP or CATALOG operation. If used in conjunction with MOVE/COPY DSGROUP, all cataloged data sets whose names are qualified by *name* are excluded from the operation. If used in conjunction with MOVE/COPY CATALOG, all catalog entries whose names are qualified by *name* are excluded from the operation.
DSNAME	MOVE DSNAME COPY DSNAME	**DSNAME=***name* specifies the fully qualified name of the data set to be moved or copied.
	INCLUDE	**DSNAME=***name* specifies the fully qualified name of a data set. If used in conjunction with MOVE/COPY DSGROUP, the named data set is included in the group. If used in conjunction with MOVE/COPY PDS, either the named partitioned data set or a member of it is included in the operation.
	REPLACE	**DSNAME=***name* specifies the fully qualified name of the partitioned data set that contains the replacement member.
EXPAND	MOVE PDS COPY PDS	**EXPAND=***nn* specifies the number of 256-byte records (up to 99 decimal) to be added to the directory of the specified partitioned data set. For COPY, EXPAND cannot be specified if space is previously allocated. For MOVE, EXPAND will be ignored if space is previously allocated.

Operands	Applicable Control Statements	Description of Operands/Parameters
FROM	MOVE DSNAME COPY DSNAME MOVE PDS COPY PDS INCLUDE SELECT MOVE CATALOG COPY CATALOG	**FROM**=*device*={ *list* | *serial*} specifies the device type and serial number(s) of the volume(s) on which the data set resides if it is not cataloged. If the data set is cataloged, FROM should not be specified. The *serial* subparameter applies to PDS and CATALOG operations. The *list* subparameter applies to DSNAME operations, but may also be used when refering to an unloaded PDS residing on more than one DASD or tape volume, and when refering to an unloaded catalog residing on more than one tape volume. When FROM is used in conjunction with a MOVE, DSNAME/PDS operation, the catalog will not be updated. When FROM is used in conjunction with a MOVE/COPY CATALOG operation, it specifies where an unloaded version of the catalog resides. When FROM refers to a tape device and the data set to be retrieved is not the first on the volume, the *serial* subparameter must be enclosed in parentheses and the volume serial number must be followed by the data set sequence number and separated from it by a comma. When FROM is to refer to a specific device, code the unit address in the *device* parameter, in place of device type. If FROM and CVOL operands are omitted from a MOVE/COPY DSNAME/PDS, INCLUDE or REPLACE operation, the data set is assumed to be cataloged in the VSAM master/JOBCAT/STEPCAT catalog. If FROM and CVOL operands are omitted from a MOVE/COPY CATALOG operation, the SYSCTLG data set to be moved or copied is assumed to reside on the system residence volume.
FROMDD	MOVE DSNAME COPY DSNAME MOVE PDS COPY PDS MOVE CATALOG COPY CATALOG	**FROMDD**=*ddname* specifies the name of the DD statement from which DCB and LABEL information (except data set sequence number), for input data sets on tape volumes, can be obtained. When FROMDD is used in conjunction with a MOVE/COPY PDS/CATALOG operation, the tape data set must be an unloaded version of a partitioned data set or an unloaded version of a catalog. The FROMDD operand can be omitted, provided the data set has standard labels and resides on a 9-track tape volume.

Operands	Applicable Control Statements	Description of Operands/Parameters
MEMBER	INCLUDE REPLACE	**MEMBER**=*membername* specifies the name of the partitioned data set named in the DSNAME parameter on the INCLUDE/REPLACE statement. When coded on an INCLUDE statement, the member is merged with the partitioned data set being moved or copied. When coded on a REPLACE statement, the member replaces an equally named member in the partitioned data set being moved or copied. Regardless of the operation, neither the partitioned data set containing the named member nor the member is scratched.
	EXCLUDE	**MEMBER**=*name* specifies the name of a member to be excluded from a MOVE/COPY PDS operation
	SELECT	**MEMBER**={*name* | (*name*[,*name*]...) | ((*name,newname*) [,(*name, newname*)]...)} specifies the names of the members to be moved or copied by a MOVE/COPY PDS operation, and optionally new names to be assigned to the members.
PASSWORD	MOVE DSGROUP COPY DSGROUP MOVE VOLUME COPY VOLUME	**PASSWORD** specifies that password protected data sets are to be included in the operation. This is not VSAM password protection, but the OS password scheme. **Default**: Only data sets that are not protected ane copied or moved.
PDS	MOVE PDS COPY PDS	**PDS**=*name* specifies the fully qualified name of the partitioned data set to be moved or copied.
RENAME	MOVE DSNAME COPY DSNAME MOVE PDS COPY PDS	**RENAME**=*name* specifies that the data set is to be renamed, and indicates the new name.
TO	MOVE DSNAME COPY DSNAME MOVE DSGROUP COPY DSGROUP MOVE VOLUME COPY VOLUME	**TO**=*device*=*list* specifies the device type and volume or volumes to which the specified group of data sets is to be moved or copied.
	MOVE PDS COPY PDS MOVE CATALOG COPY CATALOG	**TO**=*device*=*serial* specifies the device type and volume serial number of the volume to which the partitioned data set or catalog entry is to be moved or copied. The *list* parameter may be used when unloading a partitioned data set that must span tape volumes.

Operands	Applicable Control Statements	Description of Operands/Parameters
TODD	MOVE DSNAME COPY DSNAME MOVE DSGROUP COPY DSGROUP MOVE PDS COPY PDS MOVE VOLUME COPY VOLUME MOVE CATALOG COPY CATALOG	**TODD**=*ddname* specifies the name of a DD statement from which DCB (except RECFM, BLKSIZE and LRECL) and LABEL (except data set sequence number) information for output data sets on tape volumes, can be obtained. When TODD is used in conjunction with a MOVE/COPY DSNAME/DSGROUP/VOLUME operation, it describes the mode and label information to be used when creating output data sets on tape volumes. RECFM, BLKSIZE, and LRECL information, if coded, is ignored. When UNLOAD is specified, or when TODD is used in conjunction with a MOVE/COPY PDS/CATALOG operation, it describes the mode and label information to be used when creating unloaded versions of data sets on tape volumes. RECFM, BLKSIZE, and LRECL information, if coded, must specify (RECFM=FB, BLKSIZE=800, LRECL=80). The TODD operand can be omitted for 9-track tapes with standard labels and default density for the unit type specified.
UNCATLG	MOVE DSNAME COPY DSNAME MOVE DSGROUP COPY DSGROUP MOVE PDS COPY PDS	**UNCATLG** specifies that the catalog entry pertaining to the source partitioned data set is to be removed. This parameter should be used only if the source data set is cataloged. If the volume is identified by FROM, UNCATLG is ignored. Alias entries in VSAM catalogs for the source data sets are lost and can be replaced with Access Method Services if the data sets are later cataloged. For a MOVE operation, UNCATLG inhibits cataloging of the output data set.
UNLOAD	MOVE DSNAME COPY DSNAME MOVE DSGROUP COPY DSGROUP MOVE PDS COPY PDS MOVE VOLUME COPY VOLUME MOVE CATALOG COPY CATALOG	**UNLOAD** specifies that the data set is to be unloaded to the receiving volume(s).
VOLUME	MOVE VOLUME COPY VOLUME	**VOLUME**=*device*=*serial* specifies the device type and volume serial number of the source volume.

Restrictions

- The block size for the SYSPRINT data set must be a multiple of 121. The block size for the SYSIN data set must be a multiple of 80. Any blocking factor can be specified for these block sizes.
- One anyname1 DD statement must be included for each permanently mounted volume referred to in the job step.
- One anyname2 DD statement must be included for each mountable device to be used in the job step.
- When IEHMOVE is dynamically invoked in a job step containing another program, the DD statements defining mountable devices for IEHMOVE must be included in the job stream prior to DD statements defining data sets required by the other program.
- VIO is supported by IEHMOVE only for SYSIN and SYSPRINT.
- The "TO" data set must be cataloged in the VSAM master catalog to run a move of a CVOL catalog using IEHMOVE.
- DSNAME must be in the form of SYSCTLG.V*serial*
- When unloading a DASD data set to another DASD data set, the data set name (DSN=) must be coded on the DD-card for the data set to be unloaded. If the output (unloaded) data set was not preallocated, all unused space will be released.

IEHMOVE Examples

The following examples illustrate some of the uses of IEHMOVE. Figure 18-13 can be used as a quick reference guide to IEHMOVE examples. The numbers in the "Example" column point to the examples that follow.

Operation	Data Set Organization	Device	Comments	Example
MOVE	Sequential	Disk	Source volume is demounted after job completion. Two mountable disks.	1
COPY	Sequential	Disk	Three cataloged sequential data sets are to be copied. The disks are mountable.	2
MOVE	Partitioned	Disk	A partitioned data set is to be moved; a member from another PDS is to be merged with it.	3
MOVE	Volume	Disk		4
MOVE	Partitioned	Disk	A data set is to be moved to a volume on which space was previously allocated.	5
MOVE	Partitioned	Disk	Three data sets are to be moved and unloaded to a volume on which space was previously allocated.	6
MOVE	Sequential	Disk and Tape	A sequential data set is to be unloaded to an unlabeled 9-track tape volume.	7
MOVE	Sequential	Disk and Tape	Unloaded data sets are to be loaded from a single volume.	8
COPY	Sequential	Disk and Tape	Data sets are to be copied from separate source volumes.	9
COPY	Partitioned	Tape and Disk	Unloaded data sets are to be loaded from unlabeled tape to a specific device.	10
MOVE	Data Set Group	Disk	Data set group is to be moved. The 2314 disks are mountable.	11
MOVE	CVOL	Disk	SYSCTLG data set (CVOL) is to be moved from one volume to another. Source CVOL is scratched.	12
MOVE	CVOL	Disk	Selected CVOL catalog entries are moved from one CVOL to another.	13

Figure 18-13. IEHMOVE Example Directory

Note: Examples which use *disk* or *tape* in place of actual device-ids, must be changed before use. See the Device Support section in the Introduction to this manual for valid device-id notation.

IEHMOVE Example 1

In this example, three data sets (SEQSET1, SEQSET2, and SEQSET3) are to be moved from a disk volume to three separate disk volumes. Each of the three receiving volumes is mounted when it is required by IEHMOVE. The source data sets are not cataloged. Space is allocated by IEHMOVE.

```
//MOVEDS   JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=333333,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=(disk,,DEFER),DISP=OLD,
// VOLUME=(PRIVATE,,SER=(222222))
//DD3      DD   VOLUME=(PRIVATE,RETAIN,SER=(444444)),
// UNIT=disk,DISP=OLD
//SYSIN    DD   *
       MOVE    DSNAME=SEQSET1,TO=disk=222222,FROM=disk=444444
       MOVE    DSNAME=SEQSET2,TO=disk=222333,FROM=disk=444444
       MOVE    DSNAME=SEQSET3,TO=disk=222444,FROM=disk=444444
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set.
- DD1 DD defines the system residence device.
- DD2 DD defines the mountable device on which the receiving volumes will be mounted as they are required.
- DD3 DD defines a mountable device on which the source volume is to be mounted. Because the RETAIN subparameter is included, the volume remains mounted until the job has completed.
- SYSIN DD defines the control data set, which follows in the input stream.
- MOVE moves the source data sets to volumes 222222, 222333, and 222444, respectively. The source data sets are scratched.

IEHMOVE Example 2

In this example, three cataloged data sets are to be copied to a disk volume. Space is allocated by IEHMOVE. The catalog is not updated. The source data sets are not scratched.

```
//COPYPDS  JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD3      DD   UNIT=disk,VOLUME=SER=333333,DISP=OLD
//SYSIN    DD   *
         COPY   DSNAME=SEQSET1,TO=disk=333333
         COPY   DSNAME=SEQSET3,TO=disk=333333
         COPY   DSNAME=SEQSET4,TO=disk=333333
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set.
- DD1 DD defines the system residence device.
- DD2 DD defines a mountable device on which the source volume is mounted.

- DD3 DD defines a mountable device on which the receiving volume is mounted.
- SYSIN DD defines the control data set which follows in the input stream.
- COPY copies the source data sets onto volume 333333.

IEHMOVE Example 3

In this example, a partitioned data set (PARSET1) is to be moved to a disk volume. In addition, a member (PARMEM3) from another partitioned data set (PARTSET2) is to be merged with the source members on the receiving volume. The source partitioned data set (PARTSET1) is scratched. Space is allocated by IEHMOVE.

```
//MOVEPDS  JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=333000,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=disk,VOLUME=SER=222111,DISP=OLD
//DD3      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD4      DD   UNIT=disk,VOLUME=SER=222333,DISP=OLD
//SYSIN    DD   *
     MOVE    PDS=PARTSET1,TO=disk=222333,FROM=disk=222111
  INCLUDE    DSNAME=PARTSET2,MEMBER=PARMEM3,FROM=disk=222222
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set.
- DD1 DD defines the system residence device.
- The DD2, DD3, and DD4 DD statements define mountable devices that are to contain the two source volumes and the receiving volume.
- SYSIN DD defines the control data set, which follows in the input stream.
- MOVE defines the source partitioned data set, the volume that contains it, and its receiving volume.
- INCLUDE includes a member from a second partitioned data set in the operation.

IEHMOVE Example 4

In this example, a volume of data sets is to be moved to a disk volume. All data sets that are successfully moved are scratched from the source volume; however, any catalog entries pertaining to those data sets are not changed. Space is allocated by IEHMOVE. The work data set is deleted when the job step is completed.

```
//MOVEVOL  JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD3      DD   UNIT=disk,VOLUME=SER=333333,DISP=OLD
//SYSIN    DD   *
        MOVE    VOLUME=disk=333333,TO=disk=222222,PASSWORD
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set. The work data set is removed from the receiving volume when the job step is completed.
- DD1 DD defines the system residence device.
- DD2 DD defines the mountable device on which the receiving volume is to be mounted.
- DD3 DD defines a mountable device on which the source volume is to be mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- MOVE specifies a move operation for a volume of data sets and defines the source and receiving volumes. This statement also indicates that password-protected data sets are to be included in the operation.

Note: IEHPROGM can be used to uncatalog catalog entries pertaining to non-VSAM source data sets and to catalog the moved versions of those data sets.

IEHMOVE Example 5

In this example, a partitioned data set is to be moved to a disk volume on which space has been previously allocated for the data set. The source data set is scratched. The work data set is deleted when the job step is completed.

```
//ALLOCATE JOB  09#550,GREEN
//          EXEC PGM=IEFBR14
//SET1      DD   DSNAME=PDSSET1,UNIT=disk,DISP=(NEW,KEEP),
// VOLUME=SER=222222,SPACE=(TRK,(100,10,10)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//          EXEC PGM=IEHMOVE
//SYSPRINT  DD   SYSOUT=A
//SYSUT1    DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD1       DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2       DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD3       DD   UNIT=disk,VOLUME=SER=333333,DISP=OLD
//SYSIN     DD   *
        MOVE     PDS=PDSSET1,TO=disk=222222,FROM=disk=333333
/*
```

The IEFBR14 job step is used to allocate space for data set PDSSET1 on a disk volume.

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set. The data set is removed from the receiving volume at the completion of the program.
- DD1 DD defines the system residence device.
- DD2 DD defines the device on which the receiving volume is to be mounted.
- DD3 DD defines a mountable device on which the source volume is to be mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- MOVE specifies a move operation for the partitioned data set PDSSET1 and defines the source and receiving volumes.

IEHMOVE Example 6

In this example, three partitioned data sets are to be moved from three separate source volumes to a disk volume. The source data set PDSSET3 is unloaded. (The record size exceeds the track capacity of the receiving volume.) The work data set is deleted when the job step is completed.

```
                                                              72
//ALLOCATE JOB  09#550,GREEN
//         EXEC PGM=IEFBR14
//SET1     DD   DSNAME=PDSSET1,UNIT=disk,DISP=(NEW,KEEP),
// VOLUME=SER=222222,SPACE=(TRK,(50,10,5)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=1600)
//SET2     DD   DSNAME=PDSSET2,UNIT=disk,DISP=(NEW,KEEP),
// VOLUME=SER=222222,SPACE=(TRK,(25,5,5)),
// DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SET3     DD   DSNAME=PDSSET3,UNIT=disk,DISP=(NEW,KEEP),
// VOLUME=SER=222222,SPACE=(TRK,(25,5)),
// DCB=(RECFM=U,BLKSIZE=5000)
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=(disk,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(333333))
//DD3      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//SYSIN    DD   *
        MOVE    PDS=PDSSET1,TO=disk=222222,FROM=disk=333333
        MOVE    PDS=PDSSET2,TO=disk=222222,FROM=disk=222222
        MOVE    PDS=PDSSET3,TO=disk=222222,
                FROM =disk=444444,UNLOAD                       C
/*
```

The IEFBR14 job step is used to allocate space for the partitioned data sets PDSSET1, PDSSET2, and PDSSET3 on the receiving volume. The SPACE parameter in the SET3 DD statement allocates space for a sequential data set. This is necessary to successfully unload the partitioned data set PDSSET3. The DCB attributes of PDSSET3 are:

DCB=(RECFM=U,BLKSIZE=5000)

The unloaded attributes are:

DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set.
- DD1 DD defines the system residence device.
- DD2 DD defines a mountable device on which the source volumes are mounted as they are required.
- DD3 DD defines a mountable device on which the receiving volume is mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- MOVE specifies move operations for the partitioned data sets and defines the source and receiving volumes.

Note: For a discussion on estimating space allocations, refer to *OS/VS2 MVS Data Management Services Guide.*

IEHMOVE Example 7

In this example, a sequential data set is to be unloaded onto a 9-track, unlabeled tape volume (800 bits per inch). The work data set resides on the source volume and is deleted when the job step is completed.

```
                                                                       72
//UNLOAD     JOB  09#550,GREEN
//           EXEC PGM=IEHMOVE
//SYSPRINT   DD   SYSOUT=A
//SYSUT1     DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD1        DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2        DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//TAPEOUT    DD   UNIT=tape,VOLUME=SER=SCRTCH,DISP=OLD,
// DCB=(DEN=2,RECFM=FB,LRECL=80,BLKSIZE=800),
// LABEL=(,NL)
//SYSIN      DD   *
         MOVE     DSNAME=SEQSET1,TO=tape=SCRTCH,                        C
               FROM=disk=222222,TODD=TAPEOUT
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set.
- DD1 DD defines the system residence device.
- DD2 DD defines a mountable device on which the source volume is mounted.
- TAPEOUT DD defines a mountable device on which the receiving tape volume is mounted. This statement also provides label and mode information.
- SYSIN DD defines the control data set which follows in the input stream.
- MOVE moves the sequential data set SEQSET1 from a disk volume to the receiving tape volume. The data set is unloaded. The TODD parameter in this statement refers to the TAPEOUT DD statement for label and mode information.

IEHMOVE Example 8

In this example, three unloaded sequential data sets are to be loaded from a labeled, 7-track tape volume (556 bits per inch) to a disk volume. Space is allocated by IEHMOVE. The example assumes that the disk volume is capable of supporting the data sets in their original forms.

```
                                                                       72
//LOAD       JOB  09#550,GREEN
//           EXEC PGM=IEHMOVE
//SYSPRINT   DD   SYSOUT=A
//SYSUT1     DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD1        DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2        DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//TAPESETS   DD   DSNAME=UNLDSET1,UNIT=2400-2,
// VOLUME=SER=001234,DISP=OLD,
// LABEL=(1,SL),DCB=(DEN=1,TRTCH=C)
//SYSIN      DD   *
         MOVE     DSNAME=UNLDSET1,TO=disk=222222,                       C
               FROM=2400-2=(001234,1),FROMDD=TAPESETS
         MOVE     DSNAME=UNLDSET2,TO=disk=222222,                       C
               FROM=2400-2=(001234,2),FROMDD=TAPESETS
         MOVE     DSNAME=UNLDSET3,TO=disk=222222,                       C
               FROM=2400-2=(001234,3),FROMDD=TAPESETS
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set.
- DD1 DD defines the system residence device.
- DD2 DD defines a mountable device on which the receiving volume is mounted.
- TAPESETS DD defines a mountable device on which the source volume is mounted. DCB information is provided in this statement.
- SYSIN DD defines the control data set, which follows in the input stream.
- MOVE moves the unloaded data sets to the receiving volume.

Note: To move a data set from a tape volume that contains more than one data set, you must specify the sequence number of the data set in the *list* field of the FROM parameter on the utility control statement.

IEHMOVE Example 9

In this example, two sequential data sets are to be copied from separate source volumes to a disk volume. Space is allocated by IEHMOVE. Only one 9-track tape unit is available for the operation.

```
                                                           72
//DEFER    JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//TAPE1    DD   VOLUME=SER=001234,UNIT=tape,DISP=OLD
//TAPE2    DD   VOLUME=SER=001235,UNIT=AFF=TAPE1,DISP=OLD
//SYSIN    DD   *
         COPY   DSNAME=SEQSET1,TO=disk=222222,              C
               FROM=2400=(001234,2),FROMDD=TAPE1
         COPY   DSNAME=SEQSET9,TO=disk=222222,              C
               FROM=2400=(001235,4),FROMDD=TAPE2
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the volume that is to contain the work data set.
- DD1 DD defines the system residence device.
- DD2 DD defines a mountable device on which the receiving volume is mounted.
- TAPE1 DD defines a mountable device on which the first volume to be processed is mounted. The source data set is the second data set on the volume.
- TAPE2 DD defines a mountable device on which the second volume to be processed is mounted when it is required. The source data set is the fourth data set on the volume.
- SYSIN DD defines the control data set, which follows in the input stream.
- COPY copies the data sets to the receiving volume.

Note: To copy a data set from a tape volume that contains more than one data set, you must specify the sequence number of the data set in the list field of the FROM parameter on the utility control statement.

IEHMOVE Example 10

In this example, three unloaded partitioned data sets residing on an unlabeled tape volume mounted on device 282 are copied to a 2314 volume mounted on device 191.

```
//LOAD      JOB  MEDDAUGH,PS40300439,MSGLEVEL=1
//          EXEC PGM=IEHMOVE
//SYSPRINT DD    SYSOUT=A
//SYSABEND DD    SYSOUT=A
//SYSUT1    DD   UNIT=191,VOLUME=SER=231400,DISP=OLD
//DD1       DD   UNIT=191,VOLUME=SER=231400,DISP=OLD
//TAPE1     DD   UNIT=282,VOLUME=SER=NLTAPE,DISP=OLD,
// LABEL=( ,NL),DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN     DD   *
   COPY PDS=DSET1,FROM=282=(NLTAPE,1),TO=191=231400,FROMDD=TAPE1
   COPY PDS=DSET2,FROM=282=(NLTAPE,2),TO=191=231400,FROMDD=TAPE1
   COPY PDS=DSET3,FROM=282=(NLTAPE,3),TO=191=231400,FROMDD=TAPE1
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the work data set.
- DD1 DD defines the receiving volume.
- TAPE1 DD defines the source data sets. They are, in the order in which they reside on the volume, DSET1, DSET2, and DSET3.
- SYSIN DD defines the control data set, which follows in the input stream.
- COPY copies the unloaded partitioned data sets from the unlabeled tape to the receiving volume.

Note: To copy data sets from an unlabeled tape, you must place a label in the *list* field of the FROM parameter of the utility control statement. Following this label, the sequence numbers of the data sets must also be included in the same field. The unit address must appear in the device field of the FROM or TO parameter whenever you want to move from or copy to a specific device.

IEHMOVE Example 11

In this example, the data set group A.B.C—which comprises data set A.B.C.X, A.B.C.Y, and A.B.C.Z—is moved from two disk volumes onto a third volume. Space is allocated by IEHMOVE. The catalog is updated to refer to the receiving volume. The source data sets are scratched.

```
//MOVEDSG   JOB  09#550,GREEN
//          EXEC PGM=IEHMOVE
//SYSPRINT DD    SYSOUT=A
//SYSUT1    DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD1       DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2       DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD3       DD   UNIT=disk,VOLUME=SER=333333,DISP=OLD
//DD4       DD   UNIT=disk,VOLUME=SER=444444,DISP=OLD
//SYSIN     DD   *
          MOVE   DSGROUP=A.B.C,TO=disk=222222
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set.
- DD1 DD defines the system residence device.
- DD2 DD defines a mountable device on which the receiving volume is to be mounted.

- DD3 DD defines a mountable device on which one of the source volumes is to be mounted.
- DD4 DD defines a mountable device on which one of the source volumes is to be mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- MOVE moves the specified data sets to volume 222222.

Note: This example can be used to produce the same result without the use of the DD4 DD statement, using one less mountable disk device. With DD3 and DD4, both of the source volumes are mounted at the start of the job. With DD3 only, the 333333 volume is mounted at the start of the job. After the 333333 volume is processed, the utility requests that the operator mount the 444444 volume. In this case the DD3 statement is coded:

```
//DD3 DD UNIT=(disk,,DEFER),DISP=OLD,VOLUME=(PRIVATE,,
//          SER=(333333))
```

IEHMOVE Example 12

In this example, the SYSCTLG data set is to be moved from another mountable disk volume to a mountable disk volume. Space is allocated by IEHMOVE. The source catalog is scratched from the first disk volume.

```
//MOVECAT1 JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE,POWER=3
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=333333,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//SYSIN    DD   *
         MOVE   CATALOG,TO=disk=222222,CVOL=disk=111111
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set.
- DD1 DD defines the mountable device on which the source volume is to be mounted.
- DD2 DD defines the mountable device on which the receiving volume is to be mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- MOVE specifies the move operation and defines the source and receiving volumes.

Note: See "PARM Information on the EXEC Statement" for a description of the POWER PARM.

IEHMOVE Example 13

In this example, the catalog entries for data set group A.B.C—which comprises the entries A.B.C.X, A.B.C.Y, and A.B.C.Z—is to be moved from a SYSCTLG data set to a mountable disk volume. If no catalog exists on the receiving disk volume, one is created; if a catalog does exist, the specified entries are merged into it. The last INDEX of all entries in the source SYSCTLG is scratched. The work data set is deleted when the job step is completed.

```
//MOVECAT2 JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//SYSIN    DD   *
         MOVE   CATALOG=A.B.C,TO=disk=222222,CVOL=disk=111111
/*
```

The control statements are discussed below:

- SYSUT1 DD defines the device that is to contain the work data set. (Because IEHMOVE deletes the work data set at the completion of the program, it can be contained on the receiving volume, provided there is space for it.)
- DD1 DD defines the system mountable device on which the source volume is to be mounted.
- DD2 DD defines the mountable device on which the receiving volume is to be mounted.
- SYSIN DD defines the control data set, which follows in the input stream.
- MOVE specifies a move operation for selected entries and defines the source and receiving volumes.

IEHPROGM PROGRAM

IEHPROGM is a system utility used to modify system control data and to maintain data sets at an organizational level. IEHPROGM should only be used by those programmers locally authorized to do so.

IEHPROGM can be used to:

- Scratch a data set or a member.
- Rename a data set or a member.
- Catalog or uncatalog a non-VSAM data set.
- Build or delete an index or alias in a CVOL (SYSCTLG data set).
- Connect or release two CVOLs.
- Build and maintain a generation index in a CVOL.
- Maintain data set passwords.

At the completion or termination of the program, the highest return code encountered within the program is passed to the calling program.

Scratching a Data Set or Member

IEHPROGM can be used to scratch the following from a direct access volume or volumes:

- Sequential, indexed sequential, partitioned, or direct data sets.
- Members of a partitioned data set.
- Password-protected data sets.
- Data sets named by the operating system.

A data set is considered scratched when its data set control block is removed from the volume table of contents (VTOC) of the volume on which it resides; its space is made available for re-allocation.

The space occupied by a data set residing on a device that operates in split-cylinder mode is not available for re-allocation until all data sets sharing the cylinder have been scratched.

A member is considered scratched when its name is removed from the directory of the partitioned data set in which it is contained. The space occupied by a scratched member is not available for re-allocation until the partitioned data set is scratched or compressed. (When scratching a member of a partitioned data set, all aliases of that member should also be removed from the directory.)

If RACF is active, ALTER authorization is required to scratch a RACF-defined data set, and UPDATE authorization is required to scratch a member of a partitioned data set.

Renaming a Data Set or Member

IEHPROGM can be used to rename a data set or member that resides on a direct access volume. In addition, the program can be used to change any member aliases.

If RACF is active, ALTER authorization is required to rename a data set. UPDATE authorization is required to rename a member of a partitioned data set.

Cataloging or Uncataloging a Data Set

IEHPROGM can be used to catalog or uncatalog a non-VSAM sequential, indexed sequential, partitioned, or direct data set. The program catalogs a data set by generating an entry, containing the data set name and associated volume information, in the index of the catalog. A valid TTR pointer is not placed in the DSCB until the first time the data set is referenced.

The catalog function is used to catalog a non-VSAM data set that was not cataloged when it was created.

IEHPROGM uncatalogs a non-VSAM data set by removing the data set name and associated volume information from the catalog.

Building or Deleting an Index in a CVOL

IEHPROGM can be used to build a new index in a CVOL or to delete an existing index. In building an index, the program automatically creates as many higher level indexes as are necessary to complete the specified structure.

IEHPROGM can be used to delete one or more indexes from an index structure; however, an index cannot be deleted if it contains any entries. That is, it cannot be deleted if it refers to a lower level index or if it is part of a structure indicating the fully qualified name of a cataloged data set.

Figure 19-1 shows an index structure before and after a build operation. The left portion of the figure shows two cataloged data sets, A.Y.YY and A.B.X.XX, before the build operation. The right-hand portion of the figure shows the index structure after the build operation, which was used to build index A.B.C.D.E. Note in the left portion of the figure that index levels C and D do not exist before the build operation. These levels are automatically created when the level E index is built.

When the level E index is subsequently deleted, the level C and D indexes are not automatically deleted by the program. To delete these index levels, delete: A.B.C.D.E, A.B.C.D, and A.B.C, in that order. The level B index cannot be deleted because data set A.B.X.XX and the X level index are dependent upon the level B index.

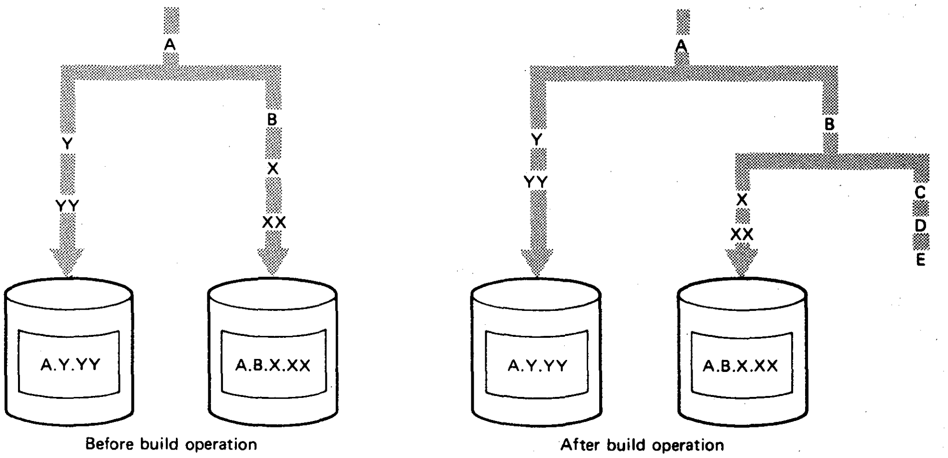


Figure 19-1. Index Structure Before and After an IEHPROGM Build Operation

Building or Deleting an Index Alias in a CVOL

IEHPROGM can be used to assign an alternative name (alias) to the highest level index of a CVOL or to delete a CVOL index alias previously assigned. An alias cannot, however, be assigned to the highest level of a generation index.

Figure 19-2 shows an alias, XX, that is assigned to index A (a high level index). The cataloged data set A.B.C can be referred to as either A.B.C or XX.B.C.

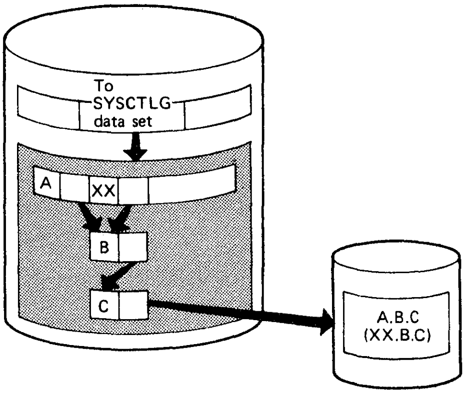


Figure 19-2. Building an Index Alias Using IEHPROGM

Connecting or Releasing Two Control Volumes (CVOLs)

IEHPROGM can be used to *connect* a volume to a second volume by placing an entry into a high level index on the first volume. The entry contains an index name and the volume serial number and device type of the second volume. The program can subsequently *release* the volumes by removing the entry from the high level index. If two volumes are connected:

- The catalog (SYSCTLG data set) must be created on the second volume for cataloging of data sets having the same high level index as the connected index.
- A high level index can only be connected to one second volume, but chaining is possible from a second to a third volume, etc.

Before any control volume (CVOL) can be accessed by the system, it must be defined in the VSAM master catalog. For details see *OS/VS2 Using OS Catalog Management with the Master Catalog: CVOL Processor.*

Figure 19-3 shows how one control volume can be connected to a second volume. Any subsequent index search for index X on the first volume is carried to the second volume.

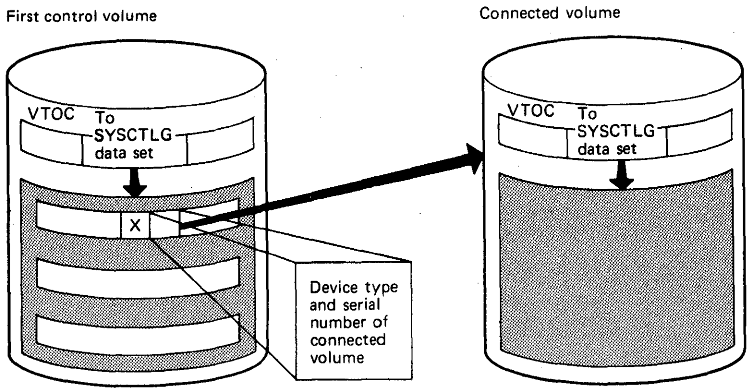


Figure 19-3. Connecting a Control Volume (CVOL) to a Second CVOL Using IEHPROGM

Note: The index name of each high level index existing on the second volume must be present in the first volume; when a new high level index is placed on a second volume, the first volume should be connected to the second volume.

Figure 19-4 shows three CVOLs connected to one CVOL. All volumes are accessible through high level indexes X, Y, and Z.

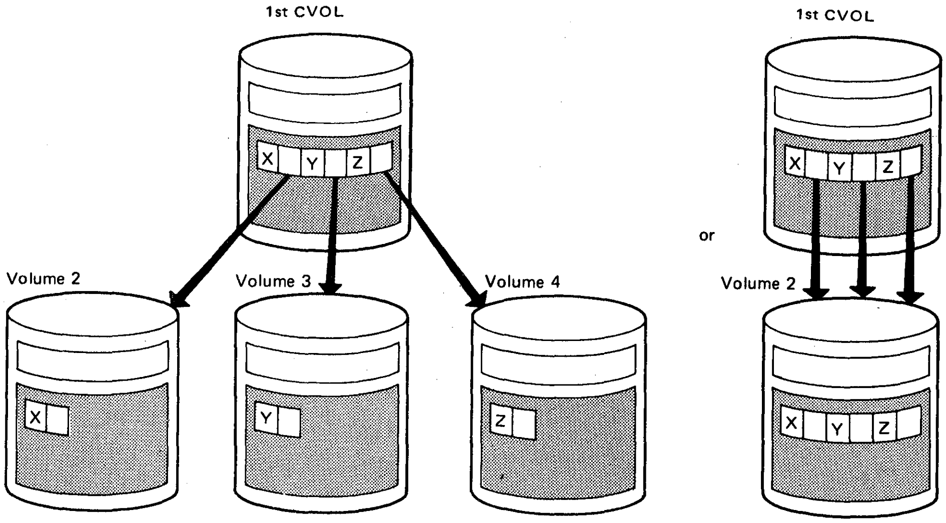


Figure 19-4. Connecting Three Volumes Using IEHPROGM

Building and Maintaining a Generation Index in a CVOL

IEHPROGM can be used to build an index structure in a CVOL for a generation data group and to define what action should be taken when the index overflows.

The lowest level index in the structure can contain up to 255 entries for successive generations of a data set. If the index overflows, the oldest entry is removed from the index, unless otherwise specified (in which case all entries are removed). If desired, the program can be used to scratch all generation data sets whose entries are removed from the index.

Figure 19-5 shows the index structure created for generation data group A.B.C. In this example, provision is made for up to five subsequent entries in the lowest level index.

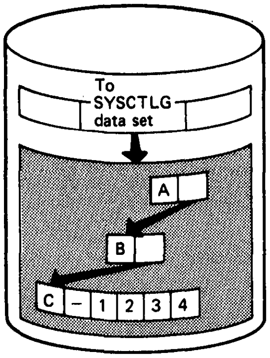


Figure 19-5. Building a Generation Index Using IEHPROGM

Note: Before a generation data group can be cataloged as such, a generation index must exist. Otherwise, a generation data set is cataloged as an individual data set, rather than as a generation.

When creating and cataloging a generation data set, the user can provide necessary DCB information. See *OS/VS2 MVS Data Management Services Guide* for a discussion of how DCB attributes are provided for a generation data group.

Maintaining Data Set Passwords

IEHPROGM can be used to maintain password entries in the PASSWORD data set and to alter the protection status of direct access data sets in the data set control block (DSCB). For a complete description of data set passwords and the PASSWORD data set, see *OS/VS2 System Programming Library: Data Management* and *OS/VS2 Data Management Services Guide.*

A data set can have one of three types of password protection, as indicated in the DSCB for direct access data sets and in the tape label for tape data sets (see *OS/VS2 Data Areas* for a description of the DSCB and tape label). The possible types of data set password protection are:

- No protection, which means that no passwords are required to read or write the data set.
- Read/write protection, which means that a password is required to read or write the data set.
- Read-without-password protection, which means that a password is required only to write the data set; the data set can be read without a password.

Note: If a system data set is password protected and a problem occurs on the data set, maintenance personnel must be provided with the password in order to access the data set and resolve the problem.

A data set can have one or more passwords assigned to it; each password has an entry in the PASSWORD data set. A password assigned to a data set can allow read *and* write access or only read access to the data set.

Figure 19-6 shows the relationship between the protection status of data set ABC and the type of access allowed by the passwords assigned to the data set. Passwords

ABLE and BAKER are assigned to data set ABC. If no password protection is set in the DSCB or tape label, data set ABC can be read or written without a password. If read/write protection is set in the DSCB or tape label, data set ABC can be read with either password ABLE or BAKER and can be written with password ABLE. If read-without-password protection is set in the DSCB or tape label, data set ABC can be read without a password and can be written with password ABLE; password BAKER is never needed.

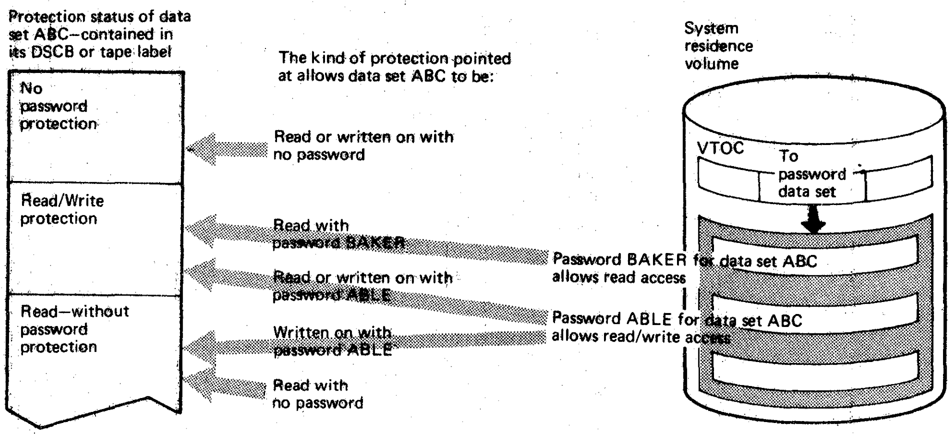


Figure 19-6. Relationship Between the Protection Status of a Data Set and Its Passwords

Before IEHPROGM is used to maintain data set passwords, the PASSWORD data set must reside on the system residence volume. IEHPROGM can then be used to:

- Add an entry to the PASSWORD data set.
- Replace an entry in the PASSWORD data set.
- Delete an entry from the PASSWORD data set.
- Provide a list of information from an entry in the PASSWORD data set.

Each entry in the PASSWORD data set contains the name of the protected data set, the password, the protection mode of the password, an access counter, and 77 bytes of optional user data. The protection mode of the password defines the type of access allowed by the password and whether the password is a control password or secondary password. The initial password, added to the PASSWORD data set for a particular data set, is marked in the entry as the control password for that data set. The second and subsequent passwords added for the same data set are marked as secondary passwords.

For direct access data sets, IEHPROGM updates the protection status in the DSCB when a control password entry is added, replaced, or deleted. This permits setting and resetting the protection status of an existing direct access data set at the same time its passwords are added, replaced, or deleted. IEHPROGM automatically alters the protection status of a data set in the DSCB if the following conditions are met:

- The control password for the data set is being added, replaced, or deleted.
- The data set is online.
- The volume on which the data set resides is specified on the utility control statement, or the data set is cataloged.
- The data set is not allocated within the IEHPROGM job.

For tape data sets, IEHPROGM cannot update the protection status in the tape label when a password entry is added, replaced, or deleted. Protection status in a tape label must be set with JCL.

Passwords to be added, replaced, deleted, or listed can be specified on utility control statements or can be entered by the console operator. IEHPROGM issues a message to the console operator when a password on a utility control statement is either missing or invalid. The message contains the job name, step name, and utility control statement name and identifies the particular password that is missing or invalid. Two invalid passwords are allowed per password entry on each utility control statement before the request is ignored; a total of five invalid passwords is allowed for the password entries on all the utility control statements in a job step before the step is canceled.

Note: If the current password is invalidly specified in the control statement, no message to the operator is issued and the request is ignored.

Adding Data Set Passwords

When a password is added for a data set, an entry is created in the PASSWORD data set with the specified data set name, password name, protection mode of the password (read/write or read only), and the optional 77 characters of user-supplied data. The access counter in the entry is set to zero.

The control password for a data set must always be specified to add, replace, or delete secondary passwords. The control password should not be specified, however, to list information from a secondary password entry.

Secondary passwords can be assigned to a data set to restrict some users to reading the data set or to record the number of times certain users access the data set. The access counter in each password entry provides a count of the number of times the password was used to successfully open the data set.

If a control password for a direct access, online data set is added, the protection status of the data set (read/write or read-without-password) is set in the DSCB. However, the data set to be protected must not be allocated within the same job as the one in which IEHPROGM is executed. If it is allocated, the DSCB cannot be accessed and the protection status is not set. If the data set to be protected is being created within the same job, use JCL to set the protection status in the DSCB.

Replacing Data Set Passwords

Any of the following information may be replaced in a password entry: the password, protection mode (read/write or read only) of the password, and the 77 characters of user data. The protection status of a data set can be changed by replacing the control entry for the data set.

If the control entry of a direct access, online data set is replaced, the DSCB is also reset to indicate any change in the protection status of the data set. Therefore, the user should ensure that the volume is online when changing the protection status of a direct access data set.

Deleting Data Set Passwords

When a control password entry is deleted from the PASSWORD data set, all secondary password entries for that data set are also deleted. However, when a secondary entry is deleted, no other password entries are deleted.

If the control password entry is deleted for an online, direct access data set, the protection status of the data set in the DSCB is also changed to indicate no protection. When deleting a control password for a direct access data set, the user should ensure that the volume is online. If the volume is not online, the password entry is removed, but data set protection is still indicated in the DSCB; the data set cannot be accessed unless another password is added for that data set.

If the control password entry is deleted for a tape data set, the user must change the protection status in the tape label to indicate no protection; otherwise, the tape volume cannot be accessed. The tape label may be changed using the IEHINITT utility program, however, the data set cannot be retrieved afterwards.

The delete function should be used to delete all the password entries for a scratched data set to make the space available for new entries.

Listing Password Entries

A list of information from any entry in the PASSWORD data set can be obtained in the SYSPRINT data set by providing the password for that entry. The list includes: the number of times the password has been used to successfully open the data set; the type of password (control password or secondary password) and type of access allowed by the password (read/write or read-only); and the user data in the entry. Figure 19-7 shows a sample list of information printed from a password entry.

```
DECIMAL ACCESS COUNT= 000025
PROTECT MODE BYTE= SECONDARY, READ ONLY
USER DATA FIELD= ASSIGNED TO J. BROWN
```

Figure 19-7. Listing of a Password Entry

Input and Output

IEHPROGM uses as input a control data set that contains utility control statements used to control the functions of the program and to indicate those data sets or volumes that are to be modified.

IEHPROGM produces as output a modified object data set or volume(s), and a message data set that contains error messages and information from the PASSWORD data set.

IEHPROGM provides a return code to indicate the results of program execution. The return codes and their meanings are:

- 00, which indicates successful completion.
- 04, which indicates that a syntax error was found in the name field of the control statement or in the PARM field in the EXEC statement. Processing is continued.
- 08, which indicates that a request for a specific operation was ignored because of an invalid control statement or an otherwise invalid request. The operation is not performed.
- 12, which indicates that an input/output error was detected when trying to read from or write to SYSPRINT, SYSIN or the VTOC.
- 16, which indicates an unrecoverable error. The job step is terminated.

Control

IEHPROGM is controlled by job control statements and utility control statements.

Job control statements are used to:

- Execute or invoke the program.
- Define the control data set.
- Define volumes and/or devices to be used during the course of program execution.
- Prevent data sets from being deleted inadvertently.
- Prevent volumes from being demounted before they have been completely processed by the program.
- Suppress listing of utility control statements.

Utility control statements are used to control the functions of the program and to define those data sets or volumes that are to be modified.

Job Control Statements

Figure 19-8 shows the job control statements necessary for using IEHPROGM.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IEHPROGM) or, if the job control statements reside in a procedure library, the procedure name. Additional PARM information can be specified to control the number of lines per page on the output listing and to suppress printing of utility control statements. See "PARM Information on the EXEC Statement" below.
SYSPRINT DD	Defines a sequential message data set.
anyname1 DD	Defines a permanently mounted volume. (The system residence volume is considered to be a permanently mounted volume.)
anyname2 DD	Defines a mountable device type.
SYSIN DD	Defines the control data set. The control data set normally follows the job control statements in the input stream; however, it can be defined as a member of a procedure library.

Figure 19-8. IEHPROGM Job Control Statements

The anyname1 DD statement can be entered:

```
//anyname1 DD UNIT=xxxx,VOLUME=SER=xxxxxx,DISP=OLD
```

The UNIT and VOLUME parameters define the device type and volume serial number. The DISP=OLD specification prevents the inadvertent deletion of a data set. The anyname1 DD statement is arbitrarily assigned the ddname DD1 in the IEHPROGM examples.

The anyname2 DD statement can be coded in the following ways:

```
//anyname2 DD  VOLUME=SER=xxxxxx,UNIT=xxxx,DISP=OLD

//anyname2 DD  VOLUME=(PRIVATE,SER=xxxxxx),
//             UNIT=(xxxx,,DEFER),DISP=OLD
```

The second example can be used to specify deferred mounting when a large number of magnetic tapes or direct access volumes are to be processed in one application of the program. The anyname2 DD statement is arbitrarily assigned the ddname DD2 in the IEHPROGM examples. DD statements defining additional mountable devices are assigned names DD3, DD4, etc.

Refer to "Appendix C: DD Statements for Defining Mountable Devices" for instructions on defining mountable volumes.

PARM Information on the EXEC Statement

Additional information can be specified in the PARM parameter of the EXEC statement to control the number of lines per page on the output listing and to suppress printing of utility control statements. The EXEC statement can be coded:

```
// EXEC PGM=IEHPROGM[,PARMLINECNT=xx,      {PRINT |
                                            NOPRINT}']
```

The LINECNT parameter specifies the number of lines per page in the listing of the SYSPRINT data set; xx is a 2-digit number, from 01 through 99. If LINECNT is omitted, or if an error is encountered in the LINECNT subparameter, the number of lines per page will be 45.

The PRINT value specifies that the utility control statements are to be written to the SYSPRINT data set. If neither PRINT nor NOPRINT is coded, PRINT is assumed.

The NOPRINT value specifies that utility control statements are not to be written to the SYSPRINT data set. Suppressing printing of utility control statements assures that passwords assigned to data sets remain confidential. However, suppressing printing may make it difficult to interpret error messages because the relevant utility control statement is not printed before the message.

Utility Control Statements

Figure 19-9 shows the utility control statements necessary for using IEHPROGM.

Statement	**Use**
SCRATCH	Scratches a data set or a member from a direct access volume.
RENAME	Changes the name or alias of a data set or member residing on a direct access volume.
CATLG	Generates an entry in the index of a catalog.
UNCATLG	Removes an entry from the lowest level index of the catalog.
BLDX	Creates a new index in the CVOL catalog (SYSCTLG data set).
DLTX	Removes a low level index from a CVOL.
BLDA	Assigns an alias to an index at the highest level of a CVOL.
DLTA	Deletes an alias previously assigned to an index at the highest level of a CVOL.
CONNECT	Connects two CVOLs together using a high level index name.
RELEASE	Removes a high level index name from one CVOL that served as a connector or pointer to a second CVOL.
BLDG	Builds an index in a CVOL for a generation data group and defines what action should be taken when the index overflows.
ADD	Adds a password entry in the PASSWORD data set.
REPLACE	Replaces information in a password entry.
DELETEP	Deletes an entry in the PASSWORD data set.
LIST	Formats and lists information from a password entry.

Figure 19-9. IEHPROGM Utility Control Statements

When a card is included for the sole purpose of continuing a comment, the continuation may start in any column between 1 and 71.

SCRATCH Statement

The SCRATCH statement is used to scratch a data set or member from a direct access volume. A data set or member is scratched only from the volumes designated in the SCRATCH statement. This function does not uncatalog scratched data sets.

When executing a SCRATCH operation, care should be taken to ensure that the data set or volume is not being used by a program executing concurrently.

The format of the SCRATCH statement is:

[*label*] **SCRATCH** {**DSNAME=***name* | **VTOC**}
,VOL=*device* =*list*
[**,PURGE**]
[**,MEMBER=***name*]
[**,SYS**]

RENAME Statement

The RENAME statement is used to change the true name or alias of a data set or member residing on a direct access volume. The name is changed only on the designated volume(s). The rename operation does not update the catalog.

The format of the RENAME statement is:

[*label*] **RENAME** **DSNAME=***name*
,VOL=*device* =*list*
,NEWNAME=*name*
[**,MEMBER=***name*]

CATLG Statement

The CATLG statement is used to generate a non-VSAM entry in the index of a CVOL. If additional levels of indexes are required in the CVOL, this function automatically creates them. When cataloging generation or VSAM data sets and the index becomes full, refer "BLDG (Build Generation Index) Statement" or *OS/VS2 Access Method Services,* respectively, for the action to be ta

When *device* is represented by a group name (for example, SYSDA) instead of a generic name (for example, 2314 or 2400) in the VOL parameter, the catalog operation does not enter the device type code in the system catalog. Instead, it places a unique entry in the device type field of the catalog. The allocation of the device for this entry may not be satisfactory to the user. The generic name should be used if the group name was generated for one or more device types. When the system is subsequently generated, this entry may no longer be valid; that is, all such group name entries should be uncataloged and then recataloged after a subsequent generation of the system.

When cataloging data sets residing on tape, specify the data set sequence number and the volume serial number, as follows:

VOL=device=(serial,seqno,...)

If a data set is created on a 9-track dual density tape unit (2400-4), the data set can be cataloged with a device specification of 2400 for an 800 bits per inch tape or 2400-3 for a 1600 bits per inch tape. If a device specification of 2400-4 is made when the data set is cataloged, any subsequent retrieval of that data set is made on a dual density unit.

If a data set is created on a 9-track dual density tape unit (3400-6), the data set can be cataloged with a device specification of 3400-3 for an 1600 bits per inch tape or 3400-5 for a 6250 bits per inch tape. If a device specification of 3400-6 is made when the data set is cataloged, any subsequent retrieval of that data set is made on a dual density unit.

The format of the CATLG statement is:

[*label*] **CATLG** **DSNAME**=*name*
,VOL=*device* =*list*
[**,CVOL**=*device* =*serial*]

UNCATLG Statement

The UNCATLG statement is used to remove a non-VSAM entry from the index of the catalog. If the entry removed was the last entry in the index, that index and all higher, unneeded, indexes, with the exception of the highest-level index, are removed from the catalog.

The format of the UNCATLG statement is:

[*label*] **UNCATLG** **DSNAME**=*name*
[**,CVOL**=*device*=*serial*]

BLDX (Build Index) Statement

The BLDX statement is used to create a new index in a CVOL catalog. If the creation of an index requires that higher level indexes be created, this function automatically creates them.

The format of the BLDX statement is:

[*label*] **BLDX** **INDEX**=*name*
[**,CVOL**=*device* =*serial*]

DLTX (Delete Index) Statement

The DLTX statement is used to remove an index from a CVOL catalog. Only an index that has no entries can be removed.

Because this function does not delete higher level indexes, it must be used repetitively to delete an entire structure. For example, to delete index structure A.B.C, delete index A.B.C, index A.B, and index A.

The format of the DLTX statement is:

[*label*] **DLTX** **INDEX**=*name*
[**,CVOL**=*device* =*serial*]

BLDA (Build Index Alias) Statement

The BLDA statement is used to assign an alias to an index at the highest level of a CVOL catalog.

The format of the BLDA statement is:

[*label*] **BLDA** **INDEX**=*name*
,ALIAS=*name*
[**,CVOL**=*device* =*serial*]

DLTA (Delete Index Alias) Statement

The DLTA statement is used to delete an alias previously assigned to an index at the highest level of a CVOL catalog.

The format of the DLTA statement is:

[*label*] **DLTA** **ALIAS=***name*
[,CVOL=*device* =*serial*]

CONNECT Statement

The CONNECT statement is used to place an entry in the high level index of a CVOL catalog. The entry identifies a second CVOL by its device type and volume serial number. In addition, it contains an index name identifying the index to be searched for (during subsequent index searches) on the second CVOL.

This function does not create an index on the second CVOL.

The CONNECT statement does not create a SYSCTLG data set on the connected volume. Before cataloging the first data set on a connected volume, the user must define a SYSCTLG data set on that volume. This can be done with the following DD statement:

```
//ddname DD DSNAME=SYSCTLG,UNIT=xxxx,DISP=(,KEEP),
//           SPACE=(CYL,1),VOLUME=SER=xxxxxx
```

If a job requires an auxiliary control volume to complete a catalog search, the user need not have the auxiliary control volume mounted before the job is begun. (The user does not have to remember the volume on which a particular data set is cataloged.) The system directs the operator to mount an auxiliary control volume if it is needed.

Before any CVOL can be accessed by the system, it must be defined in the VSAM master catalog. For details see *Using OS Catalog Management with the Master Catalog: CVOL Processor.*

The format of the CONNECT statement is:

[*label*] **CONNECT** **INDEX=***name*
,VOL=*device* =*serial*]
[,CVOL=*device* =*serial*]

RELEASE (Disconnect) Statement

The RELEASE statement is used to remove an entry from the high level index of a CVOL. This disconnects, in effect, a second CVOL from the first CVOL. The RELEASE statement does not delete an index from the second CVOL.

The format of the RELEASE statement is:

[*label*] **RELEASE** **INDEX=***name*
[,CVOL=*device* =*serial*]

BLDG (Build Generation Index) Statement

The BLDG statement is used to build an index for a generation data group, and to define what action should be taken when the index overflows.

The format of the BLDG statement is:

[*label*] **BLDG** **INDEX=***name*
,ENTRIES=*n*
[,CVOL=*device* **=***serial* **]**
[,EMPTY]
[,DELETE]

ADD (Add a Password) Statement

The ADD statement is used to add a password entry in the PASSWORD data set. When the control entry for a direct access, online data set is added, the indicated protection status of the data set is set in the DSCB; when a secondary entry is added, the protection status in the DSCB is not changed.

The format of the ADD statement is:

[*label*] **ADD** **DSNAME=***name*
[,PASWORD2=*new-password* **]**
[,CPASWORD=*control-password* **]**
[,TYPE=*code* **]**
[,VOL=*device* **=***list* **]**
[,DATA=' *user-data* **']**

REPLACE (Replace a Password) Statement

The REPLACE statement is used to replace any or all of the following information in a password entry: the password name, protection mode (read/write or read only) of the password, and user data. When the control entry for a direct access, online data set is replaced, the protection status of the data set is changed in the DSCB if necessary; when a secondary entry is replaced, the protection status in the DSCB is not changed.

The format of the REPLACE statement is:

[*label*] **REPLACE** **DSNAME=***name*
[,PASWORD1=*current-password* **]**
[,PASWORD2=*new-password* **]**
[,CPASWORD=*control-password* **]**
[,TYPE=*code* **]**
[,VOL=*device* **=***list* **]**
[,DATA=' *user-data* **']**

DELETEP (Delete a Password) Statement

The DELETEP statement is used to delete an entry in the PASSWORD data set. If a control entry is deleted, all the secondary entries for that data set are also deleted. If a secondary entry is deleted, only that entry is deleted. When the control entry for a direct access, online data set is deleted, the protection status in the DSCB is set to indicate that the data set is no longer protected.

The format of the DELETEP statement is:

[*label*] **DELETEP** **DSNAME=***name*
[,PASWORD1=*current-password***]**
[,CPASWORD=*control-password***]**
[,VOL=*device=list***]**

LIST (List Information from a Password) Statement

The LIST statement is used to format and print information from a password entry.

The format of the LIST statement is:

[*label*] **LIST** **DSNAME=***name*
,PASWORD1=*current-password*

Operands	**Applicable Control Statements**	**Description of Operands/Parameters**
ALIAS	BLDA DLTA	**ALIAS**=*name* specifies an unqualified name to be assigned as the alias or to be deleted from the index. The unqualified name must not exceed 8 characters.
CPASWORD	ADD	**CPASWORD**=*control-password* specifies the control password for the data set. CPASWORD must be specified unless this is the first password assigned to the data set, in which case PASWORD2 specifies the password to be added.
	DELETEP REPLACE	**CPASWORD**=*control-password* CPASWORD must be specified unless the control entry is being changed or deleted, in which case PASWORD1. specifies the control password.
CVOL	CATLG UNCATLG BLDX DLTX BLDA DLTA CONNECT RELEASE BLDG	**CVOL**=*device*=*serial*. For CONNECT and RELEASE, specifies the device type and volume serial number of the first CVOL. For CATLG, UNCATLG, BLDX, DLTX and BLDG, the CVOL on which the catalog search for the index (entry, for UNCATLG) is to begin. For BLDA and DLTA, the CVOL on which the catalog entry is to be made or deleted. For CONNECT and RELEASE, the first CVOL. If CVOL is omitted: For CATLG and UNCATLG, the catalog search begins with the VSAM master/JOBCAT/STEPCAT catalog. For BLDX, DLTX, BLDA, DLTA, CONNECT, RELEASE and BLDG, the system attempts to locate the proper (the first, for CONNECT) CVOL by checking the VSAM master catalog for a CVOL pointer alias name equal to the high level index specified in the INDEX (ATLAS, for DLTA) operand. The CVOL must be defined in the VSAM master catalog as: SYSCTLG.*Vserial*, where *serial* must equal the serial number of the CVOL. **Default:** The catalog search begins with the VSAM master catalog (or JOBCAT/STEPCAT, if specified).

Operands	Applicable Control Statements	Description of Operands/Parameters
DATA	ADD REPLACE	DATA=‘*user-data* ’ specifies the user data is to be placed in the password entry. The user data has a maximum length of 77 bytes and must be enclosed in apostrophes. If DATA is omitted from an ADD operation, 77 blanks are used. If DATA is omitted from a REPLACE operation, current user data is not changed.
DELETE	BLDG	**DELETE** specifies that generation data sets are to be scratched after their entries are removed from the index.
DSNAME	SCRATCH RENAME CATLG UNCATLG ADD REPLACE DELETEP LIST	DSNAME=*name* specifies the fully qualified name of either the data set to be scratched or renamed; the partitioned data set that contains the member to be scratched or renamed; the fully qualified name of the data set to be cataloged or uncataloged; or the fully qualified name of the data set whose password entry is to be changed, assigned, listed, or deleted. The qualified name must not exceed 44 characters, including delimiters.
EMPTY	BLDG	**EMPTY** specifies that all entries be removed from the generation index when it overflows. This uncatalogs, in effect, all of the generation data sets. **Default:** The entries with the largest generation numbers will be maintained in the catalog when the generation index overflows.
ENTRIES	BLDG	**ENTRIES=***n* specifies the number of entries to be contained in the generation index; *n* must not exceed 255.
INDEX	BLDG	**INDEX=***name* specifies the 1- to 35-character qualified name of the generation index.
	BLDX DLTX	**INDEX=***name* specifies the qualified name of the index to be created or deleted. The qualified name must not exceed 44 characters, including delimiters.
	BLDA	**INDEX=***name* specifies the unqualified name of the index to which an alias name is to be assigned. The unqualified name must not exceed 8 characters.
	CONNECT RELEASE	**INDEX=***name* specifies the unqualified index name to be entered or removed from the high level index on the first CVOL. The unqualified name must not exceed 8 characters.

Operands	Applicable Control Statements	Description of Operands/Parameters
MEMBER	SCRATCH RENAME	**MEMBER=***name* specifies a member name or alias of a member (in the named data set) to be renamed or removed from the directory of a partitioned data set. This name is not validity-checked because all members must be accessible, whether the name is valid or not. **Default:** The specified data set name or volume of data sets is changed or scratched.
NEWNAME	RENAME	**NEWNAME=***name* specifies the new fully qualified name for the data set, or the new member or alias.
PASWORD1	REPLACE DELETEP LIST	**PASWORD1=***current-password* specifies the password in the entry to be listed, changed, or deleted. **Default:** The operator is prompted for the current password.
PASWORD2	ADD REPLACE	**PASWORD2=***new-password* specifies the new password to be added or assigned to the entry. If the password is not to be changed, the current password must also be specified as the new password. The password can consist of one- to eight-alphameric characters. **Default:** The operator is prompted for a new password.
PURGE	SCRATCH	**PURGE** specifies that each data set specified by DSNAME or VTOC be scratched, even if its expiration date has not elapsed. **Default:** The specified data sets are scratched only if their expiration dates have elapsed.
SYS	SCRATCH	**SYS** specifies that data sets that have names that begin with "AAAAAAAA.AAAAAAAA.AAAAAAAA.AAAAAAAA." or "SYSnnnnn.T" and "F" or "V" in position 19 are to be scratched. These are names assigned to data sets by the operating system. This parameter is valid only when VTOC is specified. If the name of the data set to be scratched begins with SYS, nnnnn is the date.

Operands	Applicable Control Statements	Description of Operands/Parameters
TYPE	ADD REPLACE	**TYPE**=*code* specifies the protection code of the password and, if a control password entry is to be changed for or assigned to a direct access, online data set, specifies the protection status of the data set. The values that can be specified for *code* are: **1** specifies that the password is to allow both read and write access to the data set; if a control password is being assigned or changed, read/write protection is set in the DSCB. **2** specifies that the password is to allow only read access to the data set; if control password is being assigned or changed, read/write protection is set in the DSCB. **3** specifies that the password is to allow both read and write access to the data set; if a control password is being assigned or changed, read-without-password protection is set in the DSCB. **Default:** For ADD, if this parameter is omitted, the new password is assigned the same protection code as the control password for the data set. If a control password is being "added," TYPE=3 is the default. For REPLACE, the protection is not changed.
VOL	CONNECT	**VOL**=*device=serial* specifies the device type and serial number of the second CVOL. This information is placed in the high level index of the first CVOL.

Operands	Applicable Control Statements	Description of Operands/Parameters
VOL (continued)	ADD REPLACE DELETEP SCRATCH RENAME CATLG	VOL=*device*=*list* specifies the device type and serial number(s) of the volume(s), limited to 50, that contain the data set(s). For SCRATCH and RENAME, if VTOC or MEMBER is specified, VOL cannot specify more than one volume. Caution should be used when specifying VTOC if VOL specifies the system residence volume. For CATLG, the volume serial numbers must appear in the same order in which they were originally encountered (in DD statements within the input stream) when the data set was created. For ADD, REPLACE and DELETEP, if omitted, the protection status in the DSCB is not set or changed, unless the data set is cataloged. This parameter is not necessary for secondary password entries, or if the desired protection status in the DSCB is already set or is not to be changed by ADD or REPLACE.
VTOC	SCRATCH	**VTOC** specifies that all data sets on the specified volume, except those protected by a password or those whose expiration dates have not expired, are to be scratched. Password-protected data sets are scratched if the correct password is provided. The effect of VTOC is modified when it is used with PURGE or SYS.

Restrictions

- The block size for the SYSPRINT (message) data set must be a multiple of 121. The block size for the SYSIN (control) data set must be a multiple of 80. Any blocking factor can be specified for these block sizes.
- With the exception of the SYSIN and SYSPRINT DD statements, all DD statements in Figure 19-8 are used as device allocation statements, rather than as true data definition statements. Because IEHPROGM modifies the internal control blocks created by device allocation DD statements, the DSNAME parameter, if supplied, will be ignored by IEHPROGM. (All data sets are defined explicitly or implicitly by utility control statements.)
- One anyname1 DD statement must be included for each permanently mounted volume referred to in the job step.
- One anyname2 DD statement must be included for each mountable device to be used in the job step.
- When IEHPROGM is dynamically invoked in a job step containing a program other than IEHPROGM, the DD statements defining mountable devices for IEHPROGM must be included in the job stream prior to DD statements defining data sets required by the other program.
- Unpredictable results may occur in multi-tasking environments where dynamic allocation/deallocation of devices, by other tasks, causes changes in the TIOT during IEHPROGM execution.

IEHPROGM Examples

The following examples illustrate some of the uses of IEHPROGM. Figure 19-10 can be used as a quick reference guide to IEHPROGM examples. The numbers in the "Example" column point to the examples that follow.

Operation	Mount Volumes	Comments	Example
SCRATCH	Disk	VTOC is to be scratched.	1
SCRATCH UNCATLG	Disk	Two data sets are to be scratched and uncataloged.	2
RENAME, UNCATLG CATLG	Disks	A data set is to be renamed on two mountable devices; the old data set name is to be removed from the catalog. The data set is cataloged under its new name. Object data set resides on two mountable devices.	3
UNCATLG	Disk	Three generation data sets are to be uncataloged, their index structures deleted from the catalog.	4
RENAME, DELETEP, and ADD	Disk	The object data set exists on one mountable device.	5
LIST and REPLACE	Disk	The object data set exists on two mountable devices.	6
RENAME	Disk	Rename a member of a partitioned data set.	7
CATLG and CONNECT	Disk	Connect one CVOL to another.	8
BLDG, RENAME and CATLG	Disk	A generation index is built, three data sets are renamed and cataloged into the generation index.	9
BLDG	Disk	A new generation index is built and updated through JCL.	10,11

Figure 19-10. IEHPROGM Example Directory

Note: Examples which use *disk* or *tape*, in place of actual device-ids, must be changed before use. See the Device Support section, in the Introduction to this manual, for valid device-id notation.

In the IEHPROGM examples, the EXEC statement and the SYSPRINT DD statement can be replaced with the following job control statement:

```
//         EXEC PROC=MOD
```

which invokes the following IBM-supplied cataloged procedure:

```
//MOD EXEC      PGM=IEHPROGM,REGION=44K
//DDSRV DD      VOLUME=REF=SYS1.SVCLIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
```

IEHPROGM Example 1

In the following example, data sets are to be scratched from the volume table of contents of a mountable volume. Because the system residence volume is not referred to, no DD1 DD statement is necessary in the job stream.

```
//SCRVTOC  JOB  09#550,BROWN
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD2      DD   UNIT=disk,VOLUME=SER=222222,DISP=OLD
//SYSIN    DD   *
      SCRATCH   VTOC,VOL=disk=222222
/*
```

The SCRATCH statement, used in this example, indicates that all data sets (including those system data sets beginning with AAAAAA.AAAAAA.AAAAAA.AAAAAA) whose expiration dates have expired are to be scratched from the specified volume.

IEHPROGM Example 2

In this example, two data sets are to be scratched: SET1 is to be scratched on volume 222222, and A.B.C.D.E is to be scratched on volume 222222. Both data sets are to be uncataloged.

```
//SCRDSETS JOB  09#550,BROWN
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=disk,DISP=OLD,VOLUME=SER=222222
//SYSIN    DD   *
      SCRATCH   DSNAME=SET1,VOL=disk=222222
      UNCATLG   DSNAME=SET1
      SCRATCH   DSNAME=A.B.C.D.E,VOL=disk=222222
      UNCATLG   DSNAME=A.B.C.D.E
/*
```

The control statements are discussed below:

- The first SCRATCH statement specifies that SET1, which resides on volume 222222, is to be scratched.
- The first UNCATLG statement specifies that SET1 is to be uncataloged.
- The second SCRATCH statement specifies that A.B.C.D.E, which resides on volume 222222, is to be scratched.
- The second UNCATLG statement specifies that A.B.C.D.E is to be uncataloged.

IEHPROGM Example 3

In this example, the name of a data set is to be changed on two mountable volumes. The old data set name is to be removed from the catalog and the data set is to be cataloged under its new data set name.

```
                                                            72
//RENAMEDS JOB  09#550,BROWN
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1      DD   VOLUME=SER=111111,UNIT=disk,DISP=OLD
//DD2      DD   UNIT=(disk,,DEFER),DISP=OLD,
// VOLUME=(PRIVATE,SER=(222222,333333))
//SYSIN    DD   *
   RENAME  DSNAME=A.B.C,NEWNAME=NEWSET,                     C
               VOL=disk=(222222,333333)
  UNCATLG  DSNAME=A.B.C
    CATLG  DSNAME=NEWSET,VOL=disk=(222222,333333)
/*
```

The control statements are discussed below:

- RENAME specifies that data set A.B.C, which resides on volumes 222222 and 333333, is to be renamed NEWSET.
- UNCATLG specifies that data set A.B.C is to be uncataloged.
- CATLG specifies that NEWSET, which resides on volumes 222222 and 333333, is to be cataloged.

IEHPROGM Example 4

In this example, three data sets—A.B.C.D.E.F.SET1, A.B.C.G.H.SET2, and A.B.I.J.K.SET3—are to be uncataloged. The system residence volume resides on a disk volume.

```
//DLTSTRUC JOB  09#550,BROWN
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//SYSIN    DD   *
      UNCATLG   DSNAME=A.B.C.D.E.F.SET1
      UNCATLG   DSNAME=A.B.C.G.H.SET2
      UNCATLG   DSNAME=A.B.I.J.K.SET3
/*
```

The control statements are discussed below:

- The UNCATLG statements specify that data sets A.B.C.D.E.F.SET1, A.B.C.G.H.SET2, and A.B.I.J.K.SET3 are to be uncataloged.

IEHPROGM Example 5

In this example, a data set is to be renamed. The data set passwords assigned to the old data set name are to be deleted. Then two passwords are to be assigned to the new data set name.

Note: If the data set is not cataloged, a message indicating that the LOCATE macro instruction failed is issued. The return code is 8.

```
                                                                      72
//ADDPASS  JOB   09#550,BROWN
           EXEC  PGM=IEHPROGM,PARM='NOPRINT'
//SYSPRINT DD    SYSOUT=A
//DD1      DD    VOLUME=(PRIVATE,SER=222222),DISP=OLD,
// UNIT=(disk,,DEFER)
//SYSIN    DD    *
  RENAME DSNAME=OLD,VOL=disk=222222,NEWNAME=NEW
 DELETEP DSNAME=OLD,PASWORD1=KEY
     ADD DSNAME=NEW,PASWORD2=KEY,TYPE=1,                               C
               DATA='SECONDARY IS READ'
     ADD DSNAME=NEW,PASWORD2=READ,CPASWORD=KEY,TYPE=2,                 C
               DATA='ASSIGNED TO J. DOE'
/*
```

The control statements are discussed below:

- DELETEP specifies that the entry for the password KEY is to be deleted. Because KEY is a control password in this example, all the password entries for the data set name are deleted. The VOL parameter is not needed because the protection status of the data set as set in the DSCB is not to be changed; read/write protection is presently set in the DSCB, and read/write protection is desired when the passwords are reassigned under the new data set name.
- The ADD statements specify that entries are to be added for passwords KEY and READ. KEY becomes the control password and allows both read and write access to the data set. READ becomes a secondary password and allows only read access to the data set. The VOL parameter is not needed, because the protection status of the data set is still set in the DSCB.

Note: The operator is required to supply a password to rename the old data set.

IEHPROGM Example 6

In this example, information from a password entry is to be listed. Then the protection mode of the password, the protection status of the data set, and the user data are to be changed.

```
                                                                      72
//REPLPASS JOB   09#550,BROWN
           EXEC  PGM=IEHPROGM,PARM='NOPRINT'
//SYSPRINT DD    SYSOUT=A
//DD1      DD    UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD    VOLUME=(PRIVATE,SER=(222222,333333)),
// UNIT=(disk,,DEFER),DISP=OLD
//SYSIN    DD    *
     LIST   DSNAME=A.B.C,PASWORD1=ABLE
  REPLACE   DSNAME=A.B.C,PASWORD1=ABLE,                                C
               PASWORD2=ABLE,TYPE=3,                                   C
               VOL=disk=(222222,333333),                               C
               DATA='NO SECONDARIES; ASSIGNED TO DEPT 31'
/*
```

The control statements are discussed below:

- LIST specifies that the access counter, protection mode, and user data from the entry for password ABLE are to be listed. Listing the entry permits the content of the access counter to be recorded before the counter is reset to zero by the REPLACE statement.
- REPLACE specifies that the protection mode of password ABLE is to be changed to allow both read and write access and that the protection status of the data set is to be changed to write-only protection. The VOL parameter is required because the protection status of the data set is to be changed and the data set, in this example, is not cataloged. Because this is a control password, the CPASWORD parameter is not required.

IEHPROGM Example 7

In this example, a member of a partitioned data set is to be renamed.

```
                                                            72
//REN       JOB  09#550,BROWN
//          EXEC PGM=IEHPROGM
//SYSPRINT  DD   SYSOUT=A
//DD1       DD   VOL=SER=222222,DISP=OLD,UNIT=disk
//SYSIN     DD *
 RENAME     VOL=disk=222222,DSNAME=DATASET,NEWNAME=BC,         C
            MEMBER=ABC
/*
```

The control statements are discussed below:

- DD1 DD defines a permanently mounted volume.
- SYSIN DD defines the input data set, which immediately follows in the input stream.
- RENAME specifies that member ABC in the partitioned data set DATASET, which resides on a disk volume, is to be renamed BC.

IEHPROGM Example 8

In this example, a new CVOL catalog (SYSCTLG data set) is defined and connected to an existing CVOL. A data set is then cataloged in the new CVOL.

```
//LNKX      JOB
//STEP1     EXEC PGM=IEHPROGM
//SYSPRINT  DD SYSOUT=A
//NEWCVOL   DD DSN=SYSCTLG,UNIT=disk,VOL=SER=222222,
//  DISP=(,KEEP),SPACE=(TRK,(10,1))
//FIRST     DD UNIT=disk,VOL=SER=111111,DISP=SHR
//SYSIN     DD *
   CATLG DSNAME=SYSCTLG.V222222,VOL=disk=222222
   CONNECT INDEX=AA,VOL=disk=222222
   CATLG DSNAME=AA.BB,VOL=disk=PACK14
/*
```

This example assumes that the CVOL on volume 111111 was previously defined in the VSAM master catalog with a CVOL pointer, and "AA" was defined in the VSAM master catalog as an alias of the CVOL pointer. See *Using OS Catalog Management with the Master Catalog: CVOL Processor* for details on how this is done.

The control statements are discussed below:

- NEWCVOL DD allocates space for the new CVOL.
- The first CATLG statement establishes a CVOL pointer in the VSAM master catalog for the new CVOL.
- The CONNECT statement causes the new CVOL (on volume 222222) to be connected to the old CVOL (on volume 111111), such that any catalog management requests coming to the old CVOL having a high level index name of AA will be routed to the new CVOL.
- The second CATLG statement will cause the data set AA.BB to be cataloged in the new CVOL on volume 222222. Since this is the first request to update the new CVOL, this will cause the new CVOL to be formatted before the catalog entry is made.

IEHPROGM Example 9

In this example, a generation index for generation data group A.B.C is built in a CVOL. Three existing non-cataloged, non-generation data sets are renamed; the renamed data sets are cataloged as generations in the generation index.

```
                                                                      72
//BLDINDEX JOB
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=disk,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=(disk,,DEFER),DISP=OLD,
// VOLUME=(PRIVATE,,SER=(222222))
//SYSIN    DD   *
  BLDG     INDEX=A.B.C,ENTRIES=10,CVOL=disk=111111
  RENAME   DSNAME=DATASET1,VOL=disk=222222,                            C
               NEWNAME=A.B.C.G0001V00
  RENAME   DSNAME=DATASET2,VOL=disk=222222,                            C
               NEWNAME=A.B.C.G0002V00
  RENAME   DSNAME=DATASET3,VOL=disk=222222,                            C
               NEWNAME=A.B.C.G0003V00
  CATLG  DSNAME=A.B.C.G0001V00,VOL=disk=222222,CVOL=disk=111111
  CATLG  DSNAME=A.B.C.G0002V00,VOL=disk=222222,CVOL=disk=111111
  CATLG  DSNAME=A.B.C.G0003V00,VOL=disk-222222,CVOL=disk=111111
/*
```

The control statements are discussed below:

- DD1 DD defines the volume on which the SYSCTLG data set resides.
- BLDG specifies the generation group name A.B.C and makes provision for ten entries in the index. The oldest generation is to be uncataloged when the index becomes full. No generations are to be scratched.
- The RENAME statements rename three non-generation data sets residing on a disk volume.
- CATLG catalogs the renamed data sets in the generation index.

Note: Because the DCB parameters were supplied when the non-generation data sets were created, no DCB parameters are now specified; therefore, no model DSCB is required.

IEHPROGM Example 10

In this example, an IEHPROGM job step, STEPA, creates a model DSCB and builds a generation index. STEP B, an IEBGENER job step, creates and catalogs a sequential generation from card input.

This example assumes that the CVOL with serial number 111111 was previously defined in the VSAM master catalog with a CVOL pointer, and "A" was defined in the VSAM master catalog as an alias of the CVOL pointer. See *Using OS Catalog Management with the Master Catalog: CVOL Processor* for details on how this is done.

```
//BLDINDX  JOB
//STEPA    EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//BLDDSCB  DD   DSNAME=A.B.C,DISP=(,KEEP),SPACE=(TRK,(0)),
// DCB=(LRECL=80,RECFM=FB,BLKSIZE=800),
// VOLUME=SER=111111,UNIT=disk
//SYSIN    DD   *
           BLDG INDEX=A.B.C,ENTRIES=10,EMPTY,DELETE
/*
//STEPB    EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DUMMY
//SYSUT2   DD   DSNAME=A.B.C(+1),UNIT=disk,DISP=(,CATLG),
// VOLUME=SER=222222,SPACE=(TRK,20)
//SYSUT1   DD   DATA
```

(input card data)

```
/*
```

The control statements are discussed below:

- BLDDSCB DD creates a model DSCB on the CVOL volume.
- SYSIN DD indicates that a utility control statement (BLDG) is included next in the input stream.
- BLDG specifies the generation data group name A.B.C and makes provision for ten entries in the group. When the index is filled, it is to be emptied, and all of the generations are to be deleted.
- SYSUT2 DD defines an output sequential generation. The generation is assigned the absolute generation and version number G0001V00 in the index.
- SYSUT1 DD defines the input card data set.

Any subsequent job that causes the deletion of the generations should include DD statements defining the devices on which the volumes containing those generations are to be mounted. Each generation for which no DD statement is included is uncataloged at that time, but not deleted.

After the generation data group is emptied, the new generations continue to be assigned generation numbers according to the last generation number assigned before the empty operation. To reset the numbering operation (that is, to reset to G0000V00 or G0001V00), it is necessary to uncatalog all the old generation data sets and then rename and recatalog, beginning with G0000V00.

IEHPROGM Example 11

In this part of the example, a second generation is created and cataloged in the index built in Example 10. DCB attributes are included to override those attributes that were specified when the model DSCB was created.

```
//          JOB
//          EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DUMMY
//SYSUT2   DD   DSNAME=A.B.C(+1),UNIT=disk,DISP=(,CATLG),
// DCB=(LRECL=80,RECFM=FB,BLKSIZE=1600),
// VOLUME=SER=222222,SPACE=(TRK,20)
//SYSUT1   DD   DATA
```

(input cards)

```
/*
```

The control statements are discussed below:

- SYSUT2 DD defines an output sequential generation. The generation is assigned the absolute generation and version number G0002V00 in the index. The specified DCB attributes override those initially specified in the model DSCB. The DCB attributes specified when the model DSCB was created remain unchanged; that is, those attributes are applicable when you catalog a succeeding generation unless you specify overriding attributes at that time.
- SYSUT1 defines the input card data set.

IFHSTATR PROGRAM

IFHSTATR is a system utility used to format and print information from type 21 (error statistics by volume) records.

Figure 20-1 shows the format of the type 21 record.

4	Bytes of Record Descriptor Word			
0	System Indicator	Record Type	Time of Day	
4	Time of Day (continued)		Current Date	
8	Current Date (continued)		System Identification	
12	System identifier		Length of rest of record including this field	
16	Volume Serial Number			
20	Volume Serial No. (cont.)		22 Channel Unit Address	
24	UCB Type			
28	Temporary Read Errors	Temporary Write Errors	Start I/O's	
32	Permanent Read Errors	Permanent Write Errors	Noise Blocks	Erase Gaps
36	Erase Gaps (continued)	Cleaner Actions		Tape Density
40	Block Size		Reserved	

Figure 20-1. Type 21 (ESV) Record Format

Error statistics by volume (ESV) records should be retrieved from the IFASMPDP tape or from SYS1.MAN (on tape). ESV can also be retrieved directly from SYS1.MANX or SYS1.MANY (on a direct access storage device); however, IFHSTATR does not clear the SYS1.MANX (or SYS1.MANY) data set or make it available for additional records.

Assessing the Quality of a Tape Library

The statistics gathered by SMF in Type 21 records can be very useful in assessing the quality of a tape library. IFHSTATR prints Type 21 records in the same order that they were gathered, that is, date/time sequence. You may find it useful to sort Type 21 records into volume serial number sequence, into channel unit sequence, and into error occurrence sequence to aid in analyzing the condition of the library.

The IFHSTATR report helps to identify deteriorating media (tapes); occasionally poor performance from a particular tape drive can also be identified. The permanent read error counter or permanent write error counter is incremented by one each time the Tape Error Recovery routines (ERPs) determine that the error is permanent and is returned to the user with indication of a permanent I/O error. If a SYNAD routine to handle such errors is present, the counts in these fields can be greater than one. The temporary read error counter and temporary write error counter are incremented when the ERP initially handles an error condition which is corrected in the ERP. The severity of a temporary error can be estimated by analyzing either the erase gap counter for write errors or the noise block and cleaner action counters for read errors. The erase gap counter is incremented each time a write error is retried. For example, if the temporary write error counter contains 2 and the erase gap counter contains 5, the ERP was entered twice for write error recovery. The average recovery actions were 2.5 per error (actually may have been 1 and 4). The cleaner action counter is only incremented every fourth read retry. A ratio of one cleaner action to one temporary read error indicates, in general, recovery on the fifth retry (the first retry after the cleaner action). A ratio of ten cleaner actions to one temporary error indicates that recovery is, in general, a result of reading the tape in the opposite direction (reading backward on a read forward tape or reading forward on a read backward tape). The noise block counter is incremented once for each noise record (record less than minimum read length) encountered.

In analyzing IFHSTATR reports, the usage (SIO) count should also be considered, because it is the count of all Start I/O's to the tape drive, except those issued by the ERP in the course of error recovery. The usage count can be used to determine the ratio of error free accesses of the tape to total accesses of the tape.

Input and Output

IFHSTATR uses as input type 21 records, which contain information about errors on magnetic tape. IFHSTATR processes only type 21 records; if none are found, a message is written to the output data set.

IFHSTATR produces as output an output data set, which contains information selected from type 21 records. The output takes the form of 121-byte unblocked records, with an ASA control character in the first byte of each record.

Figure 20-2 shows a sample of printed output from IFHSTATR.

```
VOLUME          CPU  MOD    TIME    CHANNEL  TEMP  TEMP    PERM  PERM    NOISE     ERASE  CLEANER  USAGE   TAPE      BLOCK
SERIAL    DATE   ID   NO   OF DAY   / UNIT   READ  WRITE   READ  WRITE   BLOCKS     GAPS  ACTIONS (SIO's) DENSITY   LENGTH

001021  69/309  BB   40  15:55:07    181       1     0       0     0       1         0       0      10     0800        80
001022  69/309  AA   40  15:56:02    184      10     0       0     0       0         0       0      28     1600       121
000595  69/309  CC   50  15:56:20    283       0    10       0     0       0        10       0      28     0800        50
```

Figure 20-2. Sample Output from IFHSTATR

Control

IFHSTATR is controlled by job control statements. Utility control statements are not used.

Job Control Statements

Figure 20-3 shows the job control statements necessary for using IFHSTATR.

The output data set can reside on any output device supported by BSAM.

Note: The LRECL and BLKSIZE parameters are not specified by IFHSTATR. This information is taken from the DCB parameter on the SYSUT1 DD statement or from the tape label.

Statement	Use
JOB	Initiates the job.
EXEC	Specifies the program name (PGM=IFHSTATR).
SYSUT1 DD	Defines the input data set and the device on which it resides. The DSNAME, UNIT, VOLUME, LABEL, DCB, and DISP parameters should be included.
SYSUT2 DD	Defines the sequential data set on which the output is to be written.

Figure 20-3. IFHSTATR Job Control Statements

IFHSTATR Example

This example shows the JCL needed to produce a report.

```
//          JOB
//          EXEC PGM=IFHSTATR
//SYSUT1    DD   UNIT=2400,DSNAME=SYS1.MAN,LABEL=(,SL),
// VOLUME=SER=VOLID,DISP=OLD
//SYSUT2    DD   SYSOUT=A
/*
```

APPENDIX A. EXIT ROUTINE LINKAGE

Utility programs can be linked to user-supplied exit routines for additional processing.

Linking to an Exit Routine

Linking to an exit routine from a utility program is accomplished in one of the following ways:

- If the exit routine is for label processing or totaling, or if the exit routine is specified in the IEBTCRIN program by OUTREC or ERROR, linkage is performed by the BALR instruction.
- In all other cases, linkage is performed by using the LINK macro instruction.

The LINK macro instruction contains the symbolic name of the entry point of an exit routine and, if required, a list of parameters.

For further information on the use of the LINK macro instruction, see *OS/VS2 Supervisor Services and Macro Instructions* and *OS/VS2 Data Management Macro Instructions.*

At the time of the linkage operation:

- General register 1 contains the starting address of the parameter list, or contains zero to indicate end-of-file on the input data set for the IEBTCRIN OUTREC or ERROR exits.
- General register 13 contains the address of the register save area. This save area must not be used by user label processing routines. See "Appendix D: Processing User Labels."
- General register 14 contains the address of the return point in the utility program.
- General register 15 contains the address of the entry point to the exit routine.

Registers 1 through 14 must be restored before control is returned to the utility program.

The exit routine must be contained in either the job library or the link library.

The parameter lists passed to label processing routines and parameter lists passed to nonlabel processing routines are described in the topics that follow.

Label Processing Routine Parameters

The parameters passed to a user's label processing routine are addresses of the 80-byte label buffer, the DCB being processed, the status information if an uncorrectable input/output error occurs, and the totaling area.

The 80-byte label buffer contains an image of the user label when an input label is being processed. When an output label is being processed, the buffer contains no significant information at entry to the user's label processing routine. When the utility program has been requested to generate labels, the label processing routine constructs a label in the label buffer.

If standard user labels (SUL) are specified on the DD statement for a data set, but the data set has no user labels, the system still takes the specified exits to the appropriate user's routine. In such a case, the user's input label processing routine is entered with the buffer address parameter set to zero.

The format and content of the DCB are presented in *OS/VS2 MVS Data Management Macro Instructions.*

Bit 0 of flag 1 in the DCB-address parameter is set to a value of 0 except when:

- Volume trailer or header labels are being processed at volume switch time.
- The trailer labels of a MOD data set are being processed (when the data set is opened).

If an uncorrectable input/output error occurs while reading or writing a user label, the appropriate label processing routine is entered with bit 0 of flag 2 in the status information address parameter set on. The three low order bytes of this parameter contain the address of standard status information as supplied for SYNAD routines. (The SYNAD routine is not entered.)

Nonlabel Processing Routine Parameters

Figure 21-1 shows the program from which exits can be taken to nonlabel processing routines, the names of the exits, and the parameters available for each exit routine.

Program	**Exit**	**Parameters**
IEBGENER	KEY	Address at which key is to be placed (record follows key); address of DCB.
	DATA	Address of SYSUT1 record; address of DCB.
	IOERROR	Address of DECB; cause of the error and address of DCB. (Address in lower order three bytes and cause of error in high order byte.)
IEBCOMPR	ERROR	Address of DCB for SYSUT1; address of DCB for SYSUT2.[1]
	PRECOMP	Address of SYSUT1 record; length of SYSUT1 record, address of SYSUT2 record; length of SYSUT2 record.
IEBPTPCH	INREC	Address of input record; length of the input record.
	OUTREC	Address of output record; length of the output record.
IEBTCRIN	ERROR	Address of the error record; address of a full word which contains the record length.
	OUTREC	Address of the normal record; address of a full word which contains the record length.

[1] The IOBAD pointer in the DCB points to a location that contains the address of the corresponding data event control block (DECB) for these records. The format of the DECB is illustrated as part of the BSAM READ macro instruction in *OS/VS2 MVS Data Management Macro Instructions.*

Figure 21-1. Parameter Lists for Nonlabel Processing Exit Routines

Returning from an Exit Routine

An exit routine returns control to the utility program by means of the macro return instruction in the exit routine.

The format of the RETURN macro instruction is:

```
[label] RETURN     [ (r1,r2) ]
                   [,RC={n | (15)}]
```

where:

(r1,r2)
specifies the range of registers to be reloaded by the utility program from the register save area. If this parameter is omitted, the registers are considered properly restored by the exit routine.

RC=
specifies a return code in register 15. If RC is omitted, register 15 is loaded as specified by (r1,r2). These values can be coded:

n
specifies a return code to be placed in the 12 low order bits of register 15.

(15)
specifies that general register 15 already contains a valid return code.

The user's label processing routine must return a code in register 15 as shown in Figure 21-2 unless:

- The buffer address was set to zero before entry to the label processing routine. In this case, the system resumes normal processing regardless of the return code.
- The user's label processing routine was entered after an uncorrectable output error occurred. In this case the system attempts to resume normal processing.

Figure 21-2 shows the return codes that can be issued to utility programs by user exit routines. Slightly different return codes are used for the UPDATE=INPLACE option of the IEBUPDTE program. See the discussion of UPDATE=INPLACE in the chapter "IEBUPDTE Program."

Note: For a list of return codes issued by IEBTCRIN at job termination, see the "IEBTCRIN Program" chapter of this publication.

Type of Exit	Return Code	Action
Input Header or Trailer Label	0	The system resumes normal processing. If there are more labels in the label group, they are ignored.
	4	The next user label is read into the label buffer area and control is returned to the user's routine. If there are no more labels, normal processing is resumed.
	16	The utility program is terminated on request of the user routine.
Output Header or Trailer Label	0	The system resumes normal processing. No label is written from the label buffer area.
	4	The user label is written from the label buffer area. The system then resumes normal processing.
	8	The user label is written from the label buffer area. If fewer than eight labels have been created, the user's routine again receives control so that it can create another user label. If eight labels have been created, the system resumes normal processing.
	16	The utility program is terminated on request of the user routine.
Totaling Exits	0	Processing continues, but no further exits are taken.
	4	Normal operation continues.
	8	Processing ceases, except for EOD processing on output data set (user label processing).
	16	Utility program is terminated.
All other exits (except IEBTCRIN's ERROR and OUTREC, and IEBPTPCH's exit OUTREC)	0-11 (Set to next multiple of four)	Return code is compared to highest previous return code; the higher is saved and the other discarded. At the normal end of job, the highest return code is passed to the calling processor.
	12 or 16	Utility program is terminated and this return code is passed to the calling processor.
ERROR	0	Record is not placed in the error data set. Processing continues with the next record.
	4	Record is placed in the error data set (SYSUT3).
	8	Record is not placed in error data set but is processed as a valid record (sent to OUTREC and SYSUT2 if specified). IEBTCRIN removes the EDW from an edited MTDI record before processing continues.
	16	Utility program is terminated.
OUTREC (IEBTCRIN)	0	Record is not placed in normal output data set.
	4	Record is placed in normal output data set (SYSUT2).
	16	Utility program is terminated.
OUTREC (IEBPTPCH)	4	Record is not placed in normal output data set.
	12 or 16	Utility program is terminated.
	Any other number	Record is placed in normal output data set (SYSUT2).

Figure 21-2. Return Codes Issued by User Exit Routines

Further information on the use of the RETURN macro instruction is contained in *OS/VS2 MVS Supervisor Services and Macro Instructions.*

APPENDIX B. INVOKING UTILITY PROGRAMS FROM A PROBLEM PROGRAM

Utility programs can be invoked by a problem program through the use of the ATTACH or LINK macro instruction. In addition, IEBTCRIN can be invoked through the use of the LOAD or CALL macro instruction.

The problem program must supply the following to the utility program:

- The information usually specified in the PARM parameter of the EXEC statement.
- The ddnames of the data sets to be used during processing by the utility program.

The following utility programs require that calling programs be authorized via the Authorized Program Facility (APF):

IEBCOPY, IEHATLAS, IEHDASDR,
IEHINIT, IEHMOVE, IEHPRGM

See the *OS/VS2 Conversion Notebook* for details on program authorization.

Note: When IEHMOVE, IEHPROGM, or IEHLIST is dynamically invoked in a job step containing a program other than one of these three, the DD statements defining mountable devices for the IEHMOVE, IEHPROGM, or IEHLIST program must be included in the job stream prior to DD statements defining data sets required by the other program.

LINK or ATTACH Macro Instruction

The LINK or ATTACH macro instruction can be used to invoke a utility program from a problem program.

The format of the LINK or ATTACH macro instruction is:

[*label*]	**{LINK | ATTACH}**	**EP=***progname* **,PARAM=(***optionaddr* [, *ddnameaddr*] [, *hdingaddr*]) **,VL=1**

where:

EP=*progname*
specifies the symbolic name of the utility program.

PARAM=
specifies, as a sublist, address parameters to be passed from the problem program to the utility program. These values can be coded:

optionaddr
specifies the address of an option list, which is usually specified in the PARM parameter of the EXEC statement. This address must be written for all utility programs.

ddnameaddr
specifies the address of a list of alternate ddnames for the data sets used during utility program processing. If standard ddnames are used and this is not the last parameter in the list, it should point to a halfword of zeros. If it is the last parameter, it may be omitted.

hdingaddr

specifies the address of a six-byte list, HDNGLIST, which contains an EBCDIC page count for the output device. If *hdingaddr* is omitted, the page number defaults to 1.

VL=1

specifies that the sign bit of the last fullword of the address parameter list is to be set to 1.

Figure 22-1 shows these lists as they exist in the user's DC area. Note that the symbolic starting addresses for OPTLIST and DDNMELST fall on halfword boundaries.

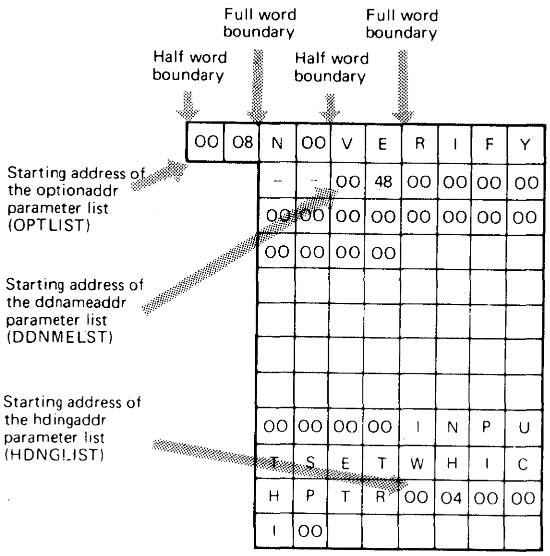


Figure 22-1. Typical Parameter Lists

The PARAM parameter of the LINK macro instruction in the calling program provides the utility program with the symbolic addresses of the parameter lists shown in Figure 22-1, as follows:

- The option list, OPTLIST, which includes the number of bytes in the list (hexadecimal 08) and the NOVERIFY option.
- The alternate ddname list, DDNMELST, which includes the number of bytes in the list (hexadecimal 48) and alternative names for the SYSIN, SYSUT1, and SYSUT2 data sets.
- The heading list, HDNGLIST, which includes the number of bytes in the list (hexadecimal 04) and indicates the starting page number (shown as 10) for printing operations controlled through the SYSPRINT data set.

The option list, OPTLIST, must begin on a halfword boundary that is not also a fullword boundary. The two high order bytes contain a count of the number of bytes in the remainder of the list. (For all programs except IEHMOVE, IEHLIST, IEHPROGM, IEHINITT, IEBUPDTE, and IEBISAM, the count must be zero.) OPTLIST is free form with fields separated by commas. No blanks or zeros should appear in the list.

The ddname list, DDNMELST, must begin on a halfword boundary that is not also a fullword boundary. The two high order bytes contain a count of the number of bytes in the remainder of the list. Each name of fewer than eight bytes must be left aligned and padded with blanks. If an alternate ddname is omitted from the list, the standard name is assumed. If the name is omitted within the list, the eight-byte entry must contain binary zeros. Names can be omitted from the end by merely shortening the list. Figure 22-2 shows the sequence of the eight-byte entries in the ddname list pointed to by *ddnameaddr*.

Entry	Standard Name
1	00000000
2	00000000
3	00000000
4	00000000
5	SYSIN
6	SYSPRINT
7	00000000
8	SYSUT1
9	SYSUT2
10	SYSUT3
11	SYSUT4

Figure 22-2. Sequence of DDNMELST Entries

The first two bytes of HDNGLIST contain the length in bytes of the heading list. The remaining four bytes contain a page number that the utility program is to place on the first page of printed output.

LOAD Macro Instruction

IEBTCRIN can be invoked through use of the LOAD macro instruction.

The LOAD macro instruction causes the control program to bring the load module containing the specified entry point into main storage unless a copy is already there. Control is not passed to the load module.

The format of the LOAD macro instruction is:

[*label*] **LOAD {EP=IEBTCRIN | EPLOC=***address of name* **}**

where:

EP=IEBTCRIN
is the entry point name of the program to be brought into main storage.

EPLOC=*address of name*
is the main storage address of the entry point name described above.

CALL Macro Instruction

The CALL macro instruction can be used to pass control to IEBTCRIN after IEBTCRIN has been loaded into main storage.

Control can be passed to IEBTCRIN via a CALL macro instruction or via a branch and link instruction. If the branch and link instruction is used, register 1 must be loaded with the address of a parameter list of full words as described under "LINK or ATTACH Macro Instruction." The last parameter list address must contain X'80' in byte 1 to indicate the last parameter in the list.

The format of the CALL macro instruction is:

[*label*] **CALL IEBTCRIN(,** *optionaddr* **[,** *ddnameaddr* **][,** *hdingaddr* **]),VL**

where:

IEBTCRIN
is the name of the entry point to be given control; the name is used in the macro instruction as the operand of a V-type address constant.

optionaddr
specifies the address of an option list, OPTLIST, usually specified in the PARM parameter of the EXEC statement. This address must be written for all utility programs.

ddnameaddr
specifies the address of a list of alternate ddnames, DDNMELST, for the data sets used during utility program processing. If standard ddnames are used and this is not the last parameter in the list it should point to a halfword of zeros. If it is the last parameter, it may be omitted.

hdingaddr
specifies the address of a six-byte list containing an EBCDIC page count for the output device.

VL
specifies that the high order bit of the last address parameter in the macro expansion is to be set to 1.

The option list, OPTLIST, must begin on a halfword boundary that is not also a fullword boundary. The two high order bytes contain a count of the number of bytes in the remainder of the list. (For all programs except IEHMOVE, IEHPROGM, IEHINITT, and IEBISAM, the count must be zero.) The option list is free form with fields separated by commas. No blanks or zeros should appear in the list.

The ddname list, DDNMELST, must begin on a halfword boundary that is not also a fullword boundary. The two high order bytes contain a count of the number of bytes in the remainder of the list. Each name of fewer than eight bytes must be left aligned and padded with blanks. If an alternate ddname is omitted from the list, the standard name is assumed. If the name is omitted within the list, the eight-byte entry must contain binary zeros. Names can be omitted from the end by merely shortening the list. The sequence of the eight-byte entries in the ddname list pointed to by *ddnameaddr* is shown earlier in Figure 22-2.

The first two bytes of the heading list, HDNGLIST, contain the length in bytes of the heading list. The remaining four bytes contain a page number that the utility program is to place on the first page of printed output.

APPENDIX C. DD STATEMENTS FOR DEFINING MOUNTABLE DEVICES

When defining mountable devices to be used by system utility programs IEHPROGM, IEHMOVE, IEHLIST, or IEHDASDR, the user must consider the implications of the DD statements he uses to define those devices.

DD statement parameters must ensure that no one else has access to either the volume or the data set. In any case, caution should be used when altering volumes that are permanently resident or reserved.

Under normal conditions, a mountable device should not be *shared* with another job step; that is, if a utility program is used to update a volume on a mountable device, the volume being updated must remain mounted until the operation is completed.

Following are ways to ensure that mountable devices are not shared:

- Specify DEFER in a DD statement defining a mountable device.
- Specify unit affinity on a second DD statement defining a mountable device.
- Specify a volume count in the VOLUME parameter of a DD statement that is greater than the number of mountable devices to be allocated.
- Specify PRIVATE in a DD statement defining a mountable device.

For a detailed discussion, see *OS/VS2 JCL.*

DD Statement Examples

In the following examples of DD statements, an IBM DASD is indicated as the mountable device. Alternative parameters are stacked.

Note: Examples which use *disk* in place of actual device-ids, must be changed before use. See the Device Support section, in the Introduction to this manual, for valid device-id notation.

DD Example 1

This DD statement makes a specific request for a private, non-sharable volume or volumes to be mounted on a single device.

```
//DD1   DD   UNIT=(disk,,DEFER),DISP=(,KEEP),
// VOLUME=(PRIVATE,SER=(123456)),SPACE=(CYL,(1,1))
```

A utility program causes a mount message to be issued for a specific volume when the volume is required for processing by the program. The user should supply the operator with the clearly marked volume or volumes to be mounted during the job step.

This DD statement ensures that the volume integrity of a mountable volume is maintained. If only one volume is to be processed, it is mounted at the start of the job step and demounted at the end of the step. If additional volumes are processed, they are mounted and demounted when needed by the utility program. The last volume to be processed is demounted at the end of the job step.

DD Example 2

This DD statement makes a request for a private, non-sharable volume.

```
//DD2   DD   UNIT=(disk,,DEFER),VOLUME=PRIVATE,DISP=(NEW,KEEP),
//     SPACE=(CYL,(1,1))
```

The results of this statement are identical to those shown in DD Example 1.

If a specific unit is requested and the volume serial number is not given in the DD statement, the user must be certain that either: (1) the desired volume is already mounted on that unit, or (2) a volume is not mounted, causing the system to issue a mount message.

Note: This statement can be used only if the user is certain that a removable volume, rather than a fixed volume, will be allocated by the scheduler. If there is any chance that a fixed volume will be allocated, this statement must not be used.

DD Example 3

This DD statement makes a specific request for a private, sharable volume to be mounted on a device.

```
//DD1 DDUNIT=disk,VOLUME=(PRIVATE,SER=(121212)),DISP=OLD
```

This DD statement does not ensure that volume integrity is maintained. It should be used with extreme caution in a multiprogramming environment because there is the possibility that a job step running concurrently might make a specific request for the volume, use the volume, and demount it.

DD Example 4

This DD statement makes a specific request for a public, non-sharable volume to be mounted on a device.

```
//DD3 DD UNIT=(disk,,DEFER),VOLUME=SER=789012,DISP=OLD
```

If the volume is already mounted, it is used. The volume remains mounted at the end of the job step, and is not demounted until another job step requires the device on which the volume is mounted.

This DD statement ensures that volume integrity is maintained between jobs; two or more such statements in a single job can allocate the same device.

DD Example 5

This DD statement makes a specific request for a public, sharable volume to be mounted on a device.

```
//DD1   DD   UNIT=disk,VOLUME=SER=654321,DISP=OLD
```

If the volume is already mounted, it is used. The volume remains mounted at the end of the job step, and is not demounted until another job step requires the device on which the volume is mounted. (This DD statement can also be used to define permanently resident devices.)

This DD statement does not ensure that the volume integrity of a mountable volume is maintained. It should be used with extreme caution in a multiprogramming environment because there is the possibility that a job step running concurrently might use the device.

APPENDIX D. PROCESSING USER LABELS

User labels can be processed by IEBGENER, IEBCOMPR, IEBPTPCH, IEHMOVE, IEBCTRIN, and IEBUPDTE. In some cases, user-label processing is automatically performed; in other cases, you must indicate the processing to be performed. In general, user label support allows the utility program user to:

- Process user labels as data set descriptors.
- Process user labels as data.
- Total the processed records prior to each WRITE command (IEBGENER and IEBUPDTE only).

For either of the first two options, the user must specify standard labels (SUL) on the DD statement that defines each data set for which user-label processing is desired. For totaling routines, OPTCD=T must be specified on the DD statement.

The user cannot update labels by means of the IEBUPDTE program. This function must be performed by a user's label processing routines. IEBUPDTE will, however, allow you to create labels on the output data set from data supplied in the input stream. See the discussion of the LABEL statement in the chapter "IEBUPDTE Program."

IEHMOVE does not allow exits to user routines and does not recognize options concerning the processing of user labels as data. IEHMOVE always moves or copies user labels directly to a new data set. See the chapters for the "IEHMOVE Program."

Volume switch labels of a multivolume data set cannot be processed by IEHMOVE, IEBGENER, or IEBUPDTE. Volume switch labels are therefore lost when these utilities create output data sets. To ensure that volume switch labels are retained, process multivolume data sets one volume at a time.

Processing User Labels as Data Set Descriptors

When user labels are to be processed as data set descriptors, one of the user's label processing routines receives control for each user label of the specified type. The user's routine can include, exclude, or modify the user label. Processing of user labels as data set descriptors is indicated on an EXITS statement with keyword parameters that name the label processing routine to be used.

The EXIT keyword parameters indicate that a user routine should receive control each time the OPEN, EOV, or CLOSE routine encounters a user label of the type specified.

Figure 24-1 illustrates the action of the system at OPEN, EOV, or CLOSE time. When OPEN, EOV, or CLOSE recognizes a user label and when SUL has been specified on the DD statement for the data set, control is passed to the utility program. Then, if an exit has been specified for this type of label, the utility program passes control to the user routine. The user's routine processes the label and returns control, along with a return code, to the utility program. The utility program then returns control to OPEN, EOV, or CLOSE.

This cycle is repeated up to eight times, depending upon the number of user labels in the group and the return codes supplied by the user's routine.

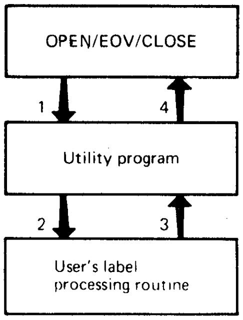


Figure 24-1. System Action at OPEN, EOV, or CLOSE Time

Exiting to a User's Totaling Routine

When an exit is taken to a user's totaling routine, an output record is passed to the user's routine just before the record is written. The first halfword of the totaling area pointed to by the parameter contains the length of the totaling area, and should not be used by the user's routine. If the user has specified user label exits, this totaling area (or an image of this area) is pointed to by the parameter list passed to the appropriate user label routine.

Note: An output record is defined as a physical record (block), except when IEBGENER is used to process and reformat a data set that contains spanned records.

Processing User Labels as Data

When user labels are processed as data, the group of user labels, as well as the data set, is subject to the normal processing done by the utility program. The user can have his labels printed or punched by IEBPTPCH, compared by IEBCOMPR, or copied by IEBGENER.

To specify that user labels are to be processed as data, include a LABELS statement in the job step that is to process user labels as data.

There is no direct relationship between the LABELS statement and the EXITS statement. Either or both can appear in the control statement stream for an execution of a utility program. If there are user label-processing routines, however, their return codes may influence the processing of the labels as data. In addition, a user's output label-processing routine can override the action of a LABELS statement because it receives control before each output label is written. At this time the label created by the utility as a result of the LABEL statement is in the label buffer, and the user's routine can modify it.

The code returned by the user's totaling routine determines system response as follows:

- 0, which specifies that processing is to continue, but no further exits are to be taken.
- 4, which specifies that normal processing is to continue.
- 8, which specifies that processing is to terminate, except for EOD processing on the output data set (user label processing).
- 16, which specifies that processing is to be terminated.

INDEX

If more than one page number is given, the primary discussion is listed first.

GC26-3902-1

OS/VS2 MVS Utilities Printed in U.S.A. GC26-3902-1

OS/VS2 MVS Utilities
GC26-3902-1

Reader's Comment Form

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM shall have the nonexclusive right, in its discretion, to use and distribute all submitted information, in any form, for any and all purposes, without obligation of any kind to the submitter. Your interest is appreciated.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Fold on two lines, tape, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

GC26-3902-1

Reader's Comment Form

Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and Tape

IBM®

OS/VS2 MVS Utilities Printed in U.S.A. GC26-3902-1

OS/VS2 MVS Utilities
GC26-3902-1

Reader's Comment Form

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM shall have the nonexclusive right, in its discretion, to use and distribute all submitted information, in any form, for any and all purposes, without obligation of any kind to the submitter. Your interest is appreciated.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Fold on two lines, tape, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

Please use pressure sensitive or other gummed tape to seal this form.

Reader's Comment Form

Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and Tape

IBM®

OS/VS2 MVS Utilities
GC26-3902-1

Reader's Comment Form

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM shall have the nonexclusive right, in its discretion, to use and distribute all submitted information, in any form, for any and all purposes, without obligation of any kind to the submitter. Your interest is appreciated.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Fold on two lines, tape, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

GC26-3902-1

Reader's Comment Form

Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and Tape

IBM®

OS/VS2 MVS Utilities Printed in U.S.A. GC26-3902-1

OS/VS2 MVS Utilities
GC26-3902-1

Reader's Comment Form

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM shall have the nonexclusive right, in its discretion, to use and distribute all submitted information, in any form, for any and all purposes, without obligation of any kind to the submitter. Your interest is appreciated.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Please use pressure sensitive or other gummed tape to seal this form.

Fold on two lines, tape, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

Reader's Comment Form

Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and Tape

IBM®