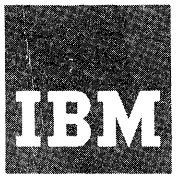
File No. S360-32
GC28-6586-11

OS



Systems Reference Library

IBM System/360 Operating System: Utilities

This publication discusses the capabilities of the IBM System/360 Operating System utility programs and the control statements used with each program. These programs are used by programmers responsible for organizing and maintaining operating system data.

Three types of utility programs are discussed: system utilities and data set utilities, which are used directly with the System/360 Operating System; and independent utilities, which operate outside the operating system. System utilities deal with operating system control data. Data set utilities manipulate data sets at the record level and above. Independent utilities initialize, dump, and restore direct access volumes.

Information concerning Model 195 support is for planning purposes only.

Preface

System/360 Operating System Utility programs provide functions to assist programmers responsible for creating and maintaining operating system data. These functions are requested through control statements, which can be used individually or in combination, to perform a variety of housekeeping and support operations.

This publication discusses the functions provided by each utility program and the control statements used to request these functions. Examples illustrating possible uses of the utilities follow the description of each utility program. Appendixes A and B contain information for linking to exit routines and invoking utility programs. Appendix C contains information explaining the control statement notation used throughout this publication. Appendix D contains information on defining mountable devices and maintaining volume integrity. Appendix E describes how to build a generation data group index and catalog a generation. Appendix F discusses utility program handling of user labels. Appendix G contains the messages and return codes issued by the utility programs.

The reader should be familiar with the concepts and terminology introduced in the prerequisite publications.

PREREQUISITE PUBLICATIONS

IBM System/360 Operating System:

Job Control Language Reference, GC28-6704

Job Control User's Guide, GC28-6703

Supervisor and Data Management Services, GC28-6646

Supervisor and Data Management Macro Instructions, GC28-6647

Twelfth Edition (June, 1970)

This is a major revision of, and obsoletes, C28-6586-10. The additions and changes to this revision include:

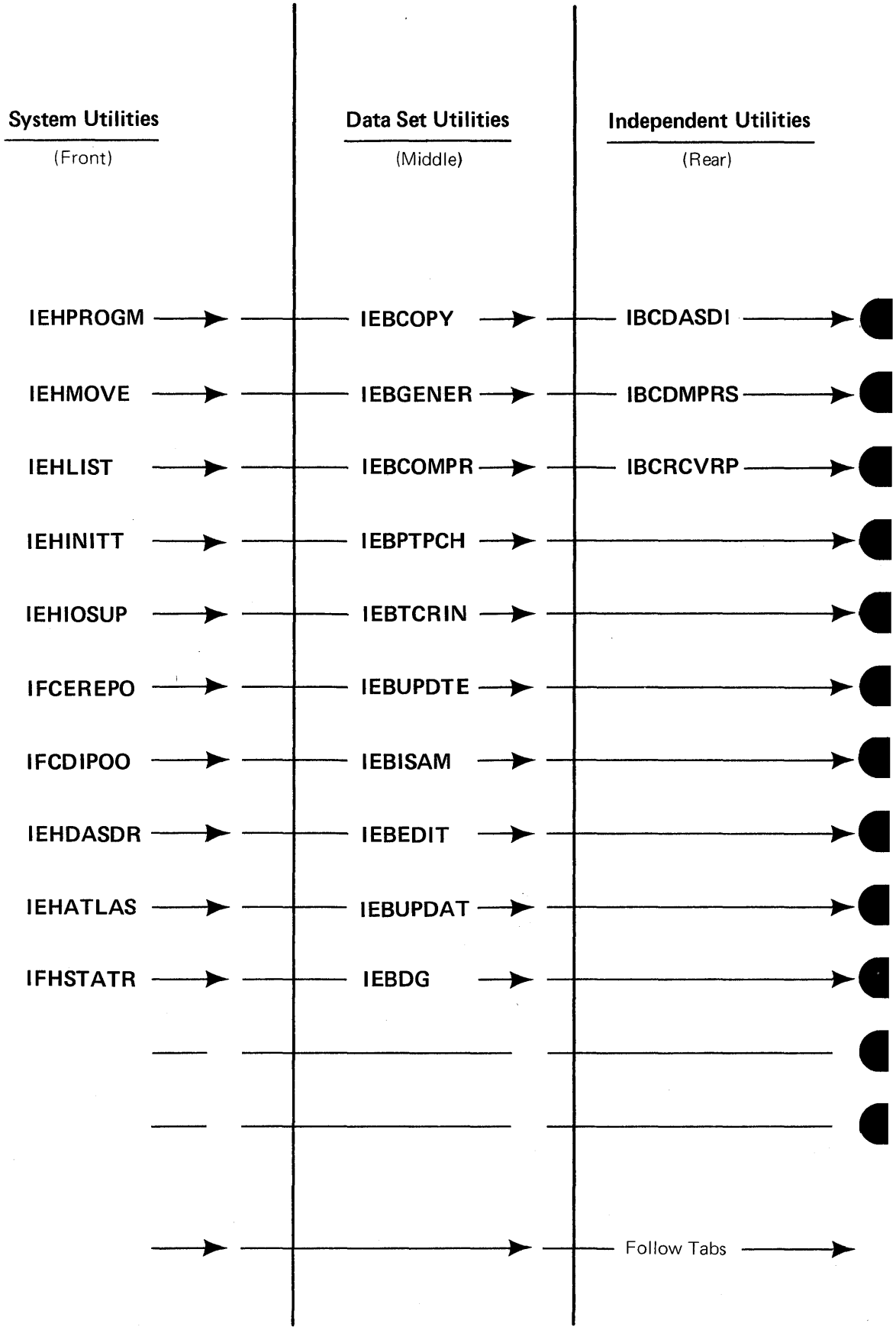
- A new utility to assign alternate tracks when defective areas are indicated (IEHATLAS).
- An improved IEBCOPY utility program.
- A new edited format for the LISTPDS function of IEHLIST.
- A new utility supporting the 2495 Tape Cartridge Reader (IEBTCRIN).
- Deletion of the IEHUCSLD program.
- IEHMOVE Support of BDAM Variable Spanned Records (VRE).
- A new utility to list error information from the SYS1.MAN data set (IFHSTATR).

Changes to the text are indicated by vertical lines to the left of the change and by bullets to the left of illustration captions.

This edition applies to release 19 of the IBM System/360 Operating System, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/360 SRL Newsletter, Order No. GN20-0360 for the editions that are applicable and current.

Requests for copies of IBM publications should be made to your IBM representative or to the branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Systems Publications, Department D58, PO Box 390, Poughkeepsie, N. Y. 12602


System Utilities
(Front)
Data Set Utilities
(Middle)
Independent Utilities
(Rear)
IEHPROGM
IEHMOVE
IEHLIST
IEHINITT
IEHIOSUP
IFCEREPO
IFCDIPOO
IEHDASDR
IEHATLAS
IFHSTATR
IEBCOPY
IEBGENER
IEBCOMPR
IEBPTPCH
IEBTCRIN
IEBUPDTE
IEBISAM
IEBEDIT
IEBUPDAT
IEBDG
IBCDASDI
IBCDMPRS
IBCRCVRP
Follow Tabs

Contents

Illustrations

Figures

Charts

Tables

Summary of Major Changes--Release 19

Utility	Description
IEHMOVE	This utility program now supports direct data sets with variable spanned (BDAM VRE) records.
IEHLIST	This utility program can now, optionally, format directory listings of partitioned data sets.
IEHUCSLD	This utility program has been deleted from the utility package.
IFCEREP0	This utility program now provides T-type (bulk data) environment records. It also supports Models 95 and 195; however, information concerning Model 195 support is for planning purposes only.
IEHDASDR	This utility program now has a "QUICK DASDI" feature applicable to all direct access volumes supported by the IEHDASDR utility program. There is also additional information concerning the RESTORE statement dealing with the restoration of tapes created by the IBCDMPRS utility program.
IEHATLAS	This is a new utility program designed to assign an alternate track when a defective one is indicated by a data check or missing address marker.
IFHSTATR	This is a new utility program designed to select, format, and write information from error statistics by volume (ESV) records found on the IFASMFDP tape or the SYS1.MAN data set.
IEBCOPY	This utility program has been completely rewritten. It now includes many new features, including replacing and/or renaming members on a partitioned data set.
IEBTCRIN	This is a new utility program designed to read, edit, and write input from the IBM 2495 Tape Cartridge Reader. Exit routine linkage for the IEBTCRIN utility program is discussed in Appendix A. Invoking the IEBTCRIN utility program is discussed in Appendix B.
Appendix G	In Appendix G, the description of most messages includes a programmer response. A more detailed response is included in the publication IBM System/360 Operating System: Messages and Codes, GC28-6631. Refer to this publication before responding to any message or calling IBM.

Introduction

The IBM System/360 Operating System provides utility programs to assist in organizing and maintaining data. Each utility program described in this publication falls into one of three general classes of programs:

- System utility programs.
- Data set utility programs.
- Independent utility programs.

The Program Classes

The program class into which a specific utility program falls is determined by the function that the utility program performs and the manner in which the program is controlled.

THE SYSTEM UTILITY PROGRAMS

The system utility programs described in this publication are:

IEHPROGM	IEHIOSUP	IEHATLAS
IEHMOVE	IFCEREP0	IFHSTATR
IEHLIST	IFCDIP00	
IEHINITT	IEHDASDR	

These programs are used to maintain system control data at an organizational or system level. The following general functions are performed by the system utility programs:

- IEHPROGM -- builds and maintains system control data.
- IEHMOVE -- moves or copies collections of data.
- IEHLIST -- lists system control data.
- IEHINITT -- writes standard labels onto magnetic tape volumes.
- IEHIOSUP -- updates entries in the supervisor call library.
- IFCEREP0 -- edits and lists error environment records.
- IFCDIP00 -- reinitializes the system data set SYS1.LOGREC.
- IEHDASDR -- initializes direct access volumes; dumps or restores data.
- IEHATLAS -- assigns alternate track when defective tracks are indicated.
- IFHSTATR -- selects, formats, and writes information about tape errors from the IFASMFDP tape or the SYS1.MAN data set.

The user controls the operation of a system utility program through use of job control statements and utility control statements. Refer to Appendix C for the format and notation of control statements.

THE DATA SET UTILITY PROGRAMS

The data set utility programs described in this publication are:

IEBCOPY	IEBTCRIN	IEBUPDAT
IEBGENER	IEBUPDTE	IEBDG
IEBCOMPR	IEBISAM	
IEBPTPCH	IEBEDIT	

These programs are used to reorganize, change, or compare data at the data set level and/or at the record level. The following general functions are performed by the data set utility programs:

- IEBCOPY -- copies, compresses or merges partitioned data sets; selects or excludes specified members in a copy step/operation; renames and/or replaces members of a partitioned data set.
- IEBGENER -- copies records from a sequential data set or converts a data set from sequential to partitioned organization.
- IEBCOMPR -- compares records in sequential or partitioned data sets.
- IEBPTPCH -- prints or punches records residing in a sequential or partitioned data set.
- IEBCTRIN -- constructs records from input read from the 2495 tape cartridge reader.
- IEBUPDTE -- updates a symbolic library.
- IEBISAM -- places source data from an indexed sequential data set into a sequential data set suitable for subsequent reconstruction.
- IEBEDIT -- creates an input stream.
- IEBUPDAT -- updates a symbolic library.
- IEBDG -- creates a patterned data set to be used as a debugging aid.

The user controls the operation of a data set utility program through use of job control statements and utility control statements. Refer to Appendix C for the format and notation of control statements.

THE INDEPENDENT UTILITY PROGRAMS

The independent utility programs operate outside, and in support of, the IBM System/360 Operating System. The independent utility programs described in this publication are:

IBCDASDI (DASDI)
IBCDMPRS (DUMP/RESTORE)
IBCRCVRP (RECOVER/REPLACE)

These programs are used to prepare direct access devices for system use and to ensure that any permanent hardware errors incurred on a direct access device (i.e., defective tracks) do not seriously degrade the performance of that device. The following general functions are performed by the independent utility programs:

- IBCDASDI -- initializes and assigns alternate tracks to a direct access volume.
- IBCDMPRS -- dumps and restores the data contents of a direct access volume.
- IBCRCVRP -- recovers usable data from a defective track, assigns an alternate track, and merges replacement data with the recovered data onto the alternate track.

The user controls the operation of an independent utility program through use of utility control statements. Since the programs are independent of the operating system, job control statements are not required. Refer to Appendix C for the format and notation of control statements.

Selecting a Program

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 (specifically the IEHPROGM program) can be used to rename the data set.

In some cases, a specific function can be performed by more than one program. The following utility program charts are provided to lead the user to examples of operations similar to those he wishes to perform.

The system utilities chart (Chart 1) is organized by function.

The data set utility chart (Chart 2) is organized first by the organization of the data set or sets to be processed, and second, by the function to be performed.

The independent utilities chart (Chart 3) is organized by function.

Chart 1. SYSTEM UTILITIES (Part 1 of 2)

For This Operation		Go to This Example
Build	a generation data group index	IEHPROGM 6 IEHPROGM 7
Catalog	a data set	IEHPROGM 3 IEHPROGM 5
	a generation data set	IEHPROGM 7
Connect	volumes	IEHPROGM 5
Copy	cataloged sequential data sets	IEHMOVE 2
	sequential data set	IEHMOVE 12 (see also-data set utilities)
Delete	an index structure	IEHPROGM 2 IEHPROGM 3 IEHPROGM 4
Dump	an entire direct access volume	IEHDASDR 4 IEHDASDR 6 IEHDASDR 8
	a group of tracks	IEHDASDR 5
Edit and list	error environment records	IFCEREP0 1-7
	error statistics by volume (ESV) records	IFHSTATR
Get Alternate Tracks	on a direct access volume	IEHDASDR 3 IEHATLAS 1-3
Initialize	a direct access volume	IEHDASDR 1 IEHDASDR 2
	the SYS1.LOGREC data set	IFCDIP00 1
Label	magnetic tape volumes	IEHINITT 1-4
Load	an unloaded data set	IEHMOVE 11

(Part 1 of 2)

Chart 1. SYSTEM UTILITIES (Part 2 of 2)

For This Operation		Go to This Example
List	a volume table of contents	IEHLIST 4
	partitioned directories	IEHLIST 3 (see also-IEBPTPCH 7)
	the contents of the catalog (SYSCTLG data set)	IEHLIST 1 IEHLIST 2
Merge	partitioned data sets	IEHMOVE 4 (see also-IEBCOPY 2 IEBCOPY 3)
Move	a catalog	IEHMOVE 5 IEHMOVE 6
	a group of cataloged data sets	IEHMOVE 3
	a volume of data sets	IEHMOVE 7
	Partitioned data sets	IEHMOVE 4 IEHMOVE 8 IEHMOVE 9
	sequential data sets	IEHMOVE 1
Rename	a data set	IEHPROGM 3
Restore	data on direct access volumes	IEHDASDR 7
Scratch	a volume table of contents	IEHPROGM 1
	data sets	IEHPROGM 1 IEHPROGM 2
Uncatalog	data sets	IEHPROGM 2 IEHPROGM 3 IEHPROGM 4
Unload	a partitioned data set	IEHMOVE 9
	a sequential data set	IEHMOVE 10
Update	TTR entries in the supervisor call library	IEHIOSUP 1 IEHIOSUP 2

Chart 2. DATA SET UTILITIES (Part 1 of 3)

For this Organization	and this Operation		Go to this Example
Sequential	Compare	9-track tape & 9-track tape	IEBCOMPR 1 IEBCOMPR 6
		9-track tape & 7-track tape	IEBCOMPR 3
		9-track tape & card	IEBCOMPR 4
		7-track tape & 7-track tape	IEBCOMPR 2
	Convert to Partitioned	card to disk	IEBUPDTE 2
	Copy (see also system utilities)	card to 9-track tape	IEBGENER 1 IEBGENER 3
		card to 7-track tape	IEBGENER 2
		card to punch	IEBPTPCH 8
		card to disk	IEBGENER 4
		card to printer	IEBGENER 5
	Create an output data set	card to disk	IEBDG 6
		tape to disk	IEBDG 2
		tape to tape	IEBDG 1
		to tape from utility control statements only	IEBDG 3
		tape cartridge to direct access storage facility	IEBCTRIN
	Create an output job stream	9-track tape to 9-track tape	IEBEDIT 1
		7-track tape to 7-track tape	IEBEDIT 2
		Disk to 9-track tape	IEBEDIT 3
		Disk to Disk	IEBEDIT 4
		Input stream to 9-track tape	IEBEDIT 5
	Edit & convert to partitioned	drum to drum	IEBGENER 7
		7-track tape to disk	IEBUPDTE 6
	Edit & copy	7-track tape to 7-track tape	IEBGENER 8
		disk to disk	IEBGENER 9
	Edit & print	9-track tape to system output	IEBPTPCH 5
		disk to system output	IEBPTPCH 9
	Edit & punch	disk to punch	IEBPTPCH 6

(Part 1 of 3)

Chart 2. DATA SET UTILITIES (Part 2 of 3)

For this Organization	and this Operation		Go to this Example
Sequential	Expand or compress (see also system utilities)	drum to drum (expand)	IEBGENER 7
		disk to disk (compress)	IEBGENER 9
	Print	card to system output	IEBGENER 5
		9-track tape to system output	IEBPTPCH 1
		disk to system output	IEBPTPCH 7
	Punch	7-track tape to punch	IEBPTPCH 2
		card to punch	IEBPTPCH 8
Partitioned	Compare	disk & disk	IEBCOMPR 5 IEBCOMPR 7
	Compress in place	disk	IEBCOPY 9 IEBCOPY 10
	Convert to sequential	drum to 9-track tape	IEBUPDTE 5
	Copy (see also system utilities)	disk to disk	IEBCOPY 1 IEBCOPY 4 IEBCOPY 5 IEBUPDTE 3
	Create a member	disk	IEBDG 5
	Delete records	drum to 9-track tape	IEBUPDTE 5
	Enter a procedure into a procedure library	input stream to disk	IEBUPDTE 1
	Expand (see also system utilities	drum to disk	IEBCOPY 2
		disk to disk	IEBCOPY 2
	Insert records	disk	IEBUPDTE 2
		disk to disk	IEBUPDTE 3 IEBUPDTE 7 IEBUPDTE 8
		drum to 9-track tape	IEBUPDTE 5
	Number records	disk to disk	IEBUPTDE 1 IEBUPDTE 3 IEBUPDTE 4 IEBUPDTE 7 IEBUPDTE 8

(Part 2 of 3)

Chart 2. DATA SET UTILITIES (Part 3 of 3)

For this Organization	and this Operation		Go to this Example
Partitioned	Print	drum to system output	IEBPTPCH 3
		disk to system output	IEBPTPCH 4
	Reblock a data set	disk	IEBDG 4
	Replace records	disk to disk	IEBUPDTE 3 IEBUPDTE 4 IEBCOPY 3
		drum to 9-track tape	IEBUPDTE 5
		disk to disk	IEBCOPY 3 IEBCOPY 6 IEBCOPY 7
	Rename	disk to disk	IEBCOPY 7
	Exclude	disk to disk	IEBCOPY 8
	Update in place	disk	IEBUPDTE 4
Indexed Sequential	Unload	disk to 9-track tape disk to 7-track tape	IEBISAM 1 IEBISAM 2
	Load	9-track tape to disk	IEBISAM 3

Chart 3. INDEPENDENT UTILITIES

For this Operation		Go to this Example
Assign alternate tracks	direct access volume	IBCDASDI 4
Dump	direct access volume onto 9-track tape	IBCDMPRS 1
Initialize	direct access volume	IBCDASDI 1 IBCDASDI 2 IBCDASDI 3
Recover data from defective tracks	direct access volumes	IBCRCVRP 1
Replace data on an alternate track	direct access volume	IBCRCVRP 2
Restore a dumped direct access volume	from 9-track tape	IBCDMPRS 2

CONTROL LANGUAGE NOTATION

The format for control statements is explained in Appendix C. The explanation clarifies the manner in which selective (optional) operations, operands, keywords, and parameters are indicated within this publication.

Section 1: System Utilities

System utility programs manipulate collections of data and system control information. This section describes the capabilities, requirements for execution, and examples of the use of each system utility program:

- IEHPROGM -- a program that builds and maintains system control data.
- IEHMOVE -- a program that moves or copies collections of data.
- IEHLIST -- a program that lists system control data.
- IEHINITT -- a program that writes standard labels onto magnetic tape volumes.
- IEHIOSUP -- a program that updates entries in the supervisor call library.
- IFCEREP0 -- a program that edits and lists error environment records.
- IFCDIP00 -- a program that reinitializes the system data set SYS1.LOGREC.
- IEHDASDR -- a program that initializes direct access volumes or dumps or restores data.
- IEHATLAS -- a program that assigns alternate tracks when defective tracks are indicated.
- IFHSTATR -- a program that selects, formats, and writes information about tape errors from the IFASMFDP tape or the SYS1.MAN data set.

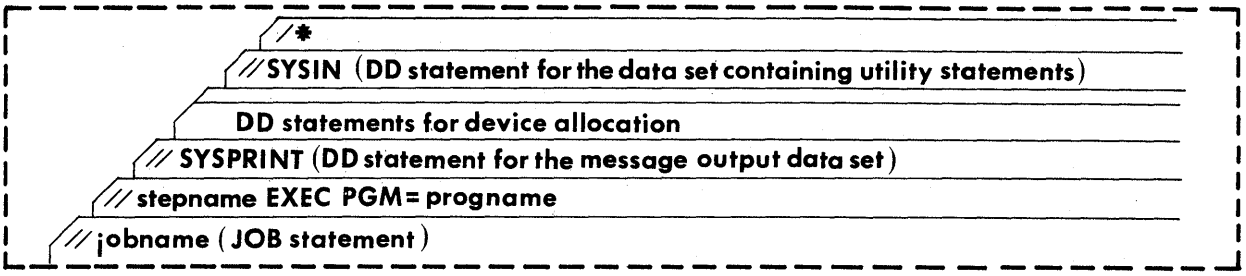
A system utility program is executed or invoked through the use of job control statements and utility control statements.

Job Control Statement Requirements

System utility programs are introduced as jobs or job steps, and are executed in response to job control statements and utility control statements. Each program to be executed requires:

- A JOB statement.
- An EXEC statement that identifies the program to be executed.
- DD statements that define the data sources and destinations, work data sets, and for programs IEHPROGM, IEHMOVE, IEHLIST, IEHDASDR, and IFHSTATR, mountable devices.

System Utility Figure 1 illustrates the general format of a set of job control statements used to execute a system utility program.



System Utility Figure 1. Executing a System Utility Program

METHODS OF EXECUTION

A set of job control statements for a utility program can be introduced to the operating system in different ways. The statements can be included in the input stream, or they can be placed in a procedure library, or the job can be invoked by a calling program.

Including the Job Control Statements in the Input Stream

A set of job control statements can be included directly in the input stream. In general, the examples throughout this publication reflect this application. In this case, the EXEC statement for the job step specifies the name of the utility program to be executed.

Entering a Set of Job Control Statements Into a Procedure Library

The execution of a program can be simplified by entering a set of job control statements for a utility program into a procedure library. The job control statements can then be referred to for subsequent applications of the program.

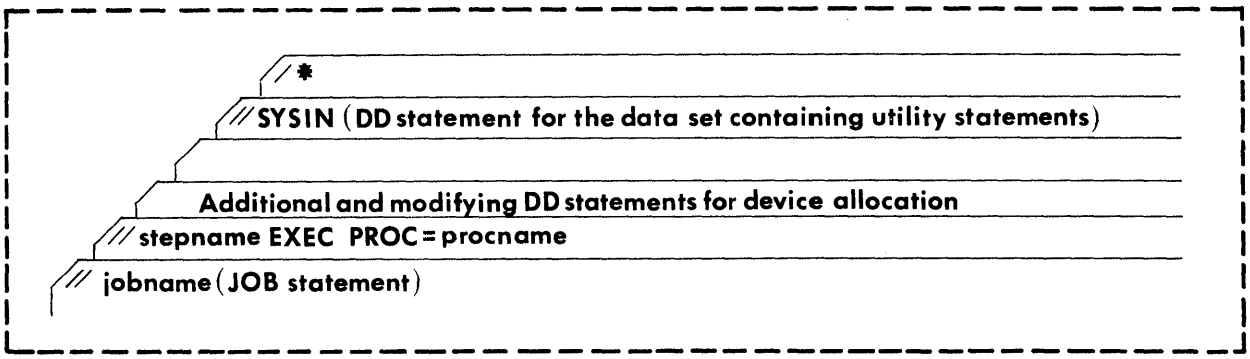
The statements in a procedure should satisfy the job control language requirements for most applications of the program; however, a procedure can be modified or supplemented for applications that require additional parameters, data sets, or devices.

Executing the Procedure: A procedure cannot contain a JOB control statement; that is, the execution of a procedure is initiated from the input stream. Within the input stream, an EXEC statement follows the JOB control statement and refers to the named procedure.

The first job statement in a procedure is an EXEC statement specifying the name of the utility program to be executed.

System Utility Figure 2 illustrates the general format of job control statements referring to a procedure. DD statements, defining additional device requirements and/or modifying DD statements in the procedure, are included.

Note: The data set utility program IEBUPDTE can be used to enter a procedure into a procedure library.



System Utility Figure 2. Executing a Cataloged Utility Procedure

Invoking a Utility Program

Except for programs IFCEREP0 and IFCDIP00, system utility programs can be employed as subroutines by use of standard operating system linkage conventions. At the completion or termination of the utility program, the highest return code encountered within the program is passed to the calling program. Invocation of utility programs and linkage conventions are discussed in Appendix B.

Note: When the IEHMOVE, IEHPROGM, or IEHLIST program 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.

MULTIPROGRAMMING CONSIDERATIONS

In an MVT environment, a region size should be specified for each application of a utility program. The region size is determined by the number of bytes in the utility program and, in the case of the IEHMOVE program, by the block size of an object data set (i.e., the largest block size in the job step). A region size can be specified as a parameter in the EXEC statement specifying the utility program name. The minimum region sizes are:

- IEHPROGM -- REGION=44K
- IEHMOVE -- REGION=16K + the largest block size in the job step rounded to the next highest multiple of 2K.
- IEHLIST -- REGION=44K
- IEHINITT -- REGION=14K
- IEHIOSUP -- REGION=10K
- IFCEREP0 -- REGION=20K
- IFCDIP00 -- REGION=10K
- IEHDASDR -- Refer to the publication IBM System/360 Operating System: Storage Estimates, GC28-6551 for detailed information on estimating REGION size.
- IEHATLAS -- Refer to the publication IBM System/360 Operating System: Storage Estimates, GC28-6551 for detailed information on estimating REGION size.
- IFHSTATR -- REGION=4K

NOTE: System utility programs should not be run in an MVT or MFT environment unless:

- Each data set to be used by the program is defined on a DD statement specifying the data set name and DISP=OLD (not applicable to programs IEHPROGM, IEHMOVE, and IEHLIST). This specification prevents other jobs from gaining access to a data set until the data set is no longer being used; that is, until the job has completed.
- DD statements defining mountable devices ensure that volumes mounted on those devices are nonsharable.
- Mountable volumes should not be made available to the system (premounted) until the user is requested by the system to mount the specified volumes.
- A reader procedure should be used which 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 in a multiprogramming environment, care should be taken to insure that the data set or volume being scratched is not being used by a program executing concurrently.

Utility Control Statement Specifications

A system utility program uses utility control statements to identify a particular function to be performed and, when required, to identify specific volumes or data sets to be processed.

Specifications of functions differ from program to program; these specifications are discussed within each of the programs.

This discussion deals specifically with device, volume, and data set specifications in utility control statements.

Identification Parameters

Utility control statements use keyword parameters to identify volumes by device types and volume serial numbers. Keyword parameters are also used to identify data sets. A data set residing on a direct access volume is identified by volume serial number and data set name; a data set residing on a magnetic tape volume must be further identified by its data set sequence number.

Applicable keyword parameters include VOL, CVOL, FROM, and TO. In general, a keyword parameter is coded as follows:

```
KEYWORD=device=({serial,seqno},...)
```

The term "device" is replaced by any of a group of device names that the installation defines when the operating system is generated. Each device name can represent:

- A single device.
- All devices of a specific type.

A device name can be a generic name, a substitute for a generic name, or a channel and unit address.

Generic name: (e.g., 2311, 2314, 2301, etc.) A generic name indicates that all devices of the specified type are represented. The generic name of a device does not differ from installation to installation or from generation to generation within an installation.

Substitute for a generic name (e.g., TAPE7, TAPE9, DRUM, DISK, or other installation designated names.) A substitute for a generic name is the equivalent of a generic name; for example, DISK might be used instead of 2311, or DRUM instead of 2301. However, unlike a generic name, which is applicable to each installation having that device type, a substitute is applicable only to the generated system in which that name is assigned.

A substitute that represents two or more different device types (e.g., DISK -- meaning 2311 and 2314) cannot be processed by a utility program.

Channel and unit address: (e.g., 190, 280, etc.) A channel and unit address represents one particular device.

The term "serial" is replaced by a 1- to 6-character volume serial number.

The term "seqno" is replaced by a data set sequence number. When a device other than tape is specified, the sequence number is omitted, as follows:

```
KEYWORD=device=(serial,serial,...)
```

If only a single direct access device is required, the parentheses can be deleted, as follows:

```
KEYWORD=device=serial
```

Hereafter, a volume parameter that may require more than one volume will be referred to as:

```
KEYWORD=device=list
```

A data set name must be specified as a fully qualified name.

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).

Program Applications

The IEHPROGM system utility program provides efficient facilities for modifying system control data and for maintaining data sets at an organizational level.

The program can be used to:

- Scratch a data set or a member.
- Rename a data set or a member.
- Catalog or uncatalog a data set.
- Build or delete an index or an index alias.
- Connect or release two volumes.
- Build and maintain a generation data group index.

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

The program can scratch the following items from a direct access volume:

- A sequential data set or sets.
- An indexed sequential data set or sets.
- A partitioned data set or sets.
- A member or members of a partitioned data set.
- A password-protected data set or sets.
- A data set or sets named by the operating system.
- A direct access data set.

In addition, multiple volumes can be specified for a scratch operation.

The Scratched Data Set: 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, and its space is made available for reallocation.

The space occupied by a data set residing on a device that operates in split-cylinder mode is not available for reallocation until all data sets sharing the cylinder have been scratched.

The Scratched Member: A member is considered scratched when its name is removed from the directory of the partitioned data set that contains it. The space occupied by a scratched member is not available for reallocation until the partitioned data set, itself, is scratched. (When scratching a member of a partitioned data set, all alias names of that member should also be removed from the directory.)

Renaming a Data Set or Member

The program can rename a data set or member that resides on a direct access volume. In addition, the program can be used to change the alias name or names, if any, of a member.

Cataloging or Uncataloging a Data Set

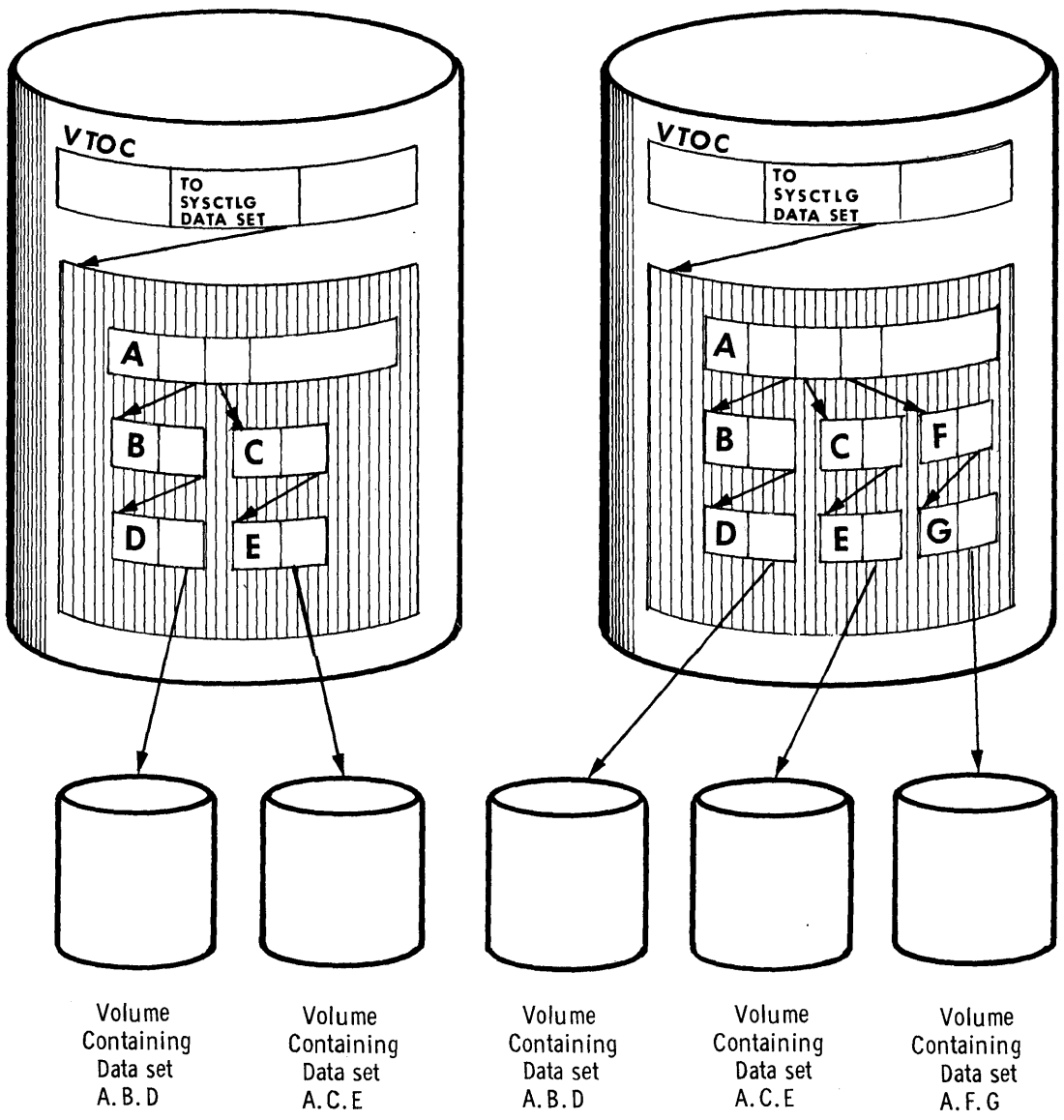
The program can catalog or uncatalog a sequential data set, an indexed sequential data set, a partitioned data set, or a direct access (BDAM) data set.

The Cataloged Data Set: A data set is cataloged when its fully qualified name and volume identification are entered in one or more index levels of the catalog (SYSCTLG 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. If higher-level indexes are necessary to catalog the data set, they are automatically created. The catalog function is used to:

- Catalog a data set that was not cataloged when it was created.
- Satisfy, if necessary, the requirement that a higher level index or indexes be created. For example, IEHPROGM Figure 1 shows how data set A.F.G is cataloged on the system residence volume. Note that the level F index does not exist in the SYSCTLG data set prior to the catalog operation.

Note: The catalog function of the IEHPROGM program differs from a DISP=(,CATLG) specification in a DD statement in that:

1. The DISP=(,CATLG) specification cannot create higher-level indexes.
2. The DISP=(,CATLG) specification cannot catalog a data set on a volume other than the system residence volume unless the system residence volume is properly "connected" to the other volume. (Refer to "Connecting or Releasing Two Volumes" for a discussion of connected volumes.)

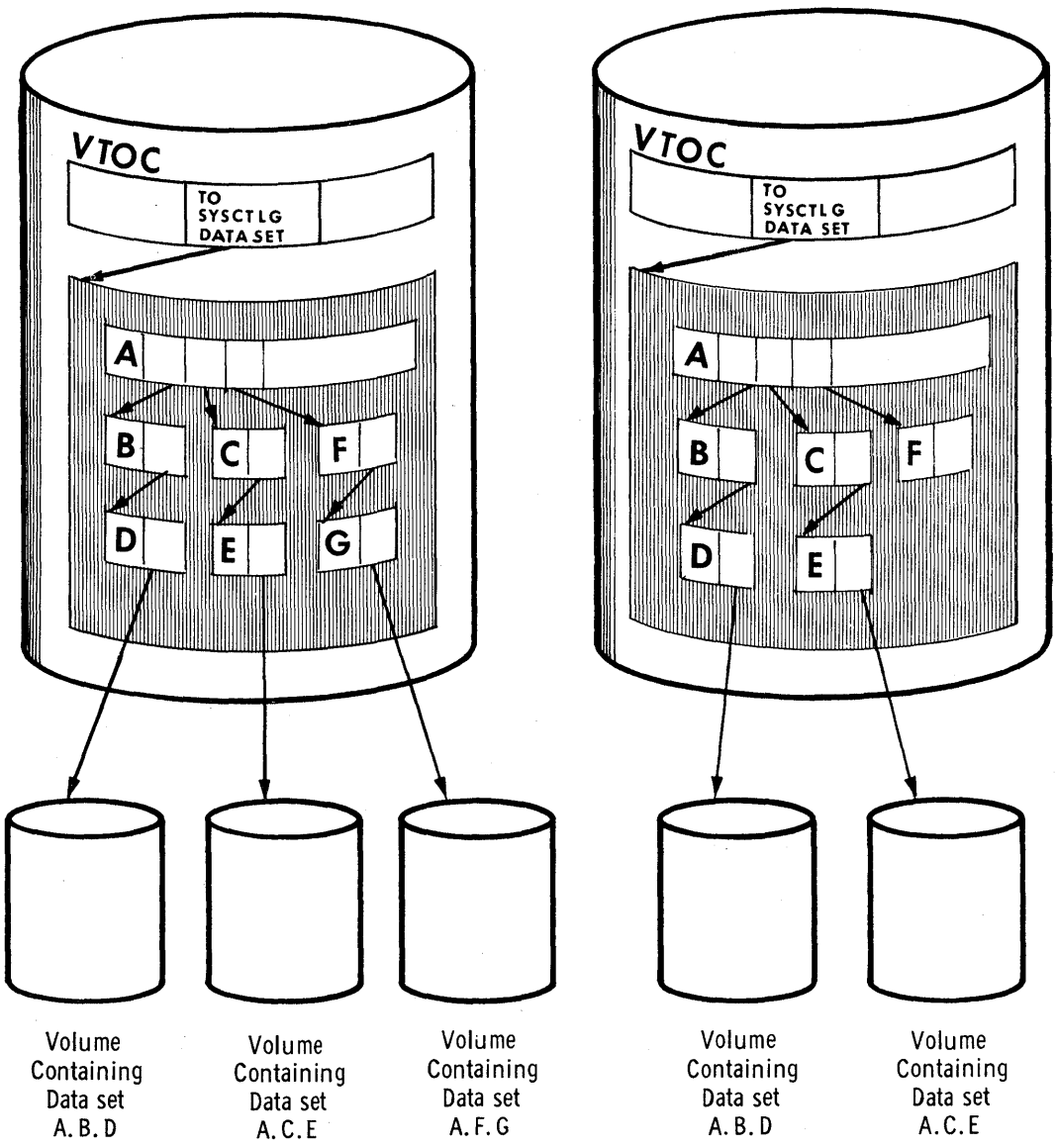


IEHPROGM Figure 1. Cataloging a Data Set

The Uncataloged Data Set: The program uncatalogs a data set by removing the data set name and associated volume information from the lowest-level index of the catalog. The uncatalog function of the program differs from a DISP=(...,UNCATLG) specification in a DD statement in that:

- The DISP=(...,UNCATLG) specification cannot remove an entry from the SYSCTLG data set on a volume other than the system residence volume unless the two volumes are properly connected.

IEHPROGM Figure 2 shows how a typical data set (A.F.G) is uncataloged by the program. Prior to the operation, the fully qualified name and associated volume information are represented in the catalog. The uncatalog operation removes the lowest-level entry from the index structure. Note that the structure A.F remains in the catalog.



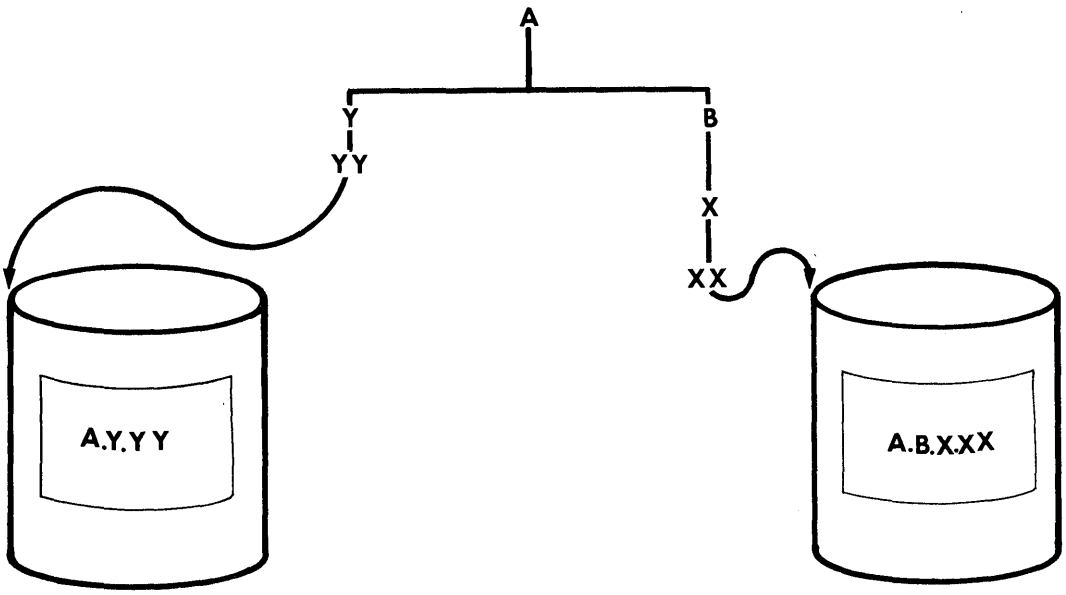
IEHPROGM Figure 2. Uncataloging a Data Set

Building or Deleting an Index

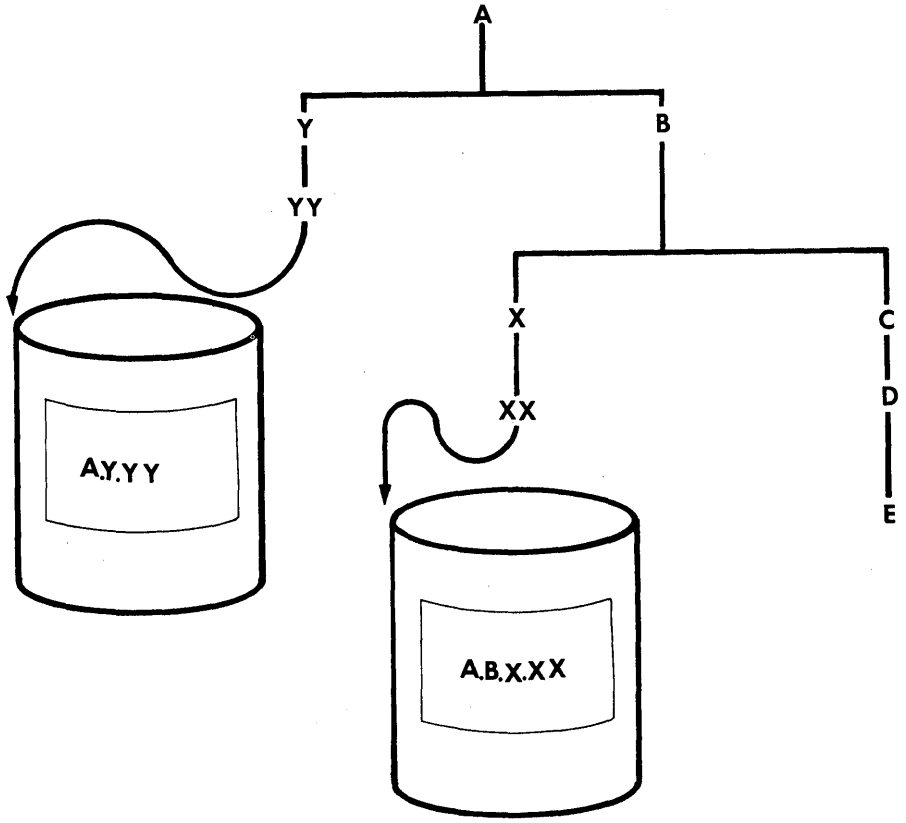
The program can build a new index in the catalog, or it can delete an existing index. In building an index, the program automatically creates as many higher-level indexes as are necessary to complete the specified structure.

The program can delete one or more (as specified) 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.

IEHPROGM Figure 3A shows an index structure representing two cataloged data sets: A.Y.YY and A.B.X.XX. Figure 3B shows how the index structure can be altered by building index A.B.C.D.E . Note in Figure 3A that index levels C and D do not exist prior to the build operation. These levels are automatically created when the level E index is built.



IEHPROGM Figure 3A. Index Structure Prior to Build Operation



IEHPROGM Figure 3B. Index Structure After Build Operation

Note: 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:

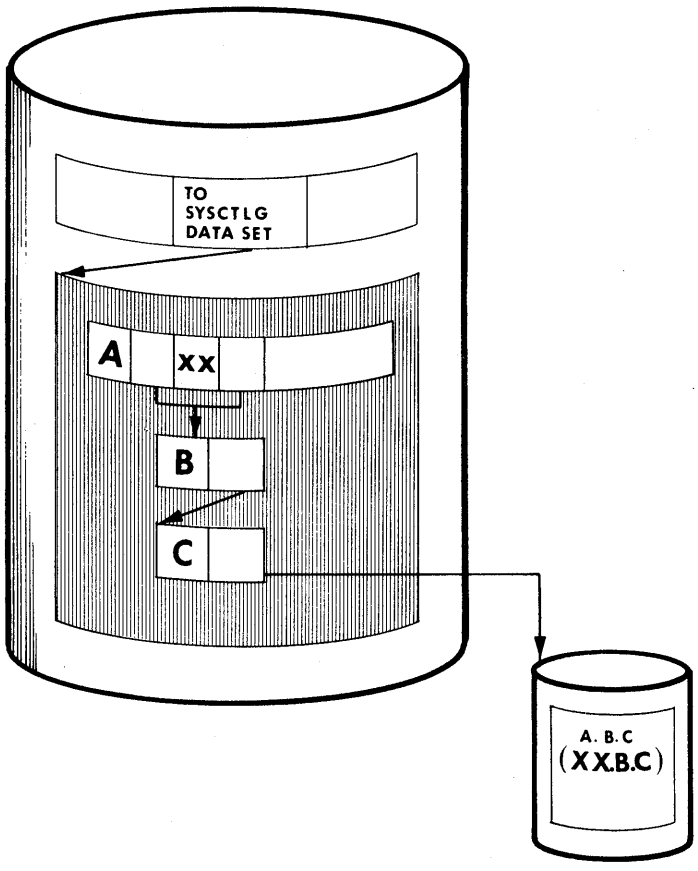
A.B.C.D.E
A.B.C.D
A.B.C

in that order. The B level index cannot be deleted because data set A.B.X.XX and the X level index are dependent upon the B level index.

Building or Deleting an Index Alias

The program can assign an alternative (alias) name to the highest-level index of a catalog, or it can delete an alias name previously assigned.

In IEHPROGM Figure 4, alias name XX has been assigned to index A (a high-level index). The cataloged data set A.B.C can now be referred to as A.B.C or XX.B.C



IEHPROGM Figure 4. Building an Index Alias

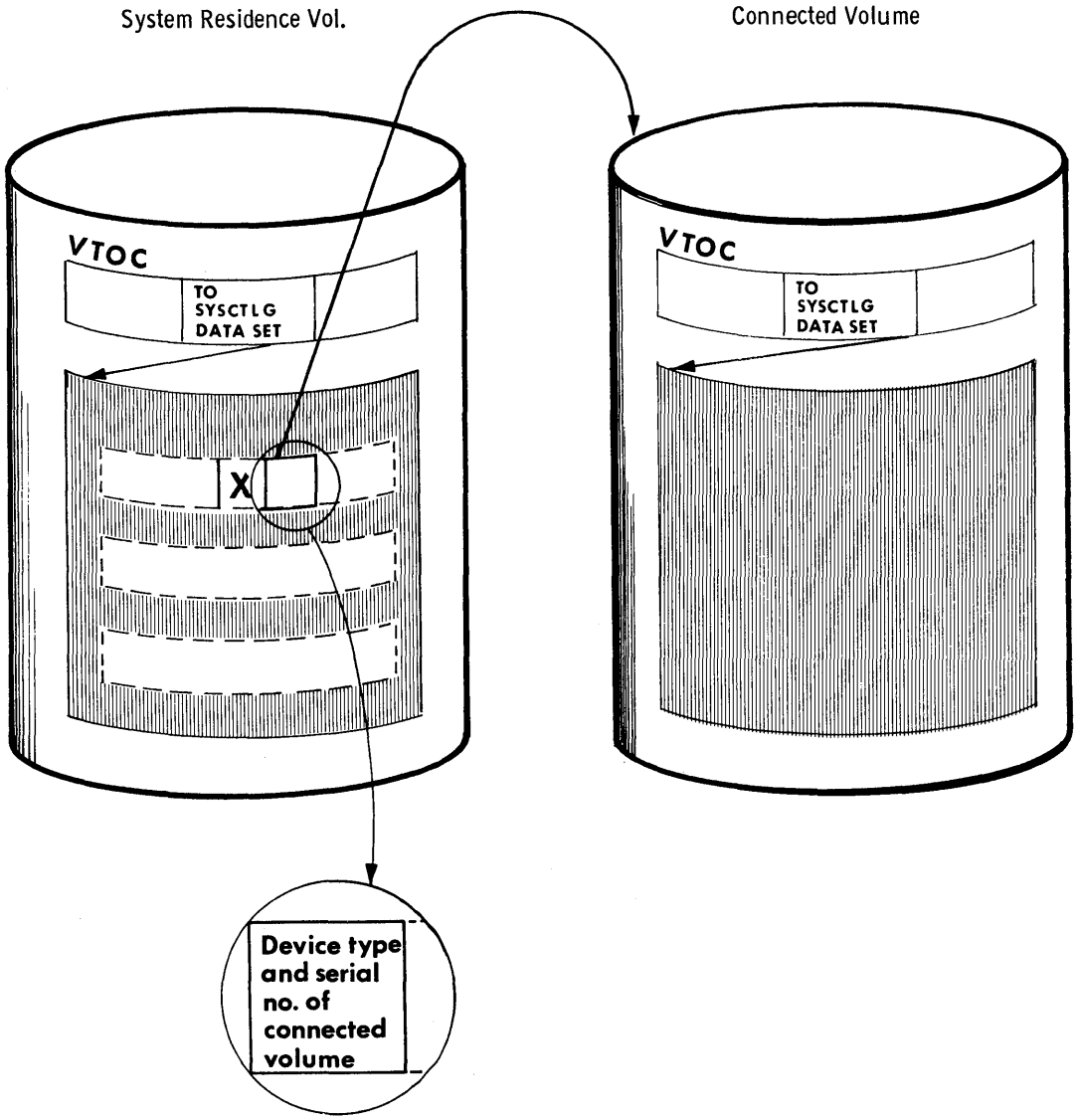
Connecting or Releasing Two Volumes

The program can "connect" a volume to a second volume by placing an entry (containing an index name and the volume serial number and device type of the second volume) into a high-level index on the first volume. The program can subsequently "release" the volumes by removing the entry from the high-level index. The main reasons for connecting two volumes are to permit:

- The catalog (SYSCTLG data set) to be extended to a second volume, if necessary.
- The use of normal JCL to process (retrieve, uncatalog, etc.) data sets cataloged on the second volume (assuming that the first volume is the system residence volume).

If the SYSCTLG data set is extended to a second volume, it must be identified on that volume.

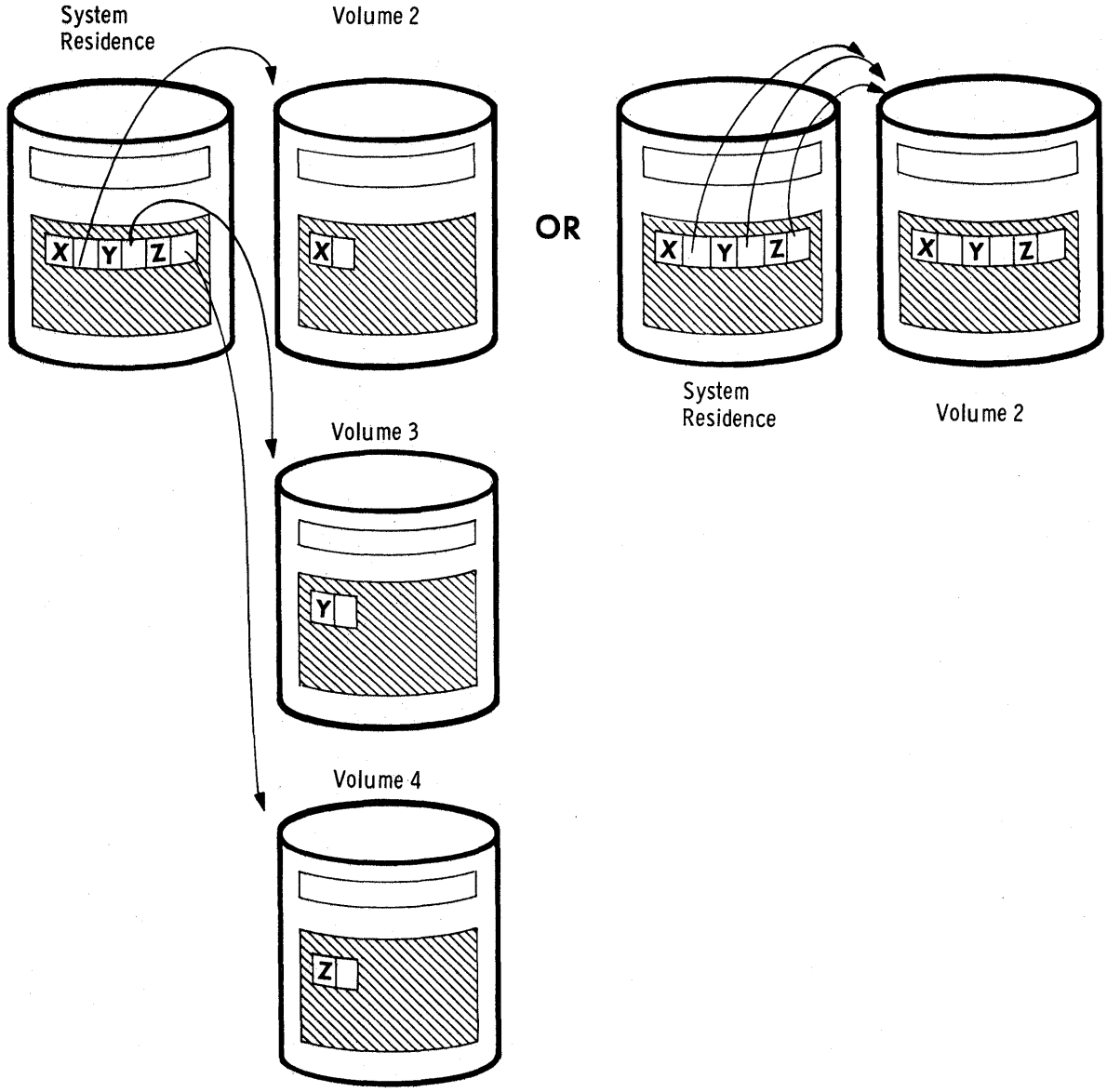
IEHPROGM Figure 5 shows how the system residence volume can be connected to a second volume. Any subsequent index search for index X on the system residence volume is carried to the second volume.



IEHPROGM Figure 5. Connecting a Volume to a Second Volume

Note: The index name of each high-level index existing on the second volume must be present in the first volume; i.e., a connect operation should be performed whenever a new high-level index is placed on the second volume.

Volumes can be connected as shown in IEHPROGM Figure 6. All volumes are accessible (through high-level indexes X, Y, and Z) to the operating system.



IEHPROGM Figure 6. Connecting Two Volumes

Building an Index for a Generation Data Group

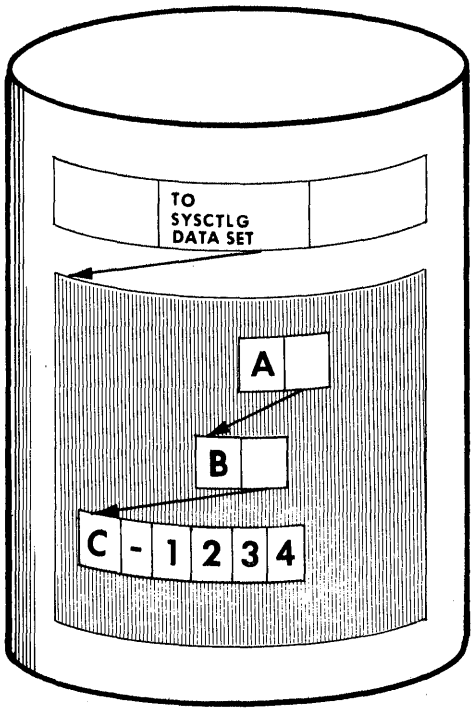
The program can build an index structure for a generation data group, and can control the action to be taken should the index overflow.

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.

IEHPROGM Figure 7 shows the index structure created for a typical generation data group (A.B.C). In this example, provision is made for up to 5 subsequent entries in the lowest-level index.

Note: Before a generation data group can be cataloged as such, a generation data group index must exist. Otherwise, a generation data set is cataloged as an individual data set, rather than as a generation.



IEHPROGM Figure 7. Building a Generation Data Group Index

When creating and cataloging a generation data set, the user can provide necessary DCB information in one of two ways: (1) by creating a model data set control block (DSCB) on the volume on which the catalog resides (prior to creating the generation data set) or (2) by specifying DCB=(dsname) in the DD statment that creates and catalogs the generation data set.

- A model DSCB is created with a DD statement that requests a space allocation of zero tracks and includes applicable DCB information.
- A DCB=(dsname) parameter refers to a cataloged data set that has DCB specifications identical to those desired for the generation data set.

Inputs and Outputs

The input to the IEHPROGM program is a control data set (containing utility control statements) used to control the functions of the program and to indicate those data sets or volumes that are to be modified.

The primary output or result of executing the IEHPROGM program is a modified object data set or volume(s).

A message data set is created to list error messages, if any.

ADDITIONAL OUTPUTS

The program provides a return code to indicate the results of program execution. The return codes and their interpretations are as follows:

00 -- successful completion.
04 -- a syntax error has been found in the name field of the control card. Processing is continued.
08 -- a request for a specific operation has been ignored because of an invalid control statement or an otherwise invalid request. The operation is not performed.
12 -- an I/O error has been detected when trying to read or write from or onto SYSPRINT, SYSIN or the VTOC.
16 -- an unrecoverable error has occurred. The job step is terminated.

Control

The IEHPROGM utility program 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.

Utility control statements are used to:

- Control the functions of the program.
- Define those data sets or volumes that are to be modified.

JOB CONTROL STATEMENTS

IEHPROGM Table 1 shows the job control statements necessary to execute or invoke the IEHPROGM program.

IEHPROGM Table 1. Job Control Statements for the IEHPROGM Program

Statement	Usage
JOB Statement	This statement initiates the job.
EXEC Statement	This statement specifies the program name (PGM=IEHPROGM) or, if the job control statements for the IEHPROGM reside in a procedure library, the procedure name.
SYSPRINT DD Statement	This statement 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.
//anyname1* DD UNIT=xxxx,VOLUME=SER=xxxxxx,DISP=OLD This DD statement defines a permanently mounted volume. One 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.) In this 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. Note: This statement is required only if a permanently mounted residence volume, this statement is required. CATLG statement specifies that an index search begin on the system	

//anyname2** DD VOLUME=SER=xxxxxx,UNIT=xxxx,DISP=OLD
 	
OR
 	
//anyname2** DD VOLUME=(PRIVATE,...),UNIT=(xxxx,,DEFER),DISP=OLD
 	
This statement defines a mountable device type. One statement must be included for each mountable device to be used in the job step.
 	
Note: The second example can be used to specify deferred mounting when a large number of magnetic tapes and direct access volumes are to be processed in one application of the program.	

SYSIN DD Statement	This statement defines the control data set. The data set, which contains utility control statements, normally follows the job control statements in the input stream; however, it can alternatively be defined as a member of a procedure library.
*This DD statement is arbitrarily assigned the ddname DD1 in the IEHPROGM examples (see the IEHPROGM examples). **This DD statement is arbitrarily assigned the ddname DD2 in the IEHPROGM examples (see the IEHPROGM examples). DD statements defining additional mountable devices are assigned names DD3, DD4, ... etc. The blocksize for the SYSPRINT (message) data set must be a multiple of 121. The blocksize for the SYSIN (control) data set must be a multiple of 80. Any blocking factor can be specified for these blocksizes.	

(Part 1 of 2)

IEHPROGM Table 1. Job Control Statements for the IEHPROGM Program (Part 2 of 2)

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. Since the IEHPROGM program 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.)
When the IEHPROGM program is dynamically invoked in a job step containing a program other than IEHPROGM, the DD statements defining mountable devices for the IEHPROGM program must be included in the job stream prior to DD statements defining data sets required by the other program.
For MVT applications, DD statements defining mountable devices must appear in the same order in the input stream as the utility control statements that refer to volumes mounted on those devices.
Refer to Appendix D for instructions on defining mountable devices.

UTILITY CONTROL STATEMENTS

Combinations of the following utility control statements are used to control the functions of the program.

- The SCRATCH statement.
- The RENAME statement.
- The CATLG (catalog) statement.
- The UNCATLG (uncatalog) statement.
- The BLDX (build index) statement.
- The DLTX (delete index) statement.
- The BLDA (build index alias) statement.
- The DLTA (delete index alias) statement.
- The CONNECT statement.
- The RELEASE (disconnect) statement.
- The BLDG (build generation data group index) statement.

The 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.

Name	Operation	Operand
[name]	SCRATCH	{DSNAME=name {VTOC}
		VOL=device=list
		[PURGE]
		[MEMBER=name]
		[SYS]

DSNAME=name
specifies the fully qualified name of either the data set to be scratched, or the partitioned data set that contains the member to be scratched.

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 a VTOC specification is modified if the VTOC keyword is used with the PURGE keyword and/or the SYS keyword.

VOL=device=list
specifies the volume or volumes that contain the data set or sets to be scratched. If VTOC is specified, VOL cannot specify more than one volume.

Caution should be used when specifying SCRATCH VTOC if VOL specifies the system residence volume.

PURGE
requests that each data set specified by DSNAME or VTOC be scratched, even if its expiration date has not elapsed.

If PURGE is omitted, the specified data sets are scratched only if their expiration dates have elapsed.

MEMBER=name
specifies a member name or alias name (of a member) to be removed from the directory of a partitioned data set.

If MEMBER=name is omitted, the entire data set or volume of data sets is scratched.

SYS
(used only with SCRATCH VTOC or SCRATCH VTOC,PURGE operations) requests that only those data sets whose names begin with AAAAAAAA.AAAAAAAA.AAAAAAAA.AAAAAAAA. and/or SYSnnnnn.Txxxxxx.RPxxx.jobname.ddname be scratched. These are names assigned to data sets by the operating system.

If the data set name begins with SYS:

- nnnnn is the date
- Txxxxxx is the time of day
- R is 'reader' (in position 18)
- S is 'system' (in position 18)
- P is PCP (in position 19)
- F is MFT (in position 19)
- V is MVT (in position 19)
- xxx is the data set number

Data sets whose expiration dates have not elapsed are not scratched unless the PURGE keyword is specified.

Caution: The action of the SYS keyword is applied to all data sets whose data set names begin with the characters SYSnnnnn.T and contain the character P, F, or V in the nineteenth position. When executing a SCRATCH operation in a multiprogramming environment, care should be taken to insure that the data set or volume is not being used by a program executing concurrently.

The RENAME Statement

The RENAME statement is used to change the true name or alias name 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.

Name	Operation	Operand
[name]	RENAME	DSNAME=name VOL=device=list NEWNAME=name [MEMBER=name]

DSNAME=name
: specifies the fully qualified name of the data set to be renamed, or the data set that contains the member to be renamed.

VOL=device=list
: specifies the volume or volumes that contain the data set or member whose name is to be changed. If MEMBER=name is specified, VOL cannot specify more than one volume.

NEWNAME=name
: specifies the new fully qualified name for the data set, or the new member or alias name.

MEMBER=name
: specifies the member name or alias name for a member (in the named data set) that is to be renamed.

  If MEMBER=name is omitted, the specified data set name is changed.

The CATLG Statement

The CATLG statement is used to generate an entry in the index of a catalog. If additional levels of indexes are required in the catalog, this function automatically creates them.

Name	Operation	Operand
[name]	CATLG	DSNAME=name VOL=device=list [CVOL=device=serial]

DSNAME=name
: specifies the fully qualified name of the data set to be cataloged. The qualified name must not exceed 44 characters including delimiters (periods).

VOL=device=list
: specifies the volume or volumes that contain the data set to be cataloged. For either a sequential data set or an indexed sequential data set, 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 an indexed sequential data set, all serial numbers specified in VOL=device=list must represent the same device type.

Note: When "device" is represented by a group name (e.g., SYSDA), instead of a generic name (e.g., 2311 or 2400), 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. If the group name is generated for one device type it may be used as a substitute for a generic name. The generic name should be used if the group name was generated for multi 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.

CVOL=device=serial
specifies the device type and volume serial number of the control volume on which the catalog search for the index is to begin.

If CVOL is omitted, the system residence volume is assumed.

Notes: If the data set to be cataloged is an indexed sequential data set, all volume serial numbers specified in VOL=device=list must represent the same device type.

When cataloging data sets residing on tape, the programmer must 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 drive (2400-4), the data set can be cataloged with a device specification of 2400 for an 800 bpi tape or 2400-3 for a 1600 bpi 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 drive.

The UNCATLG Statement

The UNCATLG statement is used to remove an entry from the lowest level index of the catalog. This function does not delete higher level indexes from the index structure.

Name	Operation	Operand
[name]	UNCATLG	DSNAME=name [CVOL=device=serial]

DSNAME=name
specifies the fully qualified name of the data set to be uncataloged.

CVOL=device=serial
specifies the device type and volume serial number of the control volume at which the search for the catalog entry is to begin.

If CVOL is omitted, the system residence volume is assumed.

The BLDX (Build Index) Statement

The BLDX statement is used to create a new index in the catalog. If the creation of an index requires that higher level indexes be created, this function automatically creates them.

Name	Operation	Operand
[name]	BLDX	INDEX=name [CVOL=device=serial]

INDEX=name
: specifies the qualified name of the index to be created. The qualified name must not exceed 44 characters, including periods.

CVOL=device=serial
: specifies the device type and volume serial number of the control volume on which the search for the index is to begin.

  If CVOL is omitted, the system residence volume is assumed.

The DLTX (Delete Index) Statement

The DLTX statement is used to remove an index from the catalog. Only an index that has no entries can be removed.

Name	Operation	Operand
[name]	DLTX	INDEX=name [CVOL=device=serial]

INDEX=name
: specifies the qualified name of the index to be deleted.

CVOL=device=serial
: specifies the device type and volume serial number of the control volume on which the search for the index is to begin.

  If CVOL is omitted, the system residence volume is assumed.

Note: Since 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 , use:

```
DLTX        INDEX=A.B.C
DLTX        INDEX=A.B
DLTX        INDEX=A
```

in that order.

The BLDA (Build Index Alias) Statement

The BLDA statement is used to assign an alias name to an index at the highest level of the catalog.

Name	Operation	Operand
[name]	BLDA	INDEX=name ALIAS=name [CVOL=device=serial]

INDEX=name
specifies the unqualified index name to which an alias name is to be assigned.

ALIAS=name
specifies an unqualified name to be assigned as the alias name.

CVOL=device=serial
specifies the device type and volume serial number of the control volume on which the catalog entry is to be made.

If CVOL is omitted, the system residence volume is assumed.

The DLTA (Delete Index Alias) Statement

The DLTA statement is used to delete an alias name previously assigned to an index at the highest level of the catalog.

Name	Operation	Operand
[name]	DLTA	ALIAS=name [CVOL=device=serial]

ALIAS=name
specifies the unqualified index alias to be deleted.

CVOL=device=serial
specifies the device type and volume serial number of the control volume containing the catalog entry to be deleted.

If CVOL is omitted, the system residence volume is assumed.

The CONNECT Statement

The CONNECT statement is used to place an entry into an index at the highest level of the catalog. The entry identifies a second volume 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 volume.

Note: This function does not create an index on the second volume.

Name	Operation	Operand
[name]	CONNECT	INDEX=name VOL=device=serial [CVOL=device=serial]

INDEX=name
specifies the index name to be entered in the high-level index on the first volume.

VOL=device=serial
specifies the device type and volume serial number of the second volume. This information is placed in the high-level index on the first volume.

CVOL=device=serial
specifies the device type and serial number of the first volume.

If CVOL is omitted, the system residence volume is assumed to be the first volume.

Note: The CONNECT statement does not create a SYSCTLG data set on the connected volume. Prior to 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=(TRK,xx),VOLUME=SER=xxxxxx
```

Note: If a job may require 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 will direct the operator to mount an auxiliary control volume if it is needed. However, the user must ensure that the auxiliary control volume is connected to the system residence volume by means of the CONNECT verb, as modified for Release 17. If an auxiliary control volume was connected before Release 17, the user must first release the auxiliary control volume for all high-level indexes on the system residence volume which point to that volume, and then use the current CONNECT verb to reconnect the auxiliary control volume with the system residence volume. (See IEHPROGM Example 8 for a sample program which performs this releasing and reconnecting of an auxiliary control volume.)

The RELEASE (Disconnect) Statement

The RELEASE statement is used to remove an entry from the high-level index of a volume. This effectively disconnects a second volume from the first volume.

Note: This function does not delete an index from the second volume.

Name	Operation	Operand
[name]	RELEASE	INDEX=name [CVOL=device=serial]

INDEX=name
specifies the index name to be removed from the high-level index of the first volume.

CVOL=device=serial
specifies the device type and volume serial number of the first volume.

If CVOL is omitted, the system residence volume is assumed to be first volume.

The BLDG (Build Generation Data Group Index) Statement

The BLDG statement is used to build an index for a generation data group, and to establish the action to be taken should the index overflow.

Name	Operation	Operand
[name]	BLDG	INDEX=name
		ENTRIES=n
		[CVOL=device=serial]
		[EMPTY]
		[DELETE]

INDEX=name
specifies the name of the generation data group index.

ENTRIES=n
specifies the number of entries to be contained in the generation data group index; n must not exceed 255.

CVOL=device=serial
specifies the device type and volume serial number of the volume on which the catalog search for the index is to begin.

If CVOL is omitted, the system residence volume is assumed.

EMPTY
specifies that all entries be removed from the generation data group index when it overflows. This effectively uncatalogs all of the generation data sets.

If EMPTY is omitted, the oldest entry is removed when the generation data group index overflows.

DELETE
specifies that generation data sets are to be scratched after their entries are removed from the index.

If DELETE is omitted, the data sets are not scratched.

IEHPROGM Examples

The following examples illustrate some of the applications for the IEHPROGM system utility program.

Note: In each of the following IEHPROGM examples, the EXEC statement and the SYSPRINT DD statement can be replaced with the following job control statement:

```
//        EXEC PROC=MOD
```

The EXEC statement invokes the following cataloged procedure, which is supplied by IBM.

```
//MOD      EXEC PGM=IEHPROGM,REGION=44K                              00000000
//DDSRV     DD  VOLUME=REF=SYS1.SVCLIB,DISP=OLD                      00000010
//SYSPRINT DD  SYSOUT=A                                              00000020
```

IEHPROGM Example 1

Operation	Number of Mountable Volumes to be Processed	Comments
SCRATCH VTOC	1 VOLUME -- 2311 DISK	1. The system residence volume is not referred to in this job step.

In this example, data sets are to be scratched from a volume table of contents of a mountable volume. Since the system residence volume is not referred to, no DD1 DD statement is necessary in the job stream.

- The SCRATCH Statement: indicates that all data sets (including those beginning with AAAAAAAA.AAAAAAAA.AAAAAAAA.AAAAAAAA) whose expiration dates have expired are to be scratched from the specified volume.

```
//SCRVTOC  JOB  09#550,BROWN
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD2      DD   UNIT=2311,VOLUME=SER=231100,DISP=OLD
//SYSIN    DD   *
      SCRATCH   VTOC,VOL=2311=231100
/*
```

IEHPROGM Example 1. Scratching Data Sets From a Volume

IEHPROGM Example 2

Operation	Number of Mountable Volumes to be Processed	Comments
SCRATCH two data sets UNCATLG two data sets DLTX	2 VOLUMES -- 2311 DISK	1. The system residence volume is on a 2301 Drum device.

In this example two data sets are to be scratched: SET1 on volume 231100 and A.B.C.D.E on volume 231101. Both data sets are to be uncataloged. In addition, index structure A.B.C.D is to be deleted from the SYSCTLG data set.

Since the system residence volume is referred to (through use of the UNCATLG and DLTX statements), a DD1 statement is included in the input stream.

```
//SCRDSETS JOB  09#550,BROWN
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=2301,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=(2311,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(231100))
//SYSIN    DD   *
      SCRATCH   DSNAME=SET1,VOL=2311=231100
      UNCATLG   DSNAME=SET1
      SCRATCH   DSNAME=A.B.C.D.E,VOL=2311=231101
      UNCATLG   DSNAME=A.B.C.D.E
         DLTX   INDEX=A.B.C.D
         DLTX   INDEX=A.B.C
         DLTX   INDEX=A.B
         DLTX   INDEX=A
/*
```

IEHPROGM Example 2. Scratching and Uncataloging Data Sets -- Deleting an Index From the Catalog

IEHPROGM Example 3

Operation	Number of Mountable Volumes to be Processed	Comments
RENAME data set UNCATLG old dsname DLTX old index structure CATLG new dsname	2 VOLUMES -- 2311 DISK	1. The system residence volume is on a 2301 Drum device. 2. The object data set exists on two mountable volumes.

In this example, the name of a data set is to be changed on two mountable volumes. The old data set name and index structure are to be removed from the catalog (UNCATLG and DLTX statements). The data set is then to be cataloged under its new data set name.

Since the system residence volume is referred to (through use of the UNCATLG, DLTX, and CATLG statements), a DD1 statement is included in the input stream.

```
//RENAMEDS JOB  09#550,BROWN
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1      DD   VOLUME=SER=111111,UNIT=2301,DISP=CLD
//DD2      DD   UNIT=(2311,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(231100,231101))
//SYSIN    DD   *
       RENAME   DSNAME=A.B.C,VOL=2311=(231100,231101),NEWNAME=NEWSET
     UNCATLG    DSNAME=A.B.C
        DLTX    INDEX=A.B
        DLTX    INDEX=A
       CATLG    DSNAME=NEWSET,VOL=2311=(231100,231101)
/*
```

IEHPROGM Example 3. Renaming a Data Set on Multiple Volumes -- Cataloging Data Set Under New Data Set Name

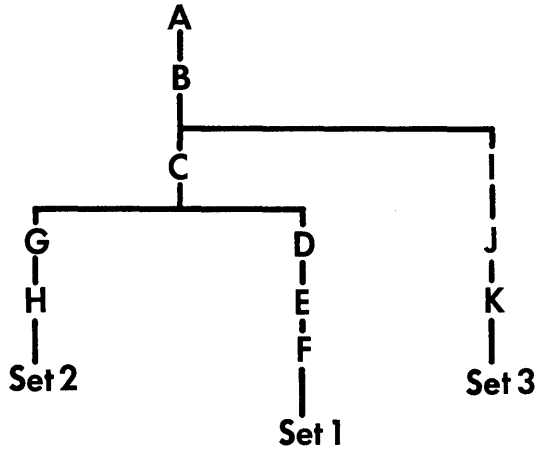
IEHPROGM Example 4

Operation	Comments
UNCATLG data sets DLTX index structures from catalog	1. No mountable volumes are to be processed. 2. The system residence volume is on a 2311 device.

In this example three data sets are to be uncataloged and their supporting index structures deleted from the catalog.

Given the cataloged data sets:

A.B.C.D.E.F.SET1
A.B.C.G.H.SET2
A.B.I.J.K.SET3



The data sets can be uncataloged and their structures deleted in the order shown in the example.

```
//DLTSTRUC JOB  09#550,BROWN
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=2311,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
         DLTX   INDEX=A.B.I.J.K
         DLTX   INDEX=A.B.I.J
         DLTX   INDEX=A.B.I
         DLTX   INDEX=A.B.C.G.H
         DLTX   INDEX=A.B.C.G
         DLTX   INDEX=A.B.C.D.E.F
         DLTX   INDEX=A.B.C.D.E
         DLTX   INDEX=A.B.C.D
         DLTX   INDEX=A.B.C
         DLTX   INDEX=A.B
         DLTX   INDEX=A
/*
```

IEHPROGM Example 4. Deleting an Index Structure

IEHPROGM Example 5

Operation	Number of Mountable Volumes to be Processed	Comments
CONNECT the system residence volume to a second volume CATLG data sets on the second volume	1 VOLUME -- 2311 DISK	1. The SYSCTLG data set has been previously defined on the second volume. 2. The system residence volume is on a 2301 Drum device.

In this example, the system residence volume is to be connected to a second volume. After the connect operation has been performed, any subsequent index search for index level X, Y, or Z will be carried to the second volume.

The CONNECT Statements: identify the second volume. The specified index names, along with the volume identification, are placed on the system residence volume.

The CATLG Statements: catalog three data sets (X.BB.CCC, Y.BB.CC, and Z.BB.XT) on the second volume. Since the volumes are connected prior to the catalog operations, the CVOL keywords are not required in the CATLG statements; they are included merely to bypass the index search on the system residence volume.

```
//CONNECT  JOB  09#550,BROWN
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=2301,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=2311,VOLUME=SER=231100,DISP=OLD
//SYSIN    DD   *
       CONNECT  INDEX=X,VOL=2311=231100
       CONNECT  INDEX=Y,VOL=2311=231100
       CONNECT  INDEX=Z,VOL=2311=231100
         CATLG  DSNAME=X.BB.CCC,VOL=2311=231101,CVOL=2311=231100
         CATLG  DSNAME=Y.BB.CC,VOL=2311=231101,CVOL=2311=231100
         CATLG  DSNAME=Z.BB.XT,VOL=2311=231101,CVOL=2311=231100
/*
```

IEHPROGM Example 5. Extending the Catalog to a Second Volume (CONNECT) and Cataloging Data Sets on That Volume

IEHPROGM Example 6

Operation	Comments
BLDG	1. The system residence volume is on a 2301 Drum device.

In this example, a generation data group index is to be built in the catalog.

The BLDG Statement: specifies the generation data group name A.B.C and makes provision for 10 entries in the index. All entries are to be removed from the index when it overflows.

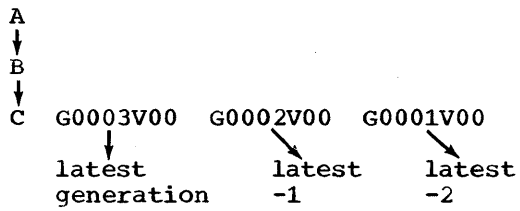
```
//BLDGDGIX JOB  09#550,BROWN
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=2301,VOLUME=SER=111111,DISP=OLD
//SYSIN    DD   *
         BLDG  INDEX=A.B.C,ENTRIES=10,EMPTY
/*
```

IEHPROGM Example 6. Building a Generation Data Group Index

IEHPROGM Example 7

Operation	Comments
BLDG CATLG generation data group index	1. The system residence volume is on a 2301 Drum device.

In this example, a generation data group index is to be built and three data sets are to be cataloged in the index. After the three generation data sets are cataloged, the index structure appears as




```
//CTLGDG   JOB  09#550,BROWN
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=2301,VOLUME=SER=111111,DISP=CLD
//SYSIN    DD   *
          BLDG   INDEX=A.B.C,ENTRIES=20,EMPTY
        CATLG    DSNAME=A.B.C.G0001V00,VOL=2311=231100
        CATLG    DSNAME=A.B.C.G0002V00,VOL=2311=231100
        CATLG    DSNAME=A.B.C.G0003V00,VOL=2311=231100
/*
```

IEHPROGM Example 7. Cataloging Three Generation Data Sets

IEHPROGM Example 8

Operation	Comments
RELEASE	1. The system residence volume is on a 2311 Disk device.
CONNECT	

This example illustrates the use of the RELEASE and CONNECT verbs to disconnect the control volume, 231100, from the system residence catalog for the high-level index A, and reconnect that same control volume for that index. This technique is necessary only if the auxiliary control volume was connected before Release 17.

```
//RECONCT  JOB  09#550,BROWN
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=2311,VOLUME=SER=111111,DISP=OLD
//SYSIN    DD   *
      RELEASE   INDEX=A
      CONNECT   INDEX=A,VOL=2311=231100
/*
```

IEHPROGM Example 8. Releasing and Reconnecting an Auxiliary Control Volume

Program Applications

The IEHMOVE system utility program moves or copies logical collections of IBM System/360 Operating System data.

The program can be used to move or copy:

- A data set residing on one or more volumes (up to 5).
- A group of cataloged data sets.
- A catalog, or portions of a catalog.
- A volume of data sets.

The scope of a basic move or copy operation can be enlarged by:

- 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, the program is unable to successfully move or copy specified data, the program attempts to reorganize the data and place it on the specified output device. The reorganized data is called an "unloaded data set." It 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 onto 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 onto a magnetic tape volume, the data is unloaded onto that volume. The user can re-create the data set simply by moving the unloaded data set onto a direct access volume.

Move Versus Copy: A move operation differs basically from a copy operation in that a move operation scratches source data (from direct access source volumes only), 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.

Allocating Space: 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 the IEHMOVE program, itself, in the IEHMOVE job step. If the source data is unmoveable (i.e., if it contains location dependent code), the user should allocate space on the receiving volume using absolute track allocation to insure that the data set is placed in the same location on the receiving volume as it was on the source volume. Unmoveable data will be moved if space is allocated by the IEHMOVE program, 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 between unlike devices, the programmer should provide secondary allocation to ensure that ample space will be available on the receiving volume.

Results of Moving and Copying Operations: A move or copy operation has one of three results.

1. A moved or copied data set.
2. No action.
3. An unloaded version of the source data set.

These results depend upon the compatibility of the source and receiving volumes with respect to:

- The size of the volumes.
- The data set organization (sequential, partitioned, or direct access).
- The movability of the source data set.
- The allocation of space on the receiving volume.

Note: Two volumes are compatible with respect to size if the source record size does not exceed the receiving track size or, if the output is to be written with track overflow, the receiving volume supports the track overflow feature. (Refer to "Job Control Statements" for notes on the track overflow feature.) When using direct access 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 direct data set, it must be loaded to the same device type from which it originally was unloaded.

IEHMOVE Table 1 shows the results of move and copy operations. The organization of the source data set set is shown along with the characteristics of the receiving device.

IEHMOVE Table 1. Results of Move and Copy Operations

Receiving Volume	Source Data Set Organization		
	Sequential	Partitioned	Direct Access
DIRECT ACCESS (size compatible with source volume)			
• 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
DIRECT ACCESS (size incompatible with source volume)			
• 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
NON-DIRECT ACCESS			
• Movable data	moved or copied	unloaded	unloaded
• Unmovable data	unloaded	unloaded	no action

Notes: 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; i.e., no provision is made for directory space. Direct data sets cannot be previously allocated because they cannot be differentiated from partially used existing direct data sets.

If a move or copy operation is unsuccessful (no action), the source data remains intact.

If a move or copy operation is unsuccessful and space was allocated by the IEHMOVE program, 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.

For example, the IEHMOVE program has moved 104 members of a 105 member partitioned data set. On moving the 105th member an I/O error is encountered.

- If space was allocated by the IEHMOVE program, the entire partitioned data set is scratched from the receiving volume.

- If space was previously allocated, no data is scratched from the receiving volume. In this case, after determining the nature of the error, the user need move only the 105th member into the receiving partitioned data set.

Default Allocation: If a sequential data set is to be moved or copied and space attributes are not available either through preallocation or from the data set control block belonging to the source data set, the IEHMOVE program makes a default space allocation. The default allocation consists of a primary allocation of 72,500 bytes of storage and up to 14 secondary allocations of 36,250 bytes each.

Password Protection: When moving or copying a data set group (DSGROUP) - or a volume - containing password protected data sets, the user must provide the password each time a data set is opened or scratched. For each COPY operation, the password must be provided twice. For each MOVE operation, the password must be provided three times.

Moving or Copying User Labels

The IEHMOVE program will always move or copy any user labels associated with an input data set. This is not an optional feature of the IEHMOVE program.

The IEHMOVE program will not take exits to a user's label processing routines. The EXITS and LABELS statements discussed in "Appendix F: Utility Program Handling of User Labels," are not valid for the IEHMOVE program.

Warning: Under the unusual condition in which a data set which has only user trailer labels is to be moved from a tape volume to a direct access volume, the user must preallocate space on the direct access volume to insure that a track will be reserved to receive the user labels.

Moving or Copying BDAM Data Sets

When moving or copying a BDAM data set(s) to like devices, both relative track and relative block integrity are maintained.

When moving or copying a BDAM data set(s) to a larger device, relative track integrity is maintained for data sets with variable or undefined record formats; and relative block integrity are maintained for data sets with fixed length record formats.

When moving or copying a BDAM data set(s) 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.

Reblocking

Reblocking Data Sets with Fixed or Variable Record Format: Data sets with fixed or variable length records can be reblocked to a different blocksize by preallocating the desired block size on the receiving volume. No reblocking can be performed when loading or unloading.

Reblocking Data Sets with Undefined Record Format: When moving or copying data sets with undefined record format and reblocking to a smaller blocksize (i.e., transferring records to a device with a track capacity smaller than the track capacity of the original device) the user must make the blocksize 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

The IEHMOVE program can move or copy a data set as follows:

Sequential data set -- move from one direct access or non-direct access volume to another (or onto the same volume provided that it is a direct access volume).
copy from one direct access or non-direct access volume to another (or onto the same volume provided that the data set name is changed and the receiving volume is a direct access volume).

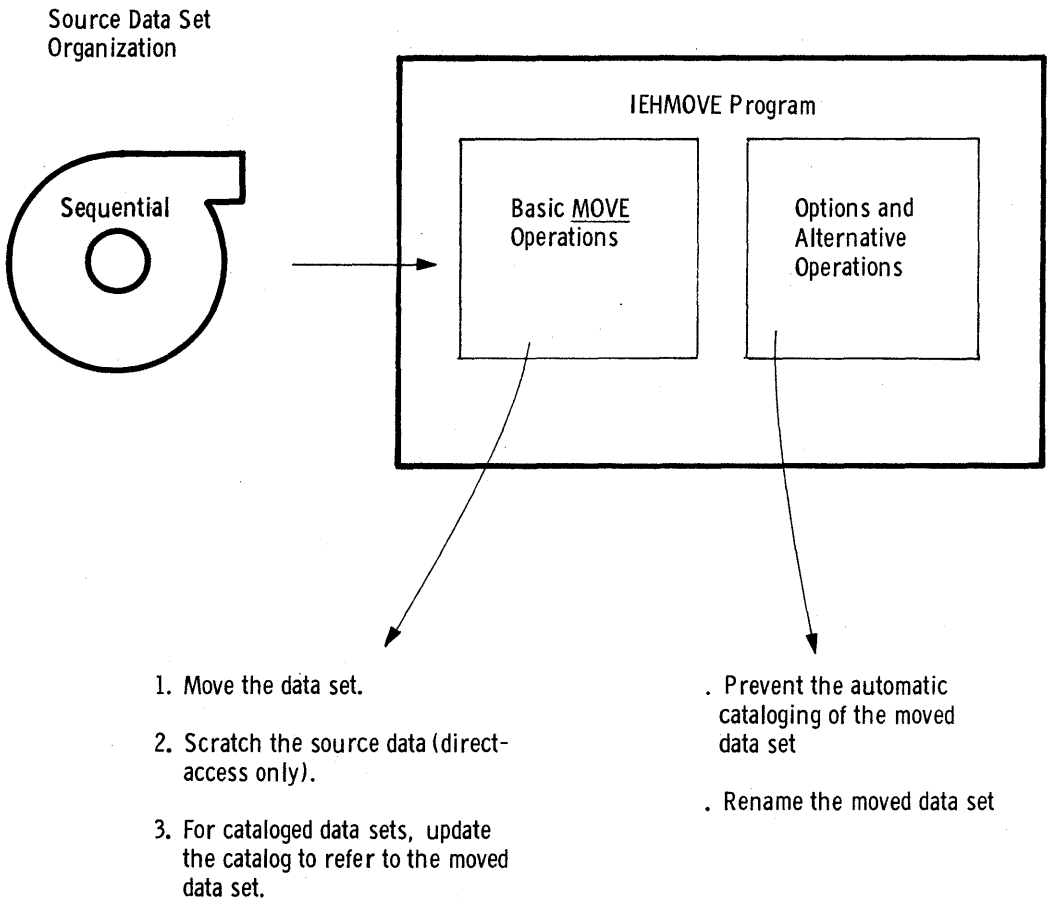
Partitioned data set -- move from one direct access volume to another (or onto the same volume).
copy from one direct access volume to another (or onto the same volume provided the data set name is changed).

Direct data set -- move or copy from one direct access volume to another provided that the receiving device is the same device type or a larger device type, and the records do not exceed 32K.

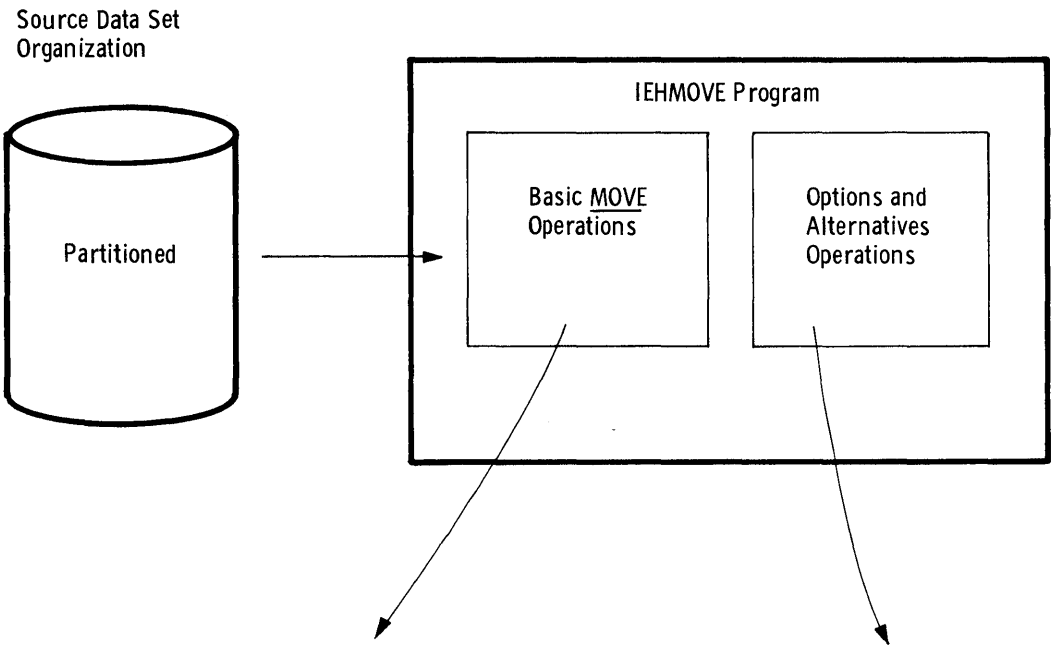
In addition, the program can move or copy multivolume data sets. To move or copy a data set that resides on more than one tape volume, you must specify, in the list field of the FROM=device=list parameter on the utility control statement, the volume serial numbers of all the tape volumes and the sequence numbers of the data set on the tape volumes. (You must specify the sequence number even if the data set to be moved or copied is the only data set on a volume. For example, if a multivolume data set on 2400 tapes resides on the 2nd file of tape 001234 and on the 1st file of tape 001235, you must specify FROM=2400=(001234,2,001235,1).) To move or copy a data set onto more than one tape volume, you must specify the volume serial numbers of all the receiving tape volumes in the list field of the TO=device=list parameter on the utility control statement.

Note: A data set with the unmovable attribute can be moved or copied from one direct access volume to another or onto the same volume provided that space has been preallocated on the receiving volume. You must change the name of a data set to move or copy it onto the same volume. SVCLIB can be moved or copied to another location on the system residence volume, provided that space is available and that space has been preallocated on that volume. The IEHPROGM utility must be employed immediately after such a move operation to rename the moved version SYS1.SVCLIB. After such a copy operation, the IEHPROGM program must be used to scratch the old version and then to rename the copied version. In either case, the IEHIOSUP program must be used immediately after the IEHPROGM step to update the new version of SVCLIB.

IEHMOVE Figures 1 and 2 show basic move operations for sequential and partitioned data sets, respectively. Options and alternative operations that can be specified by the user are also shown.



IEHMOVE Figure 1. Moving a Sequential Data Set



1. Move the data set.
2. Scratch the source data (direct-access only).
3. For cataloged data sets, update the catalog to refer to the moved data set.

. Prevent the automatic cataloging of the moved data set.

. Rename the moved data set.

. Re-allocate directory space. (This is not possible if the space previously allocated to the data set is partially used.)

. Perform a merge operation using members from two or more data sets.

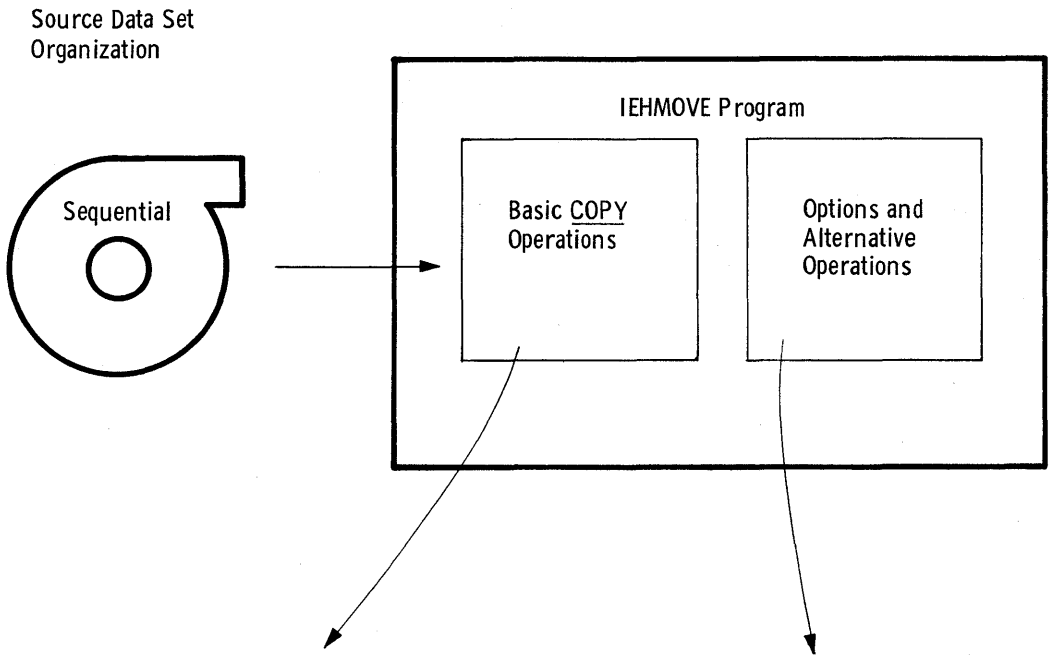
. Move only selected members.

. Replace members

. Unload the data set.

IEHMOVE Figure 2. Moving a Partitioned Data Set

IEHMOVE Figures 3 and 4 show basic copy operations for sequential and partitioned data sets, respectively. Options and alternative operations that can be specified by the user are also shown.



1. Copy the data set.

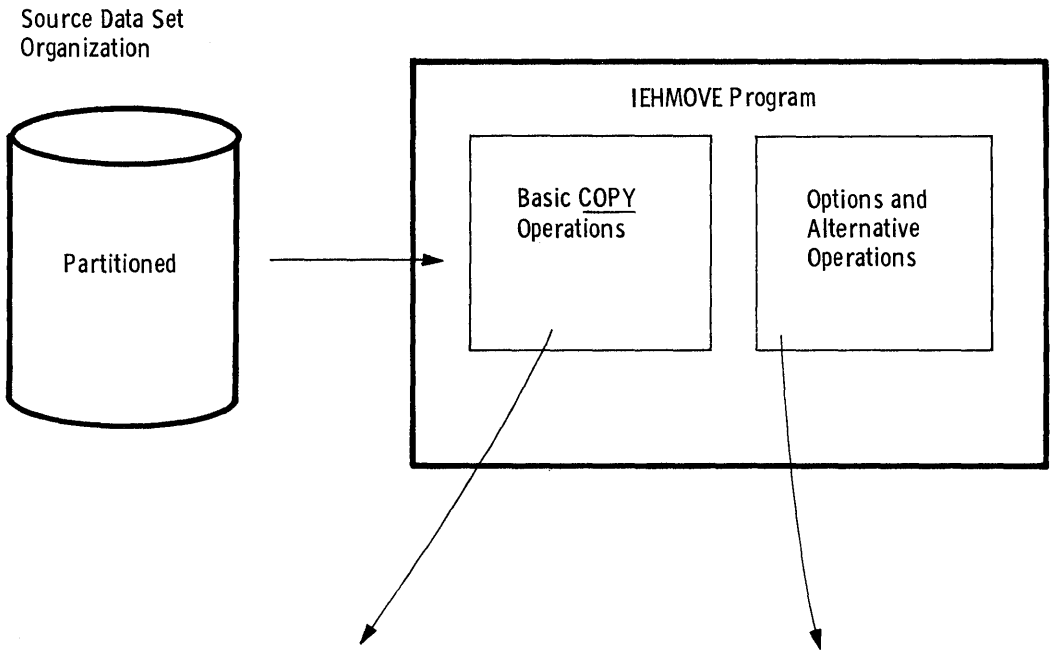
Note: The source data is not scratched. The catalog is not updated to refer to a copied data set.

. Uncatalog the source data set.

. Catalog the copied data set on the receiving volume.

. Rename the copied data set.

IEHMOVE Figure 3. Copying a Sequential Data Set



1. Copy the data set.

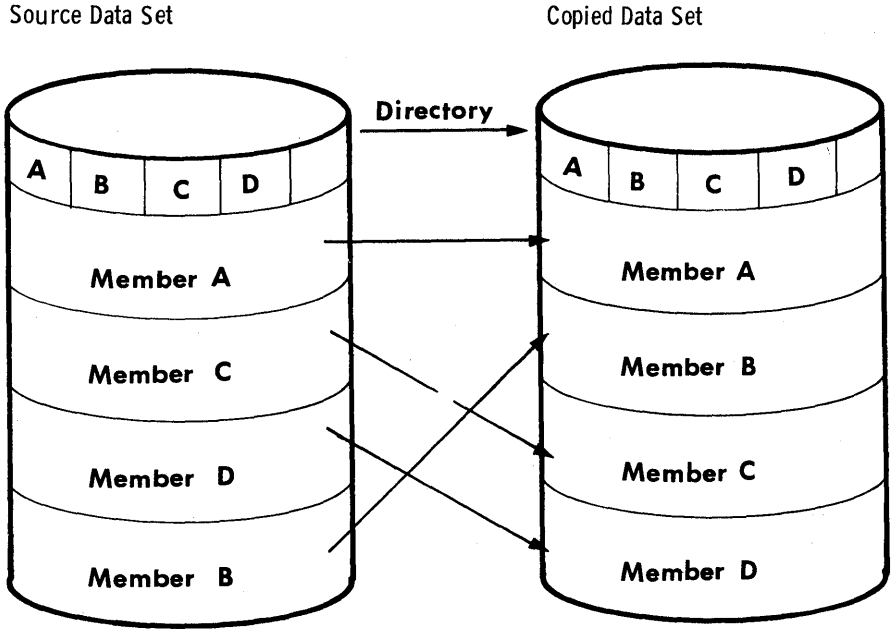
Note: The source data is not scratched. The catalog is not updated to refer to a copied data set.

- Uncatalog the source data set.
- Catalog the copied data set on the receiving volume.
- Rename the copied data set.
- Re-allocate directory space. (This is not possible if the space previously allocated to the data set 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.

IEHMOVE Figure 4. Copying a Partitioned Data Set

Order of Moved or Copied Members: The IEHMOVE program 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.

IEHMOVE Figure 5 shows a copied partitioned data set. Note that the members are reordered. The IEBCOPY data set utility program (see) can be used to copy those data sets whose members are not to be collated.

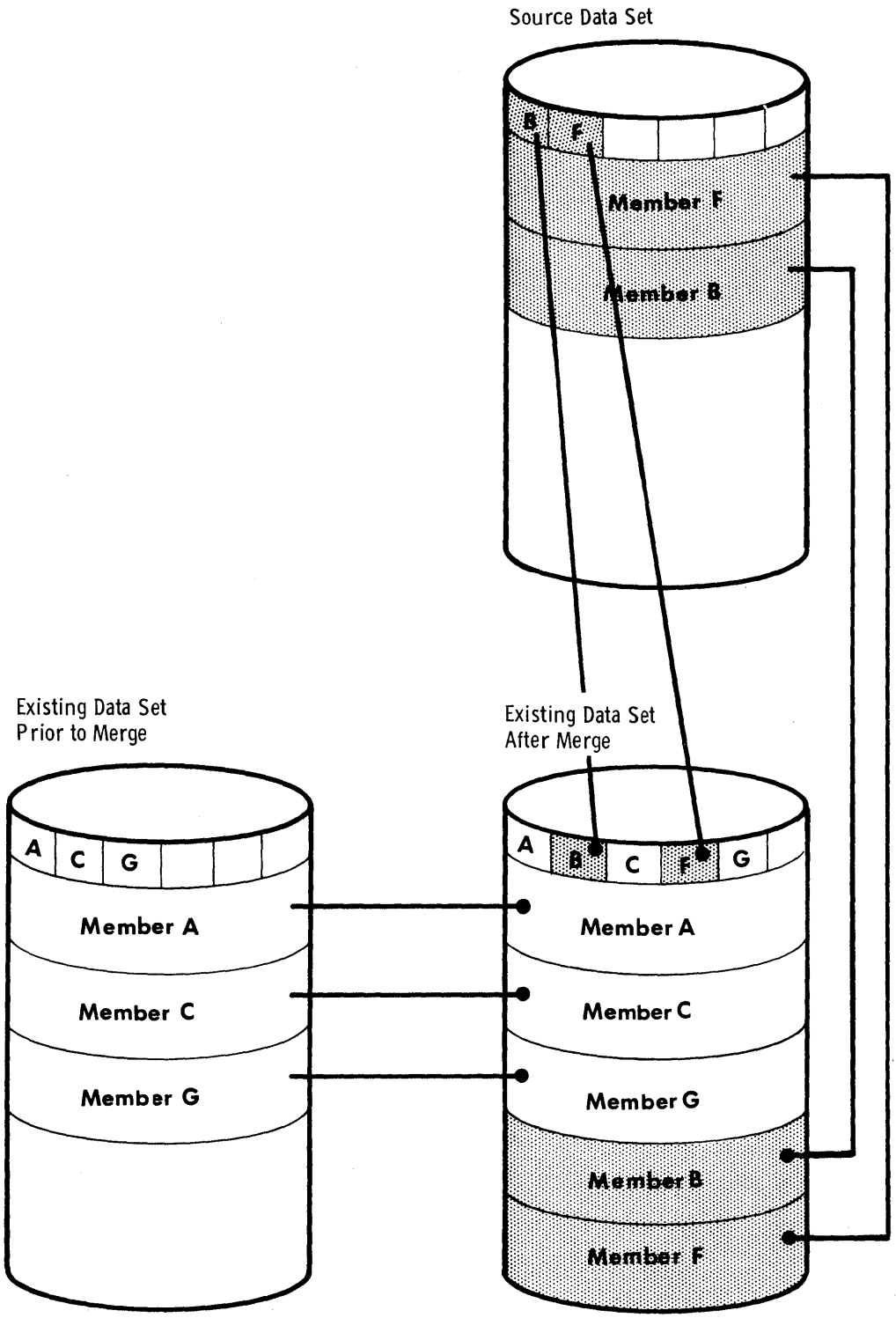


IEHMOVE Figure 5. A Copied Partitioned Data Set

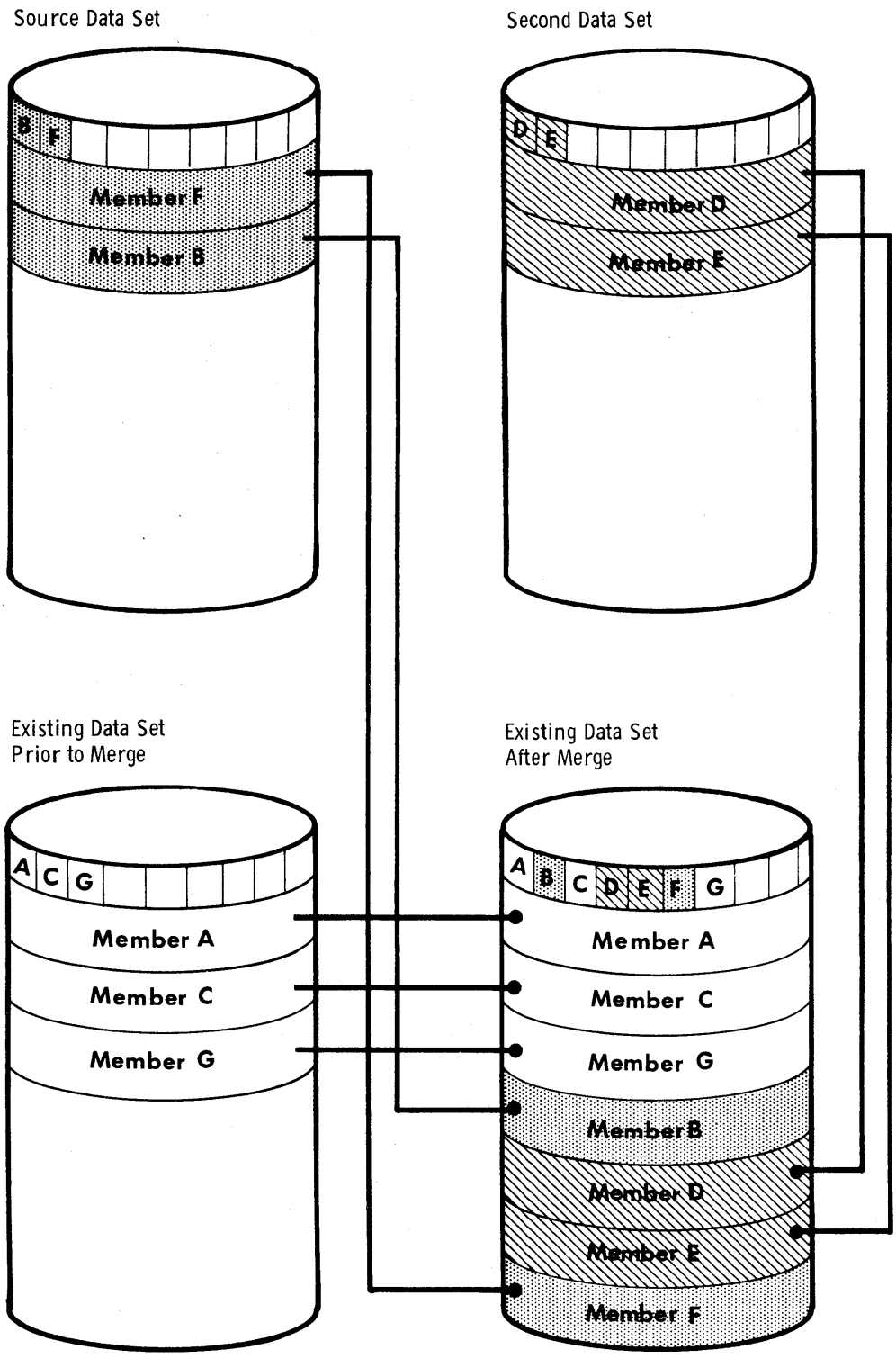
Order of Merged Members: Members that are merged into an existing data set are placed, in collating sequence, after the last member in the existing data set.

IEHMOVE Figure 6 shows how members from one data set are merged into an existing data set.

IEHMOVE Figure 7 shows how members from two data sets are merged into an existing data set. Members from additional data sets can be merged in like manner.



IEHMOVE Figure 6. Merging Two Data Sets



IEHMOVE Figure 7. Merging Three Data Sets

Moving or Copying a Group of Cataloged Data Sets

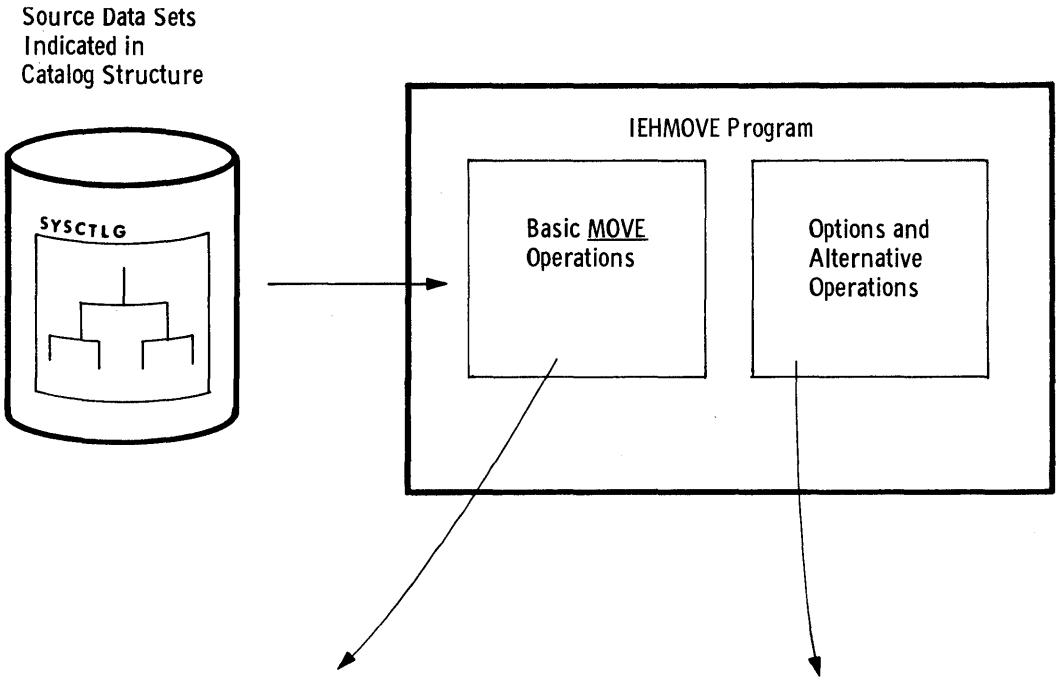
The IEHMOVE program can move or copy a group of data sets that are cataloged on the same volume and whose names are qualified by one or more identical names. For example, a group of data sets qualified by the name A.B could include data sets named A.B.D and A.B.E, but could not include data sets named A.C.D. or A.D.F.

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.

Notes: If a group of data sets is moved or copied onto magnetic tape, the data sets must be retrieved one by one by data set name and file sequence number, or (for unlabeled or non-standard labeled tapes) by file sequence number.

The IEHLIST system utility program can be used to determine the structure of the catalog.

IEHMOVE Figures 8 and 9 show basic move and copy (respectively) operations for a group of cataloged data sets. Options and alternative operations that can be specified by the user are also shown.



1. Move the data set group (not including password protected data sets) to the specified volume(s).
2. Scratch the source data sets (direct-access only).
3. Update the catalog to refer to the moved data sets.

. Prevent the updating of the catalog.

. Include password protected data sets in the operation.

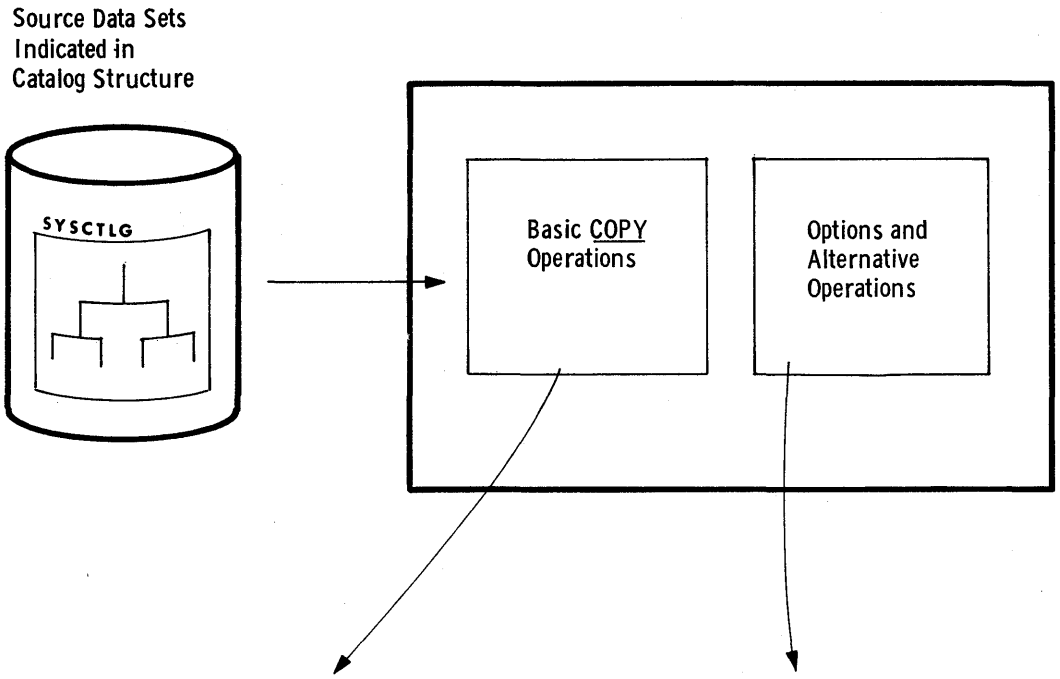
. Include an additional data set or sets in the operation.

. Exclude a data set or sets from the operation.

. Unload a data set or sets.

Note: Merging is not accomplished with this function.

IEHMOVE Figure 8. Moving a Group of Cataloged Data Sets



1. Copy the data set group (not including password protected data sets).

   Note: The source data sets are not scratched. The catalog is not updated to refer to the copied data sets.

- Include password protected data sets in the operation.
- Uncatalog the source data sets.
- Catalog the copied data sets on the receiving volume(s).
- Include an additional data set or sets in the operation.
- Exclude a data set or sets from the operation.
- Unload a data set or sets.

Note: Merging is not accomplished with this function.

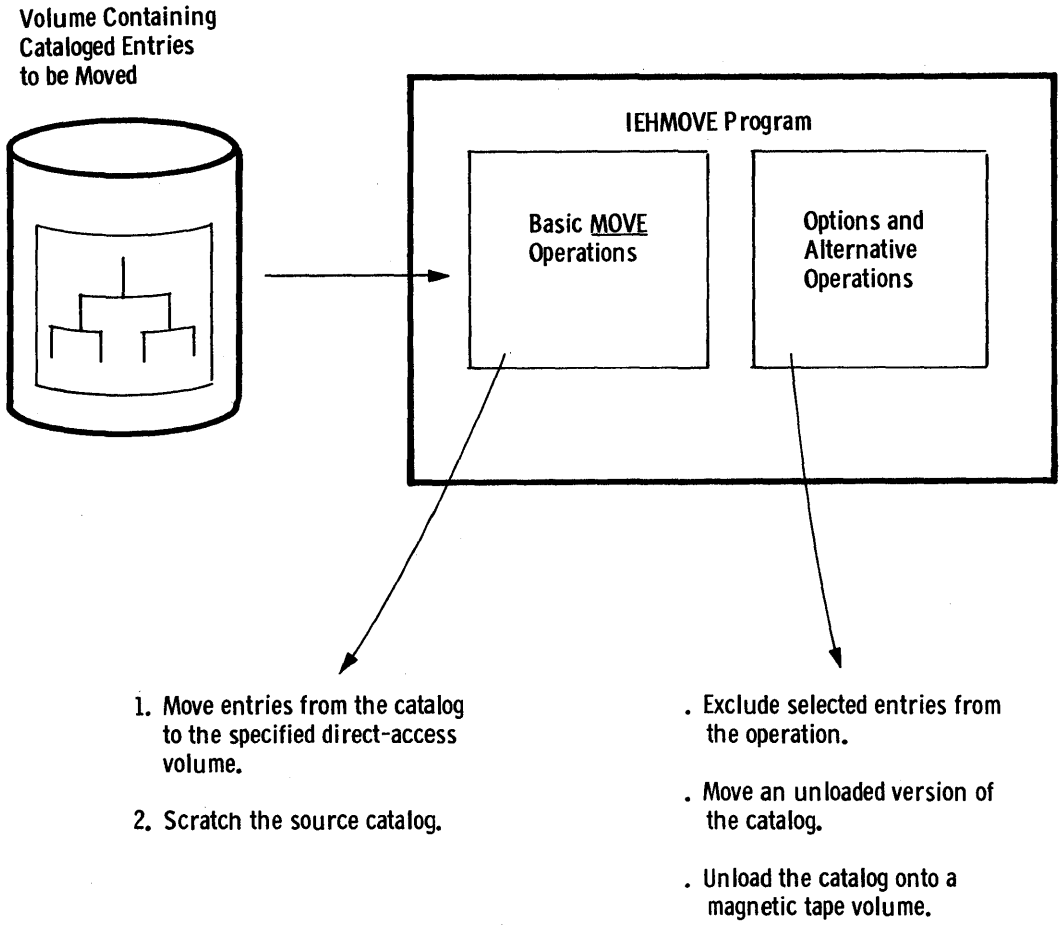
IEHMOVE Figure 9. Copying a Group of Cataloged Data Sets

Moving or Copying a Catalog

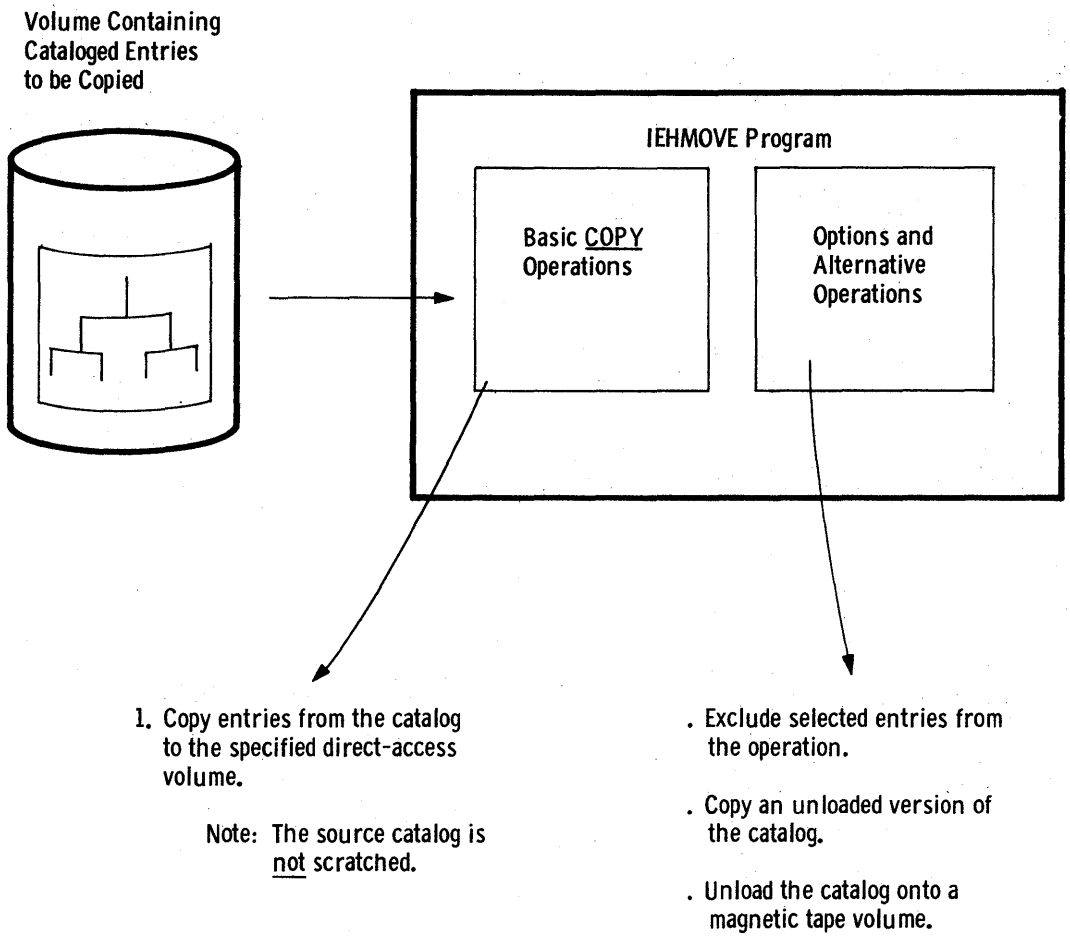
The IEHMOVE program can move or copy a catalog or portions of a catalog without copying the data sets represented by the cataloged entries. The SYSCTLG (catalog) data set need not be defined on the receiving volume prior to the operation. Moved or copied entries are merged with existing entries, if any, on the receiving volume.

Note: The receiving volume must be a direct access volume unless the catalog is to be unloaded.

IEHMOVE Figures 10 and 11 show basic move and copy (respectively) operations for catalog entries. Options and alternative operations that can be specified by the user are shown also.



IEHMOVE Figure 10. Moving the Catalog



IEHMOVE Figure 11. Copying the Catalog

Moving or Copying a Volume of Data Sets

The IEHMOVE program can move or copy the data sets of an entire direct access volume onto 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, cataloged entries (if any) associated with the source data sets remain unchanged. The IEHPROGM system utility program can be used to uncatalog all of the cataloged data sets and recatalog them according to their new location.

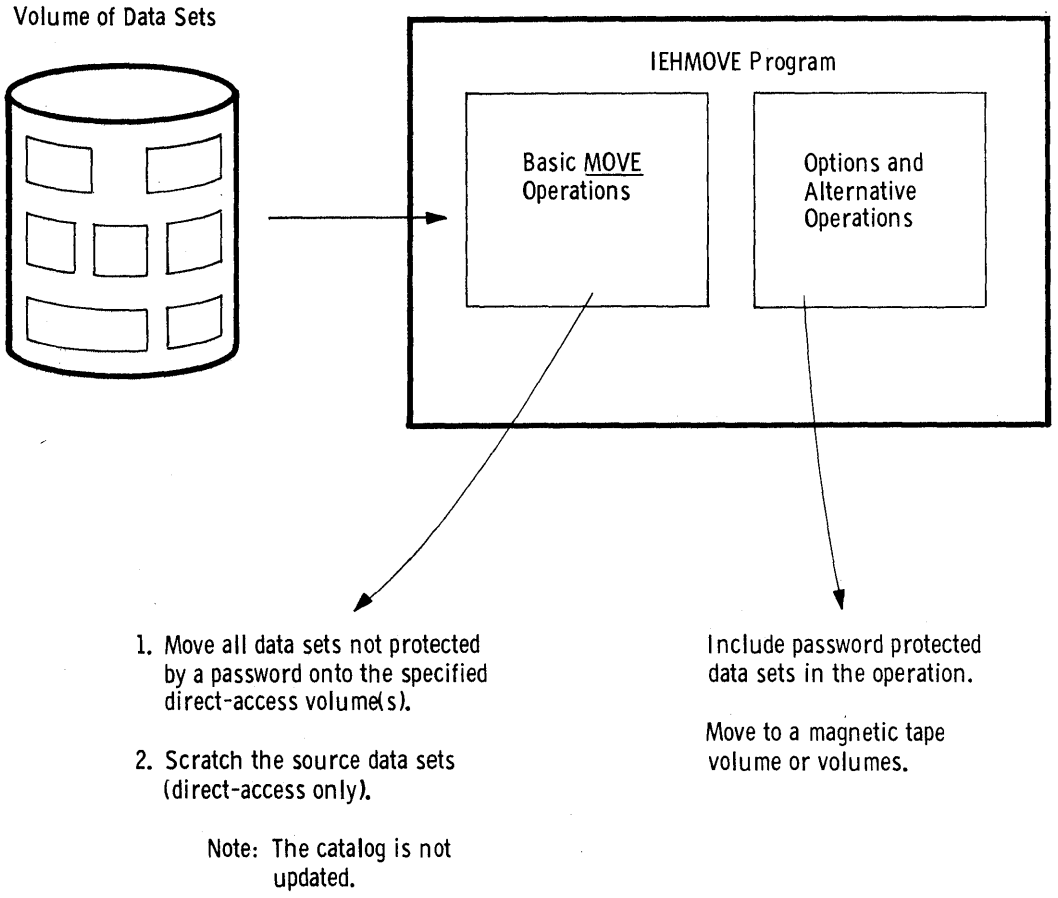
Notes: If the source volume contains a catalog (SYSCTLG) data set, that data set is the last to be moved or copied onto the receiving volume.

If a volume of data sets is moved or copied onto magnetic tape, the data sets must be retrieved one by one by data set name and file sequence number, or (for unlabeled or nonstandard labeled tapes) by file sequence number.

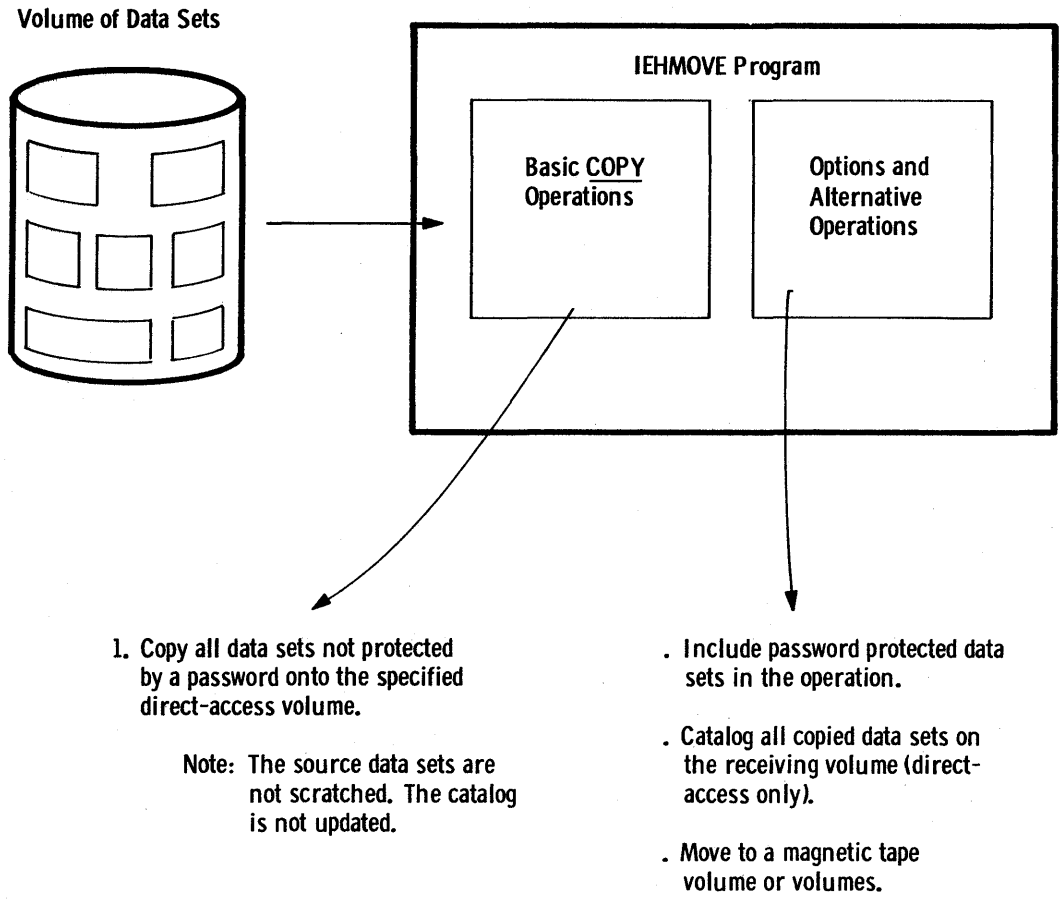
When copying a volume of data sets, the user has the option of cataloging all source data sets in a SYSCTLG data set on a receiving volume. However, if a SYSCTLG data set exists on the source volume, error messages indicating that an inconsistent index structure exists will be generated when the source SYSCTLG entries are merged into the SYSCTLG data set on the receiving volume.

The MOVE VOLUME feature does not merge partitioned data sets. If a data set on the volume to be moved or copied has a name identical to a data set name on the receiving volume, the data set is not moved, copied, or merged onto the receiving volume.

IEHMOVE Figures 12 and 13 show basic move and copy (respectively) operations for a volume of data sets. Options and alternative operations that can be specified by the user are shown also.



IEHMOVE Figure 12. Moving a Volume of Data Sets



IEHMOVE Figure 13. Copying a Volume of Data Sets

Moving or Copying Direct Data Sets With Variable Spanned Records

The IEHMOVE program can move or copy direct data sets with variable spanned records (VRE) from one direct access volume to a compatible direct access volume, provided that the records do not exceed 32K.

Since a direct data set can reside on 1-5 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 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 respanned if necessary. Since the remaining track space is available for new records, variable spanned records are unloaded before being moved or copied back to a smaller device.

Notes: 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 multi-volume data set, it is recommended that the secondary allocation for direct data sets be at least two tracks. (See the "WRITE SZ" macro in IBM Supervisor and Data Management Macro Services, GC28-6647).

Inputs and Outputs

IEHMOVE Table 2 lists the major inputs to and outputs from the IEHMOVE program.

IEHMOVE Table 2. Data Sets Used (Input) and Produced (Output) by the IEHMOVE Program

Inputs	Source Data Set(s): This data set(s) contains the data to be moved, copied, or merged into an output data set. Control Data Set: This data set contains utility control statements, which are used to control the functions of the program. Work Data Set: This data set is a work area used by the IEHMOVE program.
Outputs	Output Data Set: This data set is the result of the move, copy, or merge operation. Message Data Set: This data set contains informational messages (e.g., the names of moved or copied data sets; the contents of applicable utility control statements) and error messages, if applicable.

ADDITIONAL OUTPUTS

The IEHMOVE program produces a return code to indicate the results of program execution. The return codes and their interpretations are as follows:

00 -- successful completion.
04 -- a specified function was not completely successful. Processing continues.
08 -- a condition has occurred from which recovery is possible. Processing continues.
12 -- an unrecoverable error has occurred. The job step is terminated.

Control

The IEHMOVE program is controlled by job control statements and utility control statements. The job control statements are used to:

- Execute or invoke the program.
- Define the work and control data sets.
- Define volumes and/or devices to be used during the course of program execution.
- Prevent data sets from being deleted inadvertently.
- Provide label, density, and conversion information for magnetic tapes.

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

IEHMOVE Table 3 shows the job control statements necessary for executing or invoking the IEHMOVE program.

IEHMOVE Table 3. Job Control Statements for the IEHMOVE Program (Part 1 of 3)

Statement	Usage
JOB statement	This statement initiates the job.
EXEC statement	This statement 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 the following section.)
SYSPRINT DD statement	This statement 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 statement //SYSUT1	This statement defines a volume on which a work data set required by the IEHMOVE program is placed. The statement is coded as follows: DD UNIT=xxxx,VOLUME=SER=xxxxxx,DISP=OLD
//anyname1* DD UNIT=xxxx,VOLUME=SER=xxxxxx,DISP=OLD This statement defines a permanently mounted volume. One 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.) In this 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. Note: This statement is required only if a permanently mounted volume is referred to in the job step. For example, if a subsequent UNCATLG parameter specifies that an index search begin on the system residence volume, this statement is required.	
//anyname2** DD UNIT=xxxx,VOLUME=SER=xxxxxx,DISP=OLD This statement defines a mountable device type. One statement must be included for each mountable device to be used in the job step. Note: 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 statments) that multiple volumes are to be processed. This indication can be in the form of deferred mounting. Deferred mounting can be used by specifying: DISP=(...,KEEP),VOLUME=(PRIVATE,...),UNIT=(xxxx,,DEFER) In a DD statement defining a mountable device. Refer to Appendix D for information on defining mountable devices.	

(Part 1 of 3)

IEHMOVE Table 3. Job Control Statements for the IEHMOVE Program (Part 2 of 3)

Statement	Usage
//tape DD //	DSNAME=xxxxxxxx,UNIT=xxxx,VOLUME=SER=xxxxxx, DISP=(...,KEEP),LABEL=(...,...),DCB=(TRTCH=C,DEN=x) 7-track tape only A FROMDD or a TODD keyword in a utility control statement refers to this statement for label and mode information. A version of this statement is necessary when moving or copying from or to a 7-track magnetic tape volume, a 9-track magnetic tape volume not having standard labels, or a 1600 bpi 9-track magnetic tape volume on a single-density drive, or when copying to an 800 bpi magnetic tape on a dual-density drive. Note: The date on which a data set is moved or copied onto a magnetic tape volume is automatically recorded in the HDR 1 record of a standard tape label, provided that a TODD keyword 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 the TODD keyword.
SYSIN DD statement	This statement 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 alternatively be defined as being an unblocked sequential data set, or a member of a procedure library.

*This DD statement is arbitrarily assigned the ddname DD1 in the IEHMOVE examples.

**This 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.

Notes: At least 80 contiguous tracks must be available for work space on the volume defined by the SYSUT1 DD statement. (This figure is based on a 2311 being the work volume. If a direct access device other than a 2311 is used, an equivalent amount of space must be available.)

To define a sequence number for a data set on a tape volume, or to specify a specific device (for example, unit address 190, rather than a group name such as SYSDA or a device type such as 2400 or 2314), you must use 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, you must specify the sequence number of the data set in the list field of the FROM/TO=device=list parameter on the utility control statement. To move or copy a data set from or to a specific device you must specify the unit address (rather than a group name or device type), in the device field of the FROM/TO=device=list parameter on the utility control statement (see IEHMOVE Example 13 for an example of this function). When you want to copy onto a unit record or nonlabeled tape volume, you must fill the list field of the FROM/TO=device=list parameter with any standard name or number (see IEHMOVE Example 13).

(Part 2 of 3)

IEHMOVE Table 3. Job Control Statements for the IEHMOVE Program (Part 3 of 3)

The blocksize for the SYSPRINT (message) data set must be a multiple of 121. The blocksize for the SYSIN (control) data set must be a multiple of 80. Any blocking factor can be specified for these blocksizes.
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. Since the IEHMOVE program 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 the IEHPROGM system utility program prior to the execution of the IEHMOVE program. Examples of this are included in the IEHMOVE examples.
Blocked-format data sets that do not contain user data TTRNs or keys can be reblocked or unblocked by including the proper keyword subparamters in the DCB operand of the DD statement used to preallocate 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 the publication IBM System/360 Operating System: Supervisor and Data Management Services, GC28-6646.)
When the IEHMOVE program is dynamically invoked in a job step containing another program, the DD statements defining mountable devices for the IEHMOVE program must be included in the job stream prior to DD statements defining data sets required by the other program.

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 direct access devices that support the track overflow feature. (For BDAM 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 track overflow or vice versa, provided that:

- 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 RECFM 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 onto a device other than a direct access device.

PARM Information in the EXEC Statement

The EXEC statement for the IEHMOVE program 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=n'
                or
//  EXEC  PGM=IEHMOVE,PARM='POWER=n,LINECNT=xx'
                or
//  EXEC  PGM=IEHMOVE,PARM='LINECNT=xx'
                or
//  EXEC  PGM=IEHMOVE
```

where

POWER=n
requests that the normal amount of space allocated for work areas be increased n times. This parameter is used when 750 or more members are being moved or copied. The progression for the value of n is:

from 750 to 1500 members -- POWER=2.
from 1501 to 2250 members -- POWER=3.
from 2251 to 3000 members -- POWER=4.
etc.

Notes:

1. If POWER=2, the work space requirement on the SYSUT1 volume is doubled; if POWER=3, work space requirement is three times the basic requirement, etc. For example, if POWER=2, 160 tracks on a 2311 (or equivalent space on a device other than a 2311) must be available.

2. When moving or copying a catalog, you should count each level of index as a member in order to calculate the space required for a work area. For example, if a catalog has 200 entries and each entry has four levels of indexes, you must specify POWER=2 in order to provide a sufficient work area for the utility.

LINECNT=xx
specifies the number of lines per page in the listing of the SYSPRINT data set, where xx is a 2-digit number.

UTILITY CONTROL STATEMENTS

Combinations of the following utility control statements are used to control the functions of the program.

- The MOVE DSNAME (move a data set) statement.
- The COPY DSNAME (copy a data set) statement.
- The MOVE DSGROUP (move a group of cataloged data sets) statement.
- The COPY DSGROUP (copy a group of cataloged data sets) statement.
- The MOVE PDS (move a partitioned data set) statement.
- The COPY PDS (copy a partitioned data set) statement.
- The MOVE CATALOG (move cataloged entries) statement.
- The COPY CATALOG (copy cataloged entries) statement.
- The MOVE VOLUME (move a volume of data sets) statement.
- The COPY VOLUME (copy a volume of data sets) statement.

In addition, there are four "subordinate" control statements that can be used to modify the effect of a MOVE or COPY DSGROUP, MOVE or COPY PDS, or MOVE or COPY CATALOG operation. The subordinate statements and the control statements with which they can be combined are shown in IEHMOVE Table 4.

IEHMOVE Table 4. Valid Combinations of Control Statements

Utility Statements	Subordinate Statements
MOVE DSGROUP or COPY DSGROUP	INCLUDE EXCLUDE
MOVE PDS or COPY PDS	INCLUDE EXCLUDE REPLACE SELECT
MOVE CATALOG or COPY CATALOG	EXCLUDE

The MOVE DSNAME Statement

The MOVE DSNAME statement is used to move a data set. If the data set is cataloged, the catalog is automatically updated unless UNCATLG is specified. The source data set is scratched.

Name	Operation	Operand
[name]	MOVE	DSNAME=name TO=device=list [FROM=device=list CVOL=device=serial] [UNCATLG] [RENAME=name] [FROMDD=ddname] [TODD=ddname]

DSNAME=name
: specifies the fully qualified name of the data set to be moved.

TO=device=list
: specifies the volume or volumes to which the data set is to be moved.

FROM=device=list
: specifies the volume or volumes on which the data set currently resides, if it is not cataloged.

  If the data set is cataloged, FROM should not be written.

CVOL=device=serial
: specifies the device type and volume serial number of the volume on which the catalog search for the data set is to begin.

  If neither CVOL nor FROM is written, the data set is assumed to be cataloged on the system residence volume.

UNCATLG
specifies that the catalog entry pertaining to the data set is to be removed. This parameter should be used only if the source data set is cataloged.

UNCATLG is ignored if the volume is identified by FROM.

RENAME=name
specifies that the data set is to be renamed, and indicates the new name.

FROMDD=ddname (for data sets residing on magnetic tape volumes)
specifies the name of a DD statement from which DCB and LABEL information are obtained. DCB attributes for unloaded data sets are always (RECFM=FB,LRECL=80,BLKSIZE=800).

FROMDD can be omitted for 800 bpi 9-track tape with standard labels on single-density drives or for 800 or 1600 bpi 9-track tapes with standard labels on dual-density drives.

TODD=ddname (for data sets to be moved onto magnetic tape volumes)
specifies the name of a DD statement from which DCB and LABEL information are obtained. This information describes the mode and label of the magnetic tape where the moved data set is to reside. RECFM, LRECL, and BLKSIZE information is ignored.

TODD can be omitted for 800 bpi 9-track tape with standard labels on single-density drives or for 1600 bpi 9-track tape with standard labels on dual-density drives.

The COPY DSNAME Statement

The COPY DSNAME statement is used to copy a data set. The source data set, if cataloged, remains cataloged unless UNCATLG is specified.

Name	Operation	Operand
[name]	COPY	DSNAME=name TO=device=list [FROM=device=list [CVOL=device=serial] [UNCATLG] [CATLG] [RENAME=name] [FROMDD=ddname] [TODD=ddname]

DSNAME=name
specifies the fully qualified name of the data set to be copied.

TO=device=list
specifies the volume or volumes on which the data set is to be copied.

FROM=device=list
specifies the volume or volumes on which the data set currently resides, if it is not cataloged.

If the data set is cataloged, FROM should not be written.

CVOL=device=serial
specifies the device type and volume serial number of the volume on which the catalog search for the data set is to begin.

If neither CVOL nor FROM is written, the data set is assumed to be cataloged on the system residence volume.

UNCATLG
specifies that the catalog entry pertaining to the source data set is to be removed. This parameter should be used only if the source data set is cataloged.

UNCATLG is ignored if the volume is identified by FROM.

CATLG
specifies that the copied data set is to be cataloged on its receiving volume, if it is a direct access volume. If a catalog does not exist on the receiving volume, a new catalog is created.

RENAME=name
specifies that the data set is to be renamed, and indicates the new name.

FROMDD=ddname (For data sets residing on magnetic tape volumes)
specifies the name of the DD statement from which DCB and LABEL information are obtained. DCB attributes for unloaded data sets are always (RECFM=FB,LRECL=80,BLKSIZE=800).

FROMDD can be omitted for 800 bpi 9-track tape with standard labels on single-density drives or for 800 or 1600 bpi 9-track tapes with standard labels on dual-density drives.

TODD=ddname (For data sets to be copied onto magnetic tape volumes)
specifies the name of a DD statement from which DCB and LABEL information are obtained. This information describes the mode and label of the magnetic tape where the copied data set is to reside. RECFM, LRECL, and BLKSIZE information is ignored.

TODD can be omitted for 800 bpi 9-track magnetic tape with standard labels on single-density drives or for 1600 bpi 9-track tape with standard labels on dual-density drives.

The MOVE DSGROUP Statement

The MOVE DSGROUP statement is used to move groups of data sets that are cataloged on the same volume and whose names are partially qualified by one or more identical names. Source data sets are scratched.

Note: Data sets to be moved can reside on direct access volumes only.

MOVE DSGROUP operations cause the specified catalog to be updated automatically unless UNCATLG is specified.

INCLUDE and EXCLUDE statements, discussed later in this section, can be used to add to or delete data sets from the group.

Name	Operation	Operand
[name]	MOVE	DSGROUP[=name] TO=device=list [CVOL=device=serial] [PASSWORD] [UNCATLG] [TODD=ddname]

DSGROUP=name
specifies a qualified name (an index structure). All cataloged data sets whose names are qualified by this name are moved. If the name is a fully qualified data set name, only that data set is moved.

DSGROUP
specifies that all data sets cataloged on the specified volume are to be moved.

TO=device=list
specifies the volume or volumes to which the specified group of data sets is to be moved.

CVOL=device=serial
speciifies the device type and volume serial number of the volume on which the catalog search for the data sets is to begin.

If CVOL is omitted, the specified group of data sets is assumed to be cataloged on the system residence volume.

PASSWORD
specifies that password protected data sets contained in the group are to be moved.

If PASSWORD is omitted, only data sets that are not protected are moved.

UNCATLG
specifies that the catalog entries pertaining to the specified group of data sets are to be removed.

TODD=ddname (For groups to be moved onto magnetic tape volumes)
specifies the name of a DD statement from which DCB and LABEL information are obtained. This information describes the mode and label of the magnetic tape where the moved data sets are to reside. RECFM, LRECL, and BLKSIZE information is ignored.

TODD can be omitted for 800 bpi 9-track tape with standard labels on single-density drives or for 1600 bpi 9-track tape with standard labels on dual-density drives.

The COPY DSGROUP Statement

The COPY DSGROUP statement is used to copy groups of data sets that are cataloged on the same control volume and whose names are partially qualified by one or more identical names. The source data sets remain cataloged unless UNCATLG is specified.

Note: Data sets to be copied can reside on direct access volumes only.

INCLUDE and EXCLUDE statements, discussed later in this section, can be used to add data sets to or delete data sets from the group.

Name	Operation	Operand
[name]	COPY	DSGROUP[=name] TO=device=list [CVOL=device=serial] [PASSWORD] [UNCATLG] [CATLG] [TODD=ddname]

DSGROUP=name
specifies a qualified name (an index structure). All cataloged data sets whose names are qualified by this name are copied. If the name is a fully qualified data set name, only that data set is copied.

DSGROUP
specifies that all data sets cataloged on the specified volume are to be copied.

TO=device=list
specifies the volume or volumes on which the specified group of data sets is to be copied.

CVOL=device=serial
specifies the device type and volume serial number of the volume on which the search for the catalog is to begin.

If CVOL is omitted, the specified group of data sets is assumed to be cataloged on the system residence volume.

PASSWORD
specifies that password protected data sets contained in the group are to be copied.

If PASSWORD is omitted, only data sets that are not protected are copied.

UNCATLG
specifies that the catalog entries pertaining to the source group of data sets are to be removed.

CATLG
specifies that the copied data sets are to be cataloged on their receiving volumes, if they are direct access volumes. If catalogs do not exist on the receiving volumes, they are created.

TODD=ddname (For groups to be copied onto magnetic tape volumes)
specifies the name of a DD statement from which DCB and LABEL information are obtained. This information describes the mode and label of the magnetic tape on which the copied data sets are to reside. RECFM, LRECL, and BLKSIZE information is ignored.

TODD can be omitted for 800 bpi 9-track tapes with standard labels on single-density drives or for 1600 bpi 9-track tape with standard labels on dual-density drives.

The 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 delete members.

If the IEHMOVE program is used to allocate space for an output partitioned data set, the MOVE PDS statement can be used to expand a partitioned directory.

MOVE PDS causes the specified catalog to be updated automatically unless UNCATLG is specified.

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.

Name	Operation	Operand
name	MOVE	PDS=name
		TO=device=serial
		[FROM=device=serial]
		[CVOL=device=serial]
		[EXPAND=nn]
		[UNCATLG]
		[RENAME=name]
		[FROMDD=ddname]
		[TODD=ddname]

PDS=name
specifies the fully qualified name of the partitioned data set to be moved.

TO=device=serial
specifies the device type and volume serial number of the volume to which the partitioned data set is to be moved. The "list" subparameter (TO=device=list) may be used when unloading a partitioned data set that must span tape volumes.

FROM=device=serial
specifies the device type and volume serial number of the volume on which the partitioned data set currently resides, if it is not cataloged.

If the data set is cataloged, FROM should not be written. FROM=device=list may be used when loading a PDS.

CVOL=device=serial
specifies the device type and volume serial number of the volume on which the catalog search for the partitioned data set is to begin.

If neither FROM nor CVOL is written, the partitioned data set is assumed to be cataloged on the system residence volume.

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.

EXPAND cannot be specified if space is previously allocated.

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.

RENAME=name
specifies that the data set is to be renamed, and indicates the new name.

FROMDD=ddname (For unloaded partitioned data sets on magnetic tape volumes)
specifies the name of a DD statement from which DCB and LABEL information are obtained. This information describes the mode and label of the magnetic tape and, in addition, must include the DCB attributes of the unloaded data set (RECFM=FB,LRECL=80, BLKSIZE=800).

FROMDD can be omitted for 800 bpi 9-track tapes with standard labels on single-density drives or for 800 or 1600 bpi 9-track tapes with standard labels on dual-density drives.

TODD=ddname (For partitioned data sets to be unloaded onto magnetic tape volumes)
specifies the name of a DD statement from which DCB and LABEL information are obtained. This information describes the mode and label of the magnetic tape on which the unloaded data set is to reside and, in addition, must include the DCB attributes of the unloaded data set (RECFM=FB,LRECL=80,BLKSIZE=800).

TODD can be omitted for 800 bpi 9-track magnetic tapes with standard labels on single-density drives or for 1600 bpi 9-track tape with standard labels on dual-density drives.

The 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 delete members.

If the IEHMOVE program is used to allocate space for an output partitioned data set, the COPY PDS statement can be used to expand a partitioned directory.

The source partitioned data set remains cataloged unless UNCATLG is specified.

If the receiving volume already contains a partitioned data set with the same name, the two are merged.

Name	Operation	Operand
[name]	COPY	PDS=name
		TO=device=serial
		[FROM=device=serial]
		[CVOL=device=serial]
		[EXPAND=nn]
		[UNCATLG]
		[CATLG]
		[RENAME=name]
		[FROMDD=ddname]
		[TODD=ddname]

PDS=name
specifies the fully qualified name of the partitioned data set to be copied.

TO=device=serial
specifies the device type and volume serial number of the volume to which the partitioned data set is to be moved. The "list" subparameter (TO=device=list) may be used when unloading a partitioned data set that must span tape volumes.

FROM=device=serial
specifies the device type and volume serial number of the volume on which the partitioned data set currently resides, if it is not cataloged.
If the data set is cataloged, FROM should not be written.
FROM=device=list may be used when loading a PDS.

CVOL=device=serial
specifies the device type and volume serial number of the volume on which the catalog search for the partitioned data set is to begin. If neither FROM nor CVOL is written, the partitioned data set is assumed to be cataloged on the system residence volume.

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.

EXPAND cannot be specified if space is previously allocated.

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 partitioned data set is cataloged.

UNCATLG is ignored if the volume is identified by FROM.

CATLG
specifies that the copied partitioned data set is to be cataloged on the receiving volume, if it is a direct access volume. If a catalog does not exist on the receiving volume, a new catalog is created.

RENAME=name
specifies that the data set is to be renamed, and indicates the new name.

FROMDD=ddname (For unloaded partitioned data sets residing on magnetic tape volumes)
specifies the name of a DD statement from which DCB and LABEL information are obtained. This information describes the mode and label of the magnetic tape and, in addition, must include the DCB attributes of the unloaded data set (RECFM=FB,LRECL=80, BLKSIZE=800).

FROMDD can be omitted for 800 bpi 9-track tapes with standard labels on single-density drives or for 800 or 1600 bpi 9-track tapes with standard labels on dual-density drives.

TODD=ddname (For partitioned data sets to be unloaded onto magnetic tape volumes)
specifies the name of a DD statement from which DCB and LABEL information are obtained. This information describes the mode and label of the magnetic tape on which the unloaded data set is to reside and, in addtion, must include the DCB attributes of the unloaded data set (RECFM=FB,LRECL=80,BLKSIZE=800).

TODD can be omitted for 800 bpi 9-track tapes with standard labels on single-density drives or for 1600 bpi 9-track tape with standard labels on dual-density drives.

The MOVE CATALOG Statement

The MOVE CATALOG statement is used to move the entries of a catalog 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 catalog, the source catalog entries are merged with it.

```
┌──────┬─────────┬──────────────────────────────────────────────────┐
│Name  │Operation│Operand                                           │
├──────┼─────────┼──────────────────────────────────────────────────┤
│[name]│MOVE     │CATALOG[=name]                                    │
│      │         │TO=device=serial                                  │
│      │         │[CVOL=device=serial]                              │
│      │         │[FROM=device=serial]                              │
│      │         │[FROMDD=ddname]                                   │
│      │         │[TODD=ddname]                                     │
└──────┴─────────┴──────────────────────────────────────────────────┘
```

CATALOG=name
: specifies a qualified name. All catalog entries whose names are qualified by this name are moved. If the name is a fully qualified data set name, only the catalog entry that corresponds to that data set is moved.

CATALOG
: specifies that all entries in the catalog are to be moved.

TO=device=serial
: specifies the device type and volume serial number of the volume to which the specified catalog entries are to be moved.

CVOL=device=serial
: specifies the device type and volume serial number of the volume on which the search for the catalog is to begin.

FROM=device=serial
: specifies the device type and volume serial number of the volume on which an unloaded version of the catalog resides.

  If neither FROM nor CVOL is written, the catalog is assumed to reside on the system residence volume.

FROMDD=ddname (for unloaded catalogs residing on magnetic tape volumes)
: specifies the name of a DD statement from which DCB and LABEL information are obtained. This information describes the mode the label of the magnetic tape and, in addition, must include the DCB attributes of the unloaded catalog (RECFM=FB,LRECL=80, BLKSIZE=800).

  FROMDD can be omitted for 800 bpi 9-track magnetic tapes with standard labels on single-density drives or for 800 or 1600 bpi 9-track tapes with standard labels on dual-density drives.

TODD=ddname (For catalogs to be unloaded onto magnetic tape volumes)
: specifies the name of a DD statement from which DCB and LABEL information are obtained. This information describes the mode and label of the magnetic tape on which the unloaded catalog is to reside and, in addition, must include the DCB attributes of the unloaded catalog (RECFM=FB,LRECL=80,BLKSIZE=800).

  TODD can be omitted for 800 bpi 9-track tapes with standard labels on single-density drives or for 1600 bpi 9-track tape with standard labels on dual-density drives.

The COPY CATALOG Statement

The COPY CATALOG statement is used to copy the entries in a catalog without copying the data sets associated with these entries. Certain entries can be excluded from a copy operation with the EXCLUDE statement. If the receiving volume contains a catalog, the source catalog is merged with it.

Name	Operation	Operand
[name]	COPY	CATALOG[=name] TO=device=seial [CVOL=device=serial] [FROM=device=serial] [FROMDD=ddname] [TODD=ddname]

CATALOG=name
specifies a qualified name. All catalog entries whose names are qualified by this name are copied. If the name is a fully qualified data set name, only the catalog entry that corresponds to that data set is copied.

CATALOG
specifies that all entries in the catalog are to be copied.

TO=device=serial
specifies the device type and volume serial number of the volume onto which the specified catalog entries are to be copied.

CVOL=device=serial
specifies the device type and volume serial number of the volume on which the search for the catalog is to begin.

FROM=device=serial
specifies the device type and volume serial number of the volume on which an unloaded version of the catalog resides.

If neither FROM nor CVOL is written, the catalog is assumed to reside on the system residence volume.

FROMDD=ddname (For unloaded catalogs residing on magnetic tape volumes)
specifies the name of a DD statement from which DCB and LABEL information is obtained. This information describes the mode and label of the magnetic tape and, in addition, must include the DCB attributes of the unloaded catalog (RECFM=FB,LRECL=80,BLKSIZE=800).

FROMDD can be omitted for 800 bpi 9-track magnetic tapes with standard labels. on single-density drives or for 800 or 1600 bpi 9-track tapes with standard labels on dual-density drives.

TODD=ddname (For catalogs to be unloaded onto magnetic tape volumes)
specifies the name of a DD statement from which DCB and LABEL information are obtained. This information describes the mode and label of the magnetic tape on which the unloaded catalog is to reside and, in addition, must include the DCB attributes of the unloaded catalog (RECFM=FB,LRECL=80,BLKSIZE=800).

TODD can be omitted for 800 bpi 9-track magnetic tapes with standard labels on single-density drives or for 1600 bpi 9-track tape with standard labels on dual-density drives.

The 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.

Note: Data sets to be moved can reside on direct access volumes only.

Name	Operation	Operand
[name]	MOVE	VOLUME=device=serial TO=device=list [PASSWORD] [TODD=ddname]

VOLUME=device=serial
specifies the device type and volume serial number of the source volume.

TO=device=list
specifies the volume or volumes to which the data sets are to be moved.

PASSWORD
specifies that password protected data sets are to be included in the operation.
If PASSWORD is omitted, only data sets that are not protected are moved.

TODD=ddname (For magnetic tape volumes)
specifies the name of a DD statement from which DCB and LABEL information are obtained. This information describes the mode and label of the receiving tape volume.

TODD can be omitted for 800 bpi 9-track magnetic tapes with standard labels on single-density drives or for 1600 bpi 9-track tape with standard labels on dual-density drives.

The 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.

Note: Data sets to be copied can reside on direct access volumes only.

Name	Operation	Operand
[name]	COPY	VOLUME=device=serial TO=device=list [PASSWORD] [CATLG] [TODD=ddname]

VOLUME=device=serial
specifies the device type and volume serial number of the source volume.

TO=device=list
specifies the volume or volumes onto which the data sets are to be copied.

PASSWORD
specifies that password protected data sets are to be included in the operation.

If PASSWORD is omitted, only data sets that are not protected are copied.

CATLG
specifies that all copied data sets are to be cataloged in a SYSCTLG (catalog) data set on the receiving volume (direct access only). If a catalog does not exist on the receiving volume, it is created.

Note: If CATLG is specified and the source volume contains a SYSCTLG data set, error messages indicating that an inconsistent index structure exists will be issued when the source SYSCTLG data set entries are merged into the catalog on the receiving volume. (Since 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.)

TODD=ddname (For magnetic tape volumes)
specifies the name of a DD statement from which DCB and LABEL information are obtained. This information describes the mode and label of the receiving magnetic tape volume.

TODD can be omitted for 800 bpi 9-track tapes with standard labels on single-density drives or for 1600 bpi 9-track tape with standard labels on dual-density drives.

The INCLUDE Statement

The INCLUDE statement is used to enlarge the scope of MOVE or COPY DSGROUP or MOVE 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. Any number of INCLUDE statements can modify a MOVE or COPY statement. The INCLUDE statement is invalid when data is unloaded or when unloaded data is moved or copied.

Name	Operation	Operand
[name]	INCLUDE	DSNAME=name [MEMBER=membername] [FROM=device=list CVOL=device=serial]

DSNAME=name
specifies the fully qualified name of a data set.

With MOVE or COPY DSGROUP: the named data set is included in the group.

With MOVE or COPY PDS: either the entire named partitioned data set or a member of the data set is included in the operation.

MEMBER=membername (with MOVE or COPY PDS only)
specifies the name of a member of the partitioned data set named in the DSNAME parameter. This member is merged with the partitioned data set that is moved or copied. Its record characteristics must

be the same as those of the data set with which it it being merged. The data set containing this member is not scratched, regardless of the operation (MOVE or COPY).

This parameter must be included when modifying a MOVE or COPY PDS statement.

FROM=device=list
specifies the volume or volumes on which the data set resides, if the data set is not cataloged.

If the data set is cataloged, FROM should not be specified.

CVOL=device=serial
specifies the device type and volume serial number of the volume on which the catalog search for the data set is to begin.

If both FROM and CVOL are omitted, the specified data set is assumed to be cataloged on the system residence volume.

The EXCLUDE Statement

The exclude statement is used to restrict the scope of MOVE or COPY DSGROUP, MOVE or COPY PDS, OR MOVE 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 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 or COPY DSGROUP or MOVE or COPY CATALOG statement. The EXCLUDE statement is invalid when data is unloaded or when unloaded data is moved or copied.

Name	Operation	Operand
[name]	EXCLUDE	{DSGROUP=name {MEMBER=membername}

DSGROUP=name
specifies a qualified name.

With MOVE or COPY DSGROUP: all data sets whose names are qualified by this name are excluded from the operation.

With MOVE or COPY CATALOG: all catalog entries whose names are qualified by this name are excluded from the operation.

MEMBER=membername

With MOVE or COPY PDS: identifies a member to be excluded from the partitioned data set being moved or copied.

The SELECT Statement

The SELECT statement is used with the MOVE or COPY PDS statement to select members to be moved or copied, and to optionally rename these members. The SELECT and EXCLUDE statements and the SELECT and REPLACE statements are mutually exclusive; i.e., they cannot both be used with the same MOVE or COPY PDS statement. The SELECT statement is invalid when data is unloaded or when unloaded data is moved or copied.

Name	Operation	Operand
[name]	SELECT	MEMBER=(namelist)

MEMBER=(name[,name]...)
identifies the members to be moved or copied. These members belong to the partitioned data set identified in the preceding MOVE or COPY PDS statement.

MEMBER=((name,newname)[,(name,newname)]...)
identifies the members to be moved or copied and gives the new name for each member.

The REPLACE Statement

The REPLACE statement is used with a MOVE or COPY PDS statement to combine the exclude and include functions for a specified member of a partitioned data set. The replace function excludes a member from the operation and replaces it with a member from another partitioned data set. The "new" member must have the same name as the "old" member and must possess identical record characteristics. Any number of REPLACE statements can modify a MOVE or COPY PDS statement. The REPLACE statement is invalid when data is unloaded or when unloaded data is moved or copied.

Name	Operation	Operand
[name]	REPLACE	DSNAME=name MEMBER=membername [FROM=device=serial] [CVOL=device=serial]

DSNAME=name
specifies the fully qualified name of the partitioned data set that contains the new member.

MEMBER=membername
specifies the name of the member.

FROM=device=serial
specifies the device type and volume serial number of the volume that contains the partitioned data set named in the DSNAME parameter.

If the partitioned data set is cataloged, FROM should not be specified.

CVOL=device=serial
specifies the device type and volume serial number of the control volume on which the catalog search for the partitioned data set containing the new member is to begin.

If neither FROM nor CVOL are specified, the partitioned data set is assumed to be cataloged on the system residence volume.

IEHMOVE Examples

The following examples illustrate some of the uses of the IEHMOVE program.

IEHMOVE Example 1

Operation	Source Data	Devices Required	Comments
MOVE	Sequential data sets	1 2301 DRUM 2 2311 DISKS (mountable) 1 2311 DISK (system residence)	1. The source volume is demounted only after the job has completed. 2. Space is allocated by the IEHMOVE program.

In this example, three data sets (SEQSET1, SEQSET2, and SEQSET3) are to be moved from a disk volume (volume serial=231111) to three separate disk volumes (serials 231100, 231112, and 231113). Each of the three receiving volumes is mounted when it is required by the IEHMOVE program. The source data sets are not cataloged.

- The SYSUT1 DD Statement: defines the device that is to contain the work data set.
- The DD1 DD statement: defines the system residence device.
- The DD2 DD Statement: defines the mountable device on which the receiving volumes will be mounted as they are required.
- The DD3 DD Statement: defines a mountable device on which the source volume is to be mounted. Since the RETAIN subparameter is included, the volume remains mounted until the job has completed.
- The SYSIN DD Statement: defines the control data set which follows in the input stream.
- The MOVE Utility Control Statements: move the source data sets to volumes 231100, 231112, and 231113, respectively. The source data sets are scratched.

```
//MOVEDS   JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=2301,VOLUME=SER=230100,DISP=OLD
//DD1      DD   UNIT=2311,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=(2311,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(231100))
//DD3      DD   UNIT=2311,VOLUME=(PRIVATE,RETAIN,SER=(231111)),DISP=OLD
//SYSIN    DD   *
         MOVE   DSNAME=SEQSET1,TO=2311=231100,FROM=2311=231111
         MOVE   DSNAME=SEQSET2,TO=2311=231112,FROM=2311=231111
         MOVE   DSNAME=SEQSET3,TO=2311=231113,FROM=2311=231111
/*
```

IEHMOVE Example 1. Moving Three Sequential Data Sets Onto Three Separate Volumes

IEHMOVE Example 2

Operation	Source Data	Devices Required	Comments
COPY	Cataloged sequential data sets	1 2311 DISK 1 2301 DRUM (system residence) 2 2314 DISKS (mountable)	1. Space is allocated by the IEHMOVE program.

In this example, three cataloged data sets are to be copied onto a 2314 disk volume (volume serial=231401). The catalog is not updated. The source data sets are not scratched.

- The SYSUT1 DD Statement: defines the device that is to contain the work data set.
- The DD1 DD Statement: defines the system residence device.
- The DD2 DD Statement: defines a mountable device on which the source volume is mounted.
- The DD3 DD Statement: defines a mountable device on which the receiving volume is mounted.
- The SYSIN DD Statement: defines the control data set which follows in the input stream.
- The COPY Statements: copy the source data sets onto volume 231401.

Note: This example implies that the cataloged source data sets all exist on a 2314 volume; that is, when the catalog is searched, these data sets are found to exist on a 2314 volume. If the catalog search had found that the data sets existed on a volume or volumes other that a 2314, the necessary DD statements would have had to be included in this example to define the applicable mountable device(s).

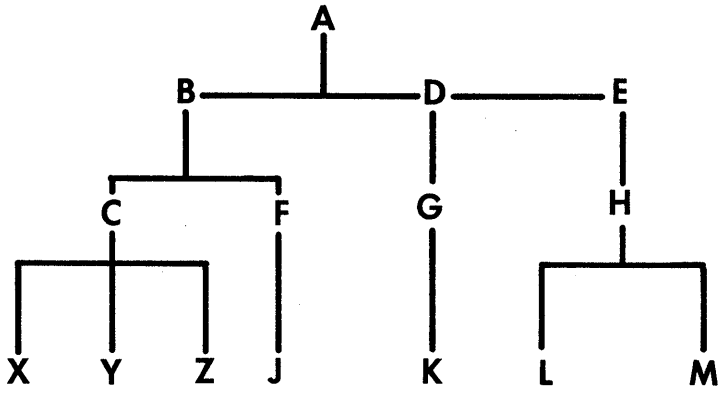
```
//COPYPDS  JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=2311,VOLUME=SER=231100,DISP=OLD
//DD1      DD   UNIT=2301,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=2314,VOLUME=SER=231400,DISP=OLD
//DD3      DD   UNIT=2314,VOLUME=SER=231401,DISP=OLD
//SYSIN    DD   *
         COPY  DSNAME=SEQSET1,TO=2314=231401
         COPY  DSNAME=SEQSET3,TO=2314=231401
         COPY  DSNAME=SEQSET4,TO=2314=231401
/*
```

IEHMOVE Example 2. Copying Three Cataloged Sequential Data Sets Onto a Volume

IEHMOVE Example 3

Operation	Source Data	Devices Required	Comments
MOVE a group of cataloged data sets	Data set group	1 2311 DISK 1 2301 DRUM (system residence) 3 2314 DISKS (mountable)	1. Space is allocated by the IEHMOVE program.

Given: the catalog structure



In this example, the data set group A.B.C (data sets A.B.C.X, A.B.C.Y, and A.B.C.Z) is moved from two 2314 volumes (231410 and 231411) onto a third volume (231401). The catalog is updated to refer to the receiving volume. The source data sets are scratched.

- The SYSUT1 DD Statement: defines the device that is to contain the work data set.
- The DD1 DD Statement: defines the system residence device.
- The DD2 DD Statement: defines the mountable device on which the receiving volume is to be mounted.
- The DD3 DD Statement: defines a mountable device on which one of the source volumes is to be mounted.
- The DD4 DD Statement: defines a mountable device on which one of the source volumes is to be mounted.
- The SYSIN DD Statement: defines the control data set which follows in the input stream.
- The MOVE Utility Control Statement: moves the specified data sets to volume 231401.

Note: This example can be used to produce the same result without the use of the DD4 DD statement; i.e., using one less mountable 2314 device.

- With DD3 and DD4: both of the source volumes are mounted at the start of the job.
- With DD3 only: The 231410 volume is mounted at the start of the job. After the 231410 volume is processed, the utility requests that the operator mount the 231411 volume. (In this case the DD3 statement would be coded:

```
//DD3      DD UNIT=(2314,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(231410))
```

```
//MOVEDSG  JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=2311,VOLUME=SER=231101,DISP=OLD
//DD1      DD   UNIT=2301,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=2314,VOLUME=SER=231401,DISP=OLD
//DD3      DD   UNIT=2314,VOLUME=SER=231410,DISP=OLD
//DD4      DD   UNIT=2314,VOLUME=SER=231411,DISP=OLD
//SYSIN    DD   *
         MOVE  DSGROUP=A.B.C,TO=2314=231401
/*
```

IEHMOVE Example 3. Moving a Data Set Group

IEHMOVE Example 4

Operation	Source Data	Devices Required	Comments
MOVE PDS (merge)	2 partitioned data sets	1 2301 DRUM 1 2311 DISK (system residence) 3 2314 DISKS (mountable)	1. Space is allocated by the IEHMOVE program.

In this example, a partitioned data set (PARTSET1) is to be moved to a disk volume (231420). 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.

- The SYSUT1 DD Statement: defines the device that is to contain the work data set.
- The DD1 DD Statement: 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.
- The SYSIN DD Statement: defines the control data set which follows in the input stream.
- The MOVE statement: defines the source partitioned data set, the volume that contains it, and its receiving volume.
- The INCLUDE Statement: Includes a member from a second partitioned data set in the operation.

```
//MOVEPDS  JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=2301,VOLUME=SER=230100,DISP=OLD
//DD1      DD   UNIT=2311,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=2314,VOLUME=SER=231400,DISP=OLD
//DD3      DD   UNIT=2314,VOLUME=SER=231410,DISP=OLD
//DD4      DD   UNIT=2314,VOLUME=SER=231420,DISP=OLD
//SYSIN    DD   *
         MOVE   PDS=PARTSET1,TO=2314=231420,FROM=2314=231400
      INCLUDE   DSNAME=PARTSET2,MEMBER=PARMEM3,FROM=2314=231410
/*
```

IEHMOVE Example 4. Moving a Partitioned Data Set and Including an Additional Member

IEHMOVE Example 5

Operation	Source Data	Devices Required	Comments
MOVE CATALOG	Catalog on system residence volume	1 2314 1 2301 (system residence) 1 2311 (mountable)	1. Space is allocated by the IEHMOVE program. 2. The source catalog is scratched from the system residence volume.

In this example, the catalog (SYSCTLG data set) is to be moved from the system residence volume to a mountable 2311 disk volume (222222). The source catalog is scratched from the system residence volume.

- The SYSUT1 DD Statement: defines the device that is to contain the work data set.
- The DD1 DD Statement: defines the system residence device. The system residence volume contains the catalog to be moved.
- The DD2 DD Statement: defines the mountable device on which the receiving volume is to be mounted.
- The SYSIN DD Statement: defines the control data set which follows in the input stream.
- The MOVE Statement: specifies the move operation and defines the receiving volume.

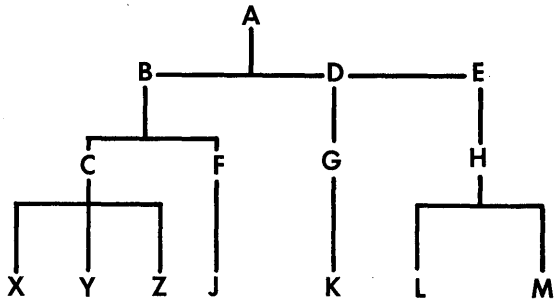
```
//MOVECAT1 JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=2314,VOLUME=SER=231400,DISP=CLD
//DD1      DD   UNIT=2301,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=2311,VOLUME=SER=222222,DISP=OLD
//SYSIN    DD   *
         MOVE  CATALOG,TO=2311=222222
/*
```

IEHMOVE Example 5. Moving the Catalog From the System Residence Volume to a Second Volume

IEHMOVE Example 6

Operation	Source Data	Devices Required	Comments
MOVE a portion of the catalog	Catalog on system residence volume	1 2301 (system residence) 1 2314 (mountable/work)	1. Space is allocated by the IEHMOVE program. 2. The work data set is deleted from the receiving volume at the completion of the program. 3. The entire SYSCTLG data set is scratched.

Given: the catalog structure



In this example, the entries A.B.C.X, A.B.C.Y, and A.B.C.Z are to be moved from the catalog (SYSCTLG data set) to a mountable 2314 volume (231402). If no catalog exists on the 2314 volume, one is created; if a catalog does exist, the specified entries are merged into it. The source SYSCTLG data set is scratched.

- The SYSUT1 DD Statement: defines the device that is to contain the work data set. (Since the IEHMOVE program deletes the work data set at the completion of the program, it can be contained on the receiving volume, provided there is room for it.)
- The DD1 DD Statement: defines the system residence device. The system residence volume contains the entries to be moved.
- The DD2 DD Statement: defines the mountable device on which the receiving volume is to be mounted.
- The SYSIN DD Statement: defines the control data set which follows in the input stream.
- The MOVE Statement: specifies a move operation for a selected entries, and defines the receiving volume.

```
//MOVECAT2 JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=2314,VOLUME=SER=231402,DISP=OLD
//DD1      DD   UNIT=2301,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=2314,VOLUME=SER=231402,DISP=OLD
//SYSIN    DD   *
         MOVE  CATALOG=A.B.C,TO=2314=231402
/*
```

IEHMOVE Example 6. Moving Selected Catalog Entries From the System Residence Volume to a Second Volume

IEHMOVE Example 7

Operation	Source Data	Devices Required	Comments
MOVE VOLUME	Volume of data sets	1 2301 DRUM (system residence) 2 2314 DISKS (mountable/work)	1. Space is allocated by the IEHMOVE program. 2. The work data set is deleted from the receiving volume at the completion of the program.

In this example, a volume of data sets is to be moved to a 2314 volume (231400). All data sets that are successfully moved are scratched from the source volume; however, catalog entries (if any) pertaining to those data sets are not changed.

- The SYSUT1 DD Statement: defines the device that is to contain the work data set. The work data set is removed from the receiving volume at the completion of the job step.
- The DD1 DD Statement: defines the system residence device.
- The DD2 DD Statement: defines the mountable device on which the receiving volume is to be mounted.
- The DD3 DD Statement: defines a mountable device on which the source volume is to be mounted.
- The SYSIN DD Statement: defines the control data set which follows in the input stream.
- The MOVE Statement: 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: The IEHPROGM system utility program can be used to uncatalog those catalog entries pertaining to source data sets, and to catalog the moved versions of those data sets, if desired.

```
//MOVEVOL  JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=2314,VOLUME=SER=231400,DISP=OLD
//DD1      DD   UNIT=2301,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=2314,VOLUME=SER=231400,DISP=OLD
//DD3      DD   UNIT=2314,VOLUME=SER=231401,DISP=OLD
//SYSIN    DD   *
         MOVE  VOLUME=2314=231401,TO=2314=231400,PASSWORD
/*
```

IEHMOVE Example 7. Moving a Volume of Data Sets

IEHMOVE Example 8

Operation	Source Data	Devices Required	Comments
MOVE a data set	a partitioned data set	1 2301 DRUM (system residence) 2 2314 DISKS (mountable/work)	1. Space is allocated on the receiving volume through the use of a dummy execution of the IEHPROGM program. 2. The work data set is deleted from the receiving volume at the completion of the program.

In this example, a partitioned data set is to be moved onto a 2314 disk volume on which space has been previously allocated for the data set. The source data set is scratched.

- The First 8 Job Control Statements: are used to allocate space for data set PDSSET1 on a 2314 disk volume.
- The SYSUT1 DD Statement: 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.
- The DD1 DD Statement: defines the system residence device.
- The DD2 DD Statement: defines the device on which the receiving volume is to be mounted.
- The DD3 DD Statement: defines a mountable device on which the source volume is to be mounted.
- The SYSIN DD Statement: defines the control data set which follows in the input stream.
- The MOVE Statement: specifies a move operation for the partitioned data set PDSSET1 and defines the source and receiving volumes.

```
//ALLOCATE JOB  09#550,GREEN
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//SET1     DD   DSNAME=PDSSET1,UNIT=2314,DISP=(NEW,KEEP),
//              VOLUME=SER=231401,SPACE=(TRK,(100,10,10)),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//SYSIN    DD   *
/*
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=2314,VOLUME=SER=231401,DISP=OLD
//DD1      DD   UNIT=2301,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=2314,VOLUME=SER=231401,DISP=OLD
//DD3      DD   UNIT=2314,VOLUME=SER=231402,DISP=OLD
//SYSIN    DD   *
         MOVE  PDS=PDSSET1,TO=2314=231401,FROM=2314=231402
/*
```

IEHMOVE Example 8. Moving a Data Set Onto a Volume on Which Space Has Been Previously Allocated

IEHMOVE Example 9

Operation	Source Data	Devices Required	Comments
MOVE 3 data sets	3 Partitioned data sets	1 2301 DRUM (system residence) 1 2311 DISK (mountable/ work) 1 2314 DISK (mountable)	1. Space is allocated on the receiving volume through use of a dummy execution of the IEHPROGM program. 2. The source data set PDSSET3 is unloaded. (The record size exceeds the track capacity of the receiving volume.) 3. The work data set is deleted from the receiving volume at the completion of the program.

In this example three partitioned data sets are to be moved from three separate source volumes onto a 2311 disk volume (231101). The source data sets are scratched.

- The Job Control Statements Prior to the EXEC PGM=IEHMOVE Statement: are used to allocate space for the partitioned data sets PDSSET1, PDSSET2, and PDSSET3 on the receiving volume.

  Note: 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 SYSUT1 DD Statement: defines the device that is to contain the work data set.

- The DD1 DD Statement: defines the system residence device.

- The DD2 DD Statement: defines a mountable device on which the source volumes are mounted as they are required.

- The DD3 DD Statement: defines a mountable device on which the receiving volume is mounted.

- The SYSIN DD Statement: defines the control data set which follows in the input stream.

- The MOVE Statements: specify move operations for the partitioned data sets and define the source and receiving volumes.

Note: For a discussion on estimating space allocations, refer to the publication IBM System/360 Operating System: Supervisor and Data Management Services, Form C28-6646.

```
//ALLOCATE JOB  09#550,GREEN
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//SET1     DD   DSNAME=PDSSET1,UNIT=2311,DISP=(NEW,KEEP),
//              VOLUME=SER=231101,SPACE=(TRK,(100,10,5)),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=1600)
//SET2     DD   DSNAME=PDSSET2,UNIT=2311,DISP=(NEW,KEEP),
//              VOLUME=SER=231101,SPACE=(TRK,(50,5,5)),
//              DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SET3     DD   DSNAME=PDSSET3,UNIT=2311,DISP=(NEW,KEEP),
//              VOLUME=SER=231101,SPACE=(TRK,(50,5)),
//              DCB=(RECFM=U,BLKSIZE=5000)
//SYSIN    DD   *
/*
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=2311,VOLUME=SER=231101,DISP=CLD
//DD1      DD   UNIT=2301,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=(2314,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(231400))
//DD3      DD   UNIT=2311,VOLUME=SER=231101,DISP=OLD
//SYSIN    DD   *
         MOVE   PDS=PDSSET1,TO=2311=231101,FROM=2314=231400
         MOVE   PDS=PDSSET2,TO=2311=231101,FROM=2314=231401
         MOVE   PDS=PDSSET3,TO=2311=231101,FROM=2314=231402
/*
```

IEHMOVE Example 9. Moving and Unloading Data Sets Onto a Volume on Which Space Has Been Previously Allocated

IEHMOVE Example 10

Operation	Source Data	Devices Required	Comments
MOVE (unload) a data set	a sequential data set	1 2311 DISK (system residence) 1 2314 DISK (mountable/ work) 1 2400 TAPE (mountable)	1. The source data set is unloaded onto a 9-track, unlabeled magnetic tape volume (800 bpi). 2. The work data set is placed on the source volume and deleted at the completion of the program.

In this example, a sequential data set is to unloaded onto a 9-track, unlabeled magnetic tape volume (800 bpi). The source data set is scratched.

- The SYSUT1 DD Statement: defines the device that is to contain the work data set.
- The DD1 DD Statement: defines the system residence device.
- The DD2 DD Statement: defines a mountable device on which the source volume is mounted.
- The TAPEOUT DD Statement: defines a mountable device on which the receiving tape volume is mounted. This statement also provides label and mode information.
- The SYSIN DD Statement: defines the control data set which follows in the input stream.
- The MOVE Statement: moves the sequential data set SEQSET1 from a 2314 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.

```
//UNLOAD   JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=2314,VOLUME=SER=231400,DISP=OLD
//DD1      DD   UNIT=2311,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=2314,VOLUME=SER=231400,DISP=OLD
//TAPEOUT  DD   UNIT=2400,VOLUME=SER=SCRTCH,DISP=OLD,
//              LABEL=(,NL),DCB=(DEN=2,RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN    DD   *
         MOVE   DSNAME=SEQSET1,TO=2400=SCRTCH,FROM=2314=231400,        C
               TODD=TAPEOUT
/*
```

IEHMOVE Example 10. Unloading a Sequential Data Set Onto an Unlabeled, 9-Track Magnetic Tape Volume

IEHMOVE Example 11

Operation	Source Data	Devices Required	Comments
MOVE (load) unloaded data sets	3 sequential data sets in unloaded form	1 2301 DRUM (system residence) 1 2314 DISK (mountable/ work) 1 2400-2 TAPE (mountable)	1. Space is allocated by the IEHMOVE program. Default space allocations are made if no space attributes are available. 2. The original organization of the unloaded data sets is sequential. 3. The data sets are processed in the order in which they exist on the source volume.

In this example, three unloaded sequential data sets are to be loaded from a labeled, 7-track magnetic tape volume (556 bpi) onto a 2314 disk volume. The example assumes that the 2314 volume is capable of supporting the data sets in their original forms.

- The SYSUT1 DD Statement: defines the device that is to contain the work data set.
- The DD1 DD Statement: defines the system residence device.
- The DD2 DD Statement: defines a mountable device on which the receiving volume is mounted.
- The TAPESETS DD Statement: defines a mountable device on which the source volume is mounted. DCB information is provided in this statement.
- The SYSIN DD Statement: defines the control data set which follows in the input stream.
- The MOVE Statements: move 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/TO=device=list parameter on the utility control statement.

```
//LOAD     JOB  09#550,GREEN
//          EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=2314,VOLUME=SER=231400,DISP=OLD
//DD1      DD   UNIT=2301,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=2314,VOLUME=SER=231400,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=2314=231400,FROM=2400-2=(001234,1), C
                FROMDD=TAPESETS
         MOVE   DSNAME=UNLDSET2,TO=2314=231400,FROM=2400-2=(001234,2), C
                FROMDD=TAPESETS
         MOVE   DSNAME=UNLDSET3,TO=2314=231400,FROM=2400-2=(001234,3), C
                FROMDD=TAPESETS
/*
```

IEHMOVE Example 11. Loading Unloaded Data Sets From a Single Source Volume

IEHMOVE Example 12

Operation	Source Data	Devices Required	Comments
COPY	2 Sequential data sets	1 2314 DISK (mountable/ work) 1 2311 DISK (system residence) 1 2400 TAPE (mountable)	1. Space is allocated by the IEHMOVE program. Default space allocations are made if no space attributes are available. 2. This example assumes that only 1 2400 Magnetic Tape Drive is available for use (deferred mounting is implied).

In this example, two sequential data sets are to be copied from separate source volumes onto a 2314 disk volume. Only one 9-track magnetic tape drive is available for the operation.

- The SYSUT1 DD Statement: defines the volume that is to contain the work data set.
- The DD1 DD Statement: defines the system residence device.
- The DD2 DD Statement: defines a mountable device on which the receiving volume is mounted.
- The TAPE1 DD Statement: defines a mountable device (9-track tape volume) on which the first volume to be processed is mounted. The source data set is the second data set on the volume.
- The TAPE2 DD Statement: 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.
- The SYSIN DD Statement: defines the control data set which follows in the input stream.
- The COPY Statements: copy the data sets onto 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/TO=device=list parameter on the utility control statement.

```
//DEFER    JOB  09#550,GREEN
//         EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=2314,VOLUME=SER=231400,DISP=OLD
//DD1      DD   UNIT=2311,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=2314,VOLUME=SER=231400,DISP=OLD
//TAPE1    DD   DSNAME=SEQSET1,UNIT=2400,DISP=OLD,LABEL=(2,SL),
//             VOLUME=SER=001234,DCB=(DEN=2,RECFM=U,BLKSIZE=2000)
//TAPE2    DD   DSNAME=SEQSET9,UNIT=AFF=TAPE1,DISP=OLD,LABEL=(4,SL),
//             VOLUME=SER=001235,DCB=(DEN=2,RECFM=FB,LRECL=80,
//             BLKSIZE=400)
//SYSIN    DD   *
         COPY   DSNAME=SEQSET1,TO=2314=231400,FROM=2400=(001234,2),    C
                FROMDD=TAPE1
         COPY   DSNAME=SEQSET9,TO=2314=231400,FROM=2400=(001235,4),    C
                FROMDD=TAPE2
/*
```

IEHMOVE Example 12. Copying Data Sets From Separate Source Volumes

IEHMOVE Example 13

Operation	Source Data	Device Required	Comments
Copy	Unloaded Partitioned data sets	1 2400 (mountable) 1 2311 (mountable) 1 2311 (mountable/work)	1. The tape volume contains three unloaded partitioned data sets. 2. The units are allocated specifically, not generically.

In this example, three unloaded partitioned data sets residing on a nonlabeled tape volume mounted on device 282 are copied to a 2311 disk volume mounted on device 191.

- The SYSUT1 DD Statement: defines the work data set.
- The TAPE1 DD Statement: defines the source data sets. They are, in the order in which they reside on the volume, DSET1,DSET2, and DSET3.
- The DD1 DD Statement: defines the receiving volume.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream.
- The COPY Statements: Copy the unloaded partitioned data sets from the nonlabeled tape to the receiving volume.

Note: To copy data sets from a nonlabeled tape, you must place a label in the list field of the FROM=device=list parameter of the utility control statement. Following this label, the sequence numbers of the data sets must also be included in this field. The unit address must appear in the device field of the FROM/TO=device=list parameter whenever you want to move or copy from or to a specific device.

```
//LOAD     JOB  MEDDAUGH,PS40300439,MSGLEVEL=1
//        EXEC PGM=IEHMOVE
//SYSPRINT DD   SYSOUT=A
//SYSABEND DD   SYSOUT=A
//SYSUT1   DD   UNIT=191,VOLUME=231100,DISP=OLD
//DD1      DD   UNIT=191,VOLUME=231100,DISP=OLD
//TAPE1    DD   UNIT=282,VOLUME=SER=NLTAPE,DISP=OLD,LABEL=(,NL)
//SYSIN    DD   *
         COPY   PDS=DSET1,FROM=282=(NLTAPE,1),TO=191=231100,FROMDD=TAPE1
         COPY   PDS=DSET2,FROM=282=(NLTAPE,2),TO=191=231100,FROMDD=TAPE1
         COPY   PDS=DSET3,FROM=282=(NLTAPE,3),TO=191=231100,FROMDD=TAPE1
/*
```

IEHMOVE Example 13. Loading Unloaded Data Sets From Nonlabeled Tape to a Specific Device

Program Applications

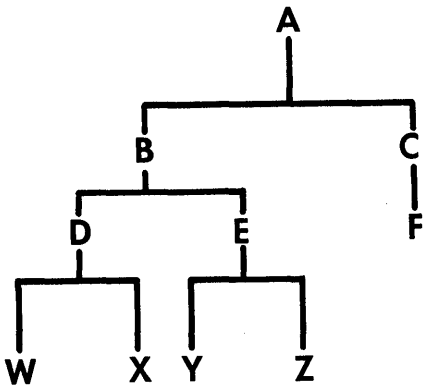
The IEHLIST program can be used to list:

- Entries in a catalog.
- Entries in the directory of one or more partitioned data sets.
- Entries in a volume table of contents.

Any number of listings can be requested in a single job.

Listing Catalog Entries

The IEHLIST program lists all catalog entries that are part of the structure of a fully qualified data set name. For example, given the index structure:

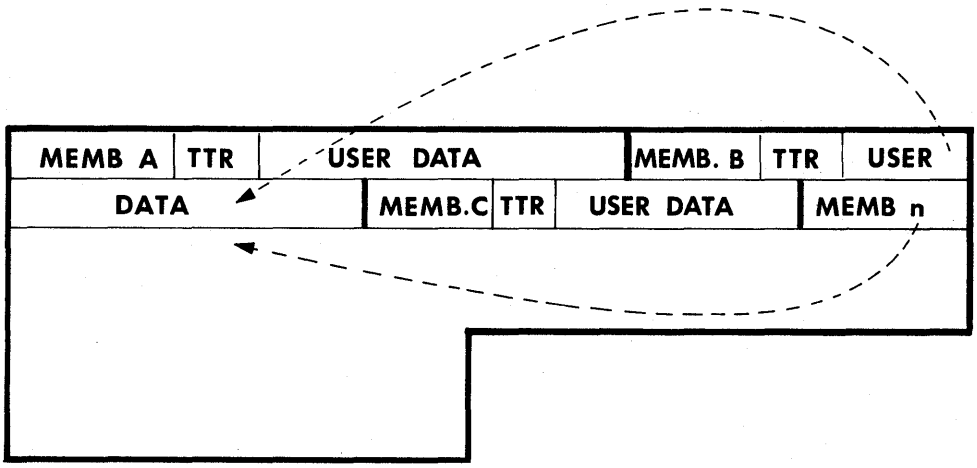


The program lists fully qualified names A.B.D.W, A.B.D.X, A.B.E.Y, and A.B.E.Z. Since A.C.F does not represent a cataloged data set (i.e., the lowest level of qualification has been deleted), it is not a fully qualified name, and it is not listed.

Listing a Partitioned Data Set Directory

The IEHLIST program can list up to 10 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.

A directory block as it exists in storage (see IEHLIST Figure 1) might contain:



IEHLIST Figure 1. A Sample Directory Block

Edited Format: The IEHLIST program optionally provides the following information which is obtained from the applicable partitioned data set directory.

- Member name.
- Entry point.
- Attributes.
- Relative Address of start of member.
- Relative address of start of text.
- Contiguous main storage requirements.
- Length of first block of text.
- Origin of first block of text.
- System status indicators.
- Other information.

In addition, prior to printing the directory entries on the first page, an index is printed explaining the asterisk (*) following a member name, the "attributes" (field 3) and "other information" (field 9). Under the ATTRIBUTE INDEX, the meaning of each attribute bit is explained; under the OTHER INFORMATION INDEX, scatter and overlay format data is described, positionally, as they appear in the listing.

Each directory entry occupies one printed line, except when the member name is an alias and the main member name and associated entry point is available. When this occurs, two lines are used, and every alias name is followed by an asterisk (*).

Note: The FORMAT option applies only to a partitioned data set whose members have been created by the Linkage Editor (i.e., 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.

IEHLIST 2 shows an edited entry for a partitioned member (IEANUC01). The entry is shown as it is listed by the IEHLIST program.

```
MEMBER     ENTRY    ATTR  REL  ADDR-HEX   CONTIG     LEN lst   ORG lst   SSI          OTHER
NAME       PT-HEX   HEX   BEGIN  lstTXT   STOR-DEC   TXT-DEC   TXT-HEX   INFO       INFORMATION

IEANUC01   000000   0662  000003 000104   00035643   01024     000000    ABSENT SCTR=000102,00168,00316,04,04
```

IEHLIST Figure 2. Edited Partitioned Directory Entry

Unedited (Dump) Format: The user may choose the unedited format. If this is the case, the IEHLIST program lists each member separately. IEHLIST Figure 3 shows how the information in Figure 1 is listed.

MEMB A	TTR	USER DATA
MEMB B	TTR	USER DATA
MEMB C	TTR	USER DATA
MEMB n	TTR	USER DATA

IEHLIST Figure 3. A Sample Partitioned Directory Listing

Note: A listing such as that shown in Figure 1 can be obtained by using the IEBPTPCH data set utility program.

To correctly interpret user data information, the user must know the format of the partitioned entry. The formats of directory entries are discussed in the publication IBM System/360 Operating System: System Control Blocks, GC28-6628.

Listing a Volume Table of Contents

The IEHLIST program can 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.

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.
- Record formats.

A VTOC consists of various DSCBs containing the information about the data sets residing on the volume. There are six types of DSCBs which may be contained in a VTOC. They are the FORMAT 1 DSCB, FORMAT 2 DSCB, FORMAT 3 DSCB, FORMAT 4 DSCB, FORMAT 5 DSCB, and FORMAT 6 DSCB. Each type of DSCB serves a particular purpose for the VTOC.

The first DSCB in a VTOC is always a 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 FORMAT 4 DSCB is augmented, when necessary, by the FORMAT 5 DSCB which contains additional information describing the space available for allocation to other data sets.

Following the FORMAT 4 (and FORMAT 5) DSCBs are the FORMAT 1 DSCBs. Each 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 (IS) organization, additional characteristics are specified in a FORMAT 2 DSCB pointed to by the FORMAT 1 DSCB.

Additional extents are described in a FORMAT 3 DSCB pointed to by the FORMAT 1 DSCB (or the FORMAT 2 DSCB when the data set is IS organization).

A FORMAT 6 DSCB is used for shared cylinder allocation. It describes the extent of space (one or more contiguous cylinders) that are being shared by two or more data sets. The FORMAT 6 DSCB is pointed to by the FORMAT 4 DSCB.

A sample listing of the edited format is presented in IEHLIST Figure 4. 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 other edited format is an abbreviated description of the data sets. It is provided by default whenever no other format is requested specifically. It provides the following information:

- Data set name.
- Creation date.
- Expiration date.
- 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.

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.

Refer to the publication IBM System/360 Operating System: System Control Blocks GC28-6628 for a discussion of the various formats that data set control blocks can assume.

Note: For easier reading of unedited VTOC listings, refer to the transparent VTOC Overlay, order number MO 80033.

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
	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)
	154 16	10	30	200 0		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 ADDR	FULL CYLS	A	TRK ADDR	FULL CYLS	A	TRK ADDR	FULL CYLS	A	TRK ADDR	FULL CYLS	A	TRK ADDR	FULL CYLS	A	TRK ADDR	FULL 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 NO EXT DSORG RECFM OPTCD BLKSIZE

EXAMPLE.OF.COMBINED.FORMATS.ONE.AND.TWO	1	EXAMPL	1	36699	27469	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)/L2MEN(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) /NOLEV /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 NO EXT DSORG RECFM OPTCD BLKSIZE

EXAMPLE.OF.COMBINED.FORMATS.ONE.AND.THREE	1	EXAMPL	1	36699	27069	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

IEHLIST Figure 4. Sample Printout of a Volume Table of Contents

Inputs and Outputs

IEHLIST Table 1 shows the major inputs to and outputs from the IEHLIST program.

IEHLIST Table 1. Data Sets Used (Input) and Produced (Output) by the IEHLIST Program

Inputs	Source Data Set(s): This data set (or sets) contains the information to be listed. Source data can be found in three places: • A VTOC data set. • A partitioned data set or sets. • A catalog (SYSCTLG) data set. Control Data Set: This data set contains utility control statements that are used to control the functions of the program.
Output	Output/Message Data Set: This data set is the result of the IEHLIST operations. It includes the listed data and error messages, if applicable.

ADDITIONAL OUTPUTS

The IEHLIST program produces a return code to indicate the results of program execution. The return codes and their interpretations are as follows:

00 -- successful completion.

08 -- due to an error condition, a specified request was ignored. Processing continues.

16 -- an unrecoverable error occurred while reading the data set. The request is terminated.

Control

The IEHLIST program is controlled by job control statements and utility control statements. The job control statements are used to:

- Execute or invoke the program.
- Define the output/message data set and the control data set.
- Define a device or devices to be used during the course of program execution.
- 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 to be modified.

JOB CONTROL STATEMENTS

IEHLIST Table 2 shows the job control statements necessary for executing or invoking the IEHLIST program.

IEHLIST Table 2. Job Control Statements for the IEHLIST Program (Part 1 of 2)

Statement	Usage
JOB statement	This statement initiates the job.
EXEC statement	This statement specifies the program name (PGM=IEHLIST) or, if the job control statements reside in a procedure library, the procedure name.
SYSPRINT DD statement	This statement defines a sequential output/message data set. The data set can be written onto a system output device, a magnetic tape volume, or a direct access volume.
//anyname* DD UNIT=xxxx,VOLUME=SER=xxxxxx,DISP=OLD This statement defines a permanently mounted volume. One statement must be included for each permanently mounted volume referred to in the job step. (The system residence volume is considered to be permanently resident.) In this 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.	
//anyname** DD UNIT=xxxx,VOLUME=SER=xxxxxx,DISP=OLD This statement defines a mountable device type. One statement must be included for each mountable device to be used in the job step. For those applications in which deferred mounting is necessary, it can be specified by including: VOLUME=(PRIVATE,...) and UNIT=(xxxx,,DEFER) in the DD statement defining the mountable device. Refer to Appendix D for information on defining mountable devices.	
SYSIN DD statement	This statement defines the control data set. The data which contains utility control statements, usually follows the job control language in the input stream; however, it can alternatively be defined as being an unblocked sequential data set or member of a procedure library.
*This statement is arbitrarily assigned the ddname DD1 in the IEHLIST examples. **This statement is arbitrarily assigned the ddname DD2 in the IEHLIST examples. DD statements defining additional mountable devices are assigned ddnames DD3, DD4, ... etc.	

(Part 1 of 2)

IEHLIST Table 2. Job Control Statements for the IEHLIST Program (Part 2 of 2)

The blocksize for the SYSPRINT (message) data set must be a multiple of 121. The blocksize for the SYSIN (control) data set must be a multiple of 80. Any blocking factor can be specified for these blocksizes. 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. Since the IEHLIST program 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.) When the IEHLIST program is dynamically invoked in a job step containing another program, the DD statements defining mountable devices for the IEHLIST program must be included in the job stream prior to DD statements defining data sets required by the other program.

UTILITY CONTROL STATEMENTS

Combinations of the following utility control statements are used to control the functions of the program.

- The LISTCTLG (list the catalog) statement.
- The LISTPDS (list a partitioned directory) statement.
- The LISTVTOC (list a volume table of contents) statement.

The 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.

Name	Operation	Operand
[name]	LISTCTLG	[VOL=device=serial] [NODE=name]

VOL=device=serial
: specifies the device type and volume serial number of the control volume on which the specified portion of the catalog resides.

  If VOL is omitted, the catalog is assumed to reside on the system residence volume.

NODE=name
: specifies a qualified name. All data set entries whose names are qualified by this name are listed.

  If NODE is omitted, all data set entries are listed.

The 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.

Name	Operation	Operand
[name]	LISTPDS	DSNAME=(namelist) [VOL=device=serial] {DUMP} {FORMAT}

DSNAME=(name [,name]...)
specifies the fully qualified names of the partitioned data sets whose directories are to be listed. A maximum of 10 names is allowed. If the list consists of a single name, the parentheses can be deleted.

VOL=device=serial
specifies the device type and volume serial number of the volume on which the partitioned data sets reside.

If VOL is omitted, the data sets are assumed to reside on the system residence volume.

DUMP
specifies that the listing is to be in unedited, hexadecimal form.

FORMAT
specifies that the listing is to be edited for each directory entry into the following fields:

- Member name.
- Entry point.
- Attributes.
- Relative address of start of member.
- Relative address of start of text.
- Contiguous main storage requirements.
- Length of first block of text.
- Origin of first block of text.
- System status indicators.
- Other information.

In addition, prior to printing the directory entries on the first page, an index is printed explaining the "attributes" (field 3) and "other information" (field 9). ATTRIBUTE INDEX explains each attribute bit; OTHER INFORMATION INDEX explains scatter and overlay format data as it appears in the listing.

Note: This option 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 LISTVTOC Statement

The LISTVTOC statement is used to request a partial or complete listing of the entries in a specified volume table of contents.

Name	Operation	Operand
[name]	LISTVTOC	{ DUMP }
		{ FORMAT }
		[DATE=dddyy]
		[VOL=device=serial]
		[DSNAME=(namelist)]

DUMP
specifies that the listing is to be in unedited, hexadecimal form.

FORMAT
specifies that a comprehensive edited listing is to be generated.

Note: If FORMAT and DUMP are omitted, an abbreviated edited format will be generated by default.

DATE=dddyy (applicable only to the abbreviated edited format)
specifies that each entry that expires before this date is to be flagged with an asterisk (*) in the listing. The date is represented by:

ddd day of the year.
yy last two digits of the year.

If DATE is omitted, no asterisks appear in the listing.

VOL=device=serial
specifies the device type and volume serial number of the volume whose table of contents is to be listed.

If VOL is omitted, the system residence volume is assumed.

DSNAME=(name[,name]...)
specifies the fully qualified names of the data sets whose entries are to be listed. A maximum of 10 names is allowed.

If DSNAME is omitted, the entire volume table of contents is listed.

IEHLIST Examples

The following examples show some of the uses of the IEHLIST program.

Note: In each of the following 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 cataloged procedure, which is supplied by IBM.

```
//LIST     EXEC PGM=IEHLIST,REGION=44K                                  00000000
//DDSRV     DD  VOLUME=REF=SYS1.SVCLIB,DISP=OLD                         00000010
//SYSPRINT DD   SYSOUT=A                                                00000020
```

IEHLIST Example 1

Operation	Number of Devices Reqired	Comments
List a catalog	1 2311 DISK (mountable) 1 system output device	1. The entire source catalog is listed on the system output device.

In this example a catalog residing on a 2311 disk volume (231100) is to be listed.

- The DD2 DD Statement: defines a mountable device on which the volume containing the source catalog is mounted.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream.
- The LISTCTLG Utility Control Statement: defines the source volume and specifies the list operation.

Note: The data set name of the catalog data set is SYSCTLG.

```
//LISTCAT  JOB  09#550,BLUE
//        EXEC PGM=IEHLIST
//SYSPRINT DD   SYSOUT=A
//DD2      DD   UNIT=2311,VOLUME=SER=231100,DISP=OLD
//SYSIN    DD   *
     LISTCTLG   VOL=2311=231100
/*
```

IEHLIST Example 1. Listing a Catalog

IEHLIST Example 2

Operation	Number of Devices Required	Comments
List 3 catalogs and a portion of a fourth.	1 2314 DISK (mountable) 1 system residence device 1 system output device	1. All applicable data is listed on the system output device

In this example a catalog residing on the system residence volume, two catalogs residing on 2314 disk volumes, and a portion of a catalog residing on a 2314 volume, are to be listed.

- The DD1 Statement: defines a system residence device. (The first catalog to be listed resides on the system residence volume.)
- The DD2 Statement: defines a mountable device on which each 2314 volume is mounted as it is required by the program.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream.
- The First LISTCTLG Utility Control Statement: indicates that the catalog residing on the system control volume is to be listed.
- The Second and Third LISTCTLG Statements: identify two 2314 disk volumes containing catalogs to be listed.
- The Fourth LISTCTLG Statement: identifies a 2314 disk volume containing a catalog that is to be partially copied. All data set entries whose beginning qualifiers are A.B.C are copied.

```
//LISTCATS JOB  09#550,BLUE
//         EXEC PGM=IEHLIST
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=2301,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=(2314,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(231400))
//SYSIN    DD   *
     LISTCTLG
     LISTCTLG   VOL=2314=231400
     LISTCTLG   VOL=2314=231401
     LISTCTLG   VOL=2314=231402,NODE=A.B.C
/*
```

IEHLIST Example 2. Listing a Number of Catalogs

IEHLIST Example 3

Operation	Number of Devices Required	Comments
List three partitioned directories	1 2314 DISK (mountable) 1 system residence device (2301 DRUM) 1 system output device	1. The directories are listed on a system output device.

In this example, a partitioned directory existing on the system residence volume is to be listed. In addition, two partitioned directories existing on a 2314 volume are to be listed.

- The DD1 DD Statement: defines the system residence device.
- The DD2 DD Statement: defines a mountable device on which a 2314 source volume (231400) is to be mounted.
- The SYSIN DD Statement: 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 2314 disk volume (231400).

```
//LISTPDIR JOB  09#550,BLUE
//         EXEC PGM=IEHLIST
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=2301,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=2314,VOLUME=SER=231400,DISP=OLD
//SYSIN    DD   *
      LISTPDS  DSNAME=PARSET1
      LISTPDS  DSNAME=(PART1,PART2),VOL=2314=231400
/*
```

IEHLIST Example 3. Listing Partitioned Directories

IEHLIST Example 4

Operation	Number of Devices Required	Comments
List a volume table of contents	1 2311 DISK (mountable) 1 system output device	1. The specified volume table of contents is listed in edited form, and then, selected data set control blocks are listed in unedited form.

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.

- The DD2 DD Statement: defines a mountable device on which the volume containing the specified volume table of contents is to be mounted.
- The SYSIN DD Statement: defines the control data set which follows in the input stream.
- The First LISTVTOC Utility Control Statement: indicates that the volume table of contents on the specified 2311 disk volume is to be listed in edited form.
- The Second LISTVTOC Utility Control Statement: indicates that the data set control blocks representing data sets SET1, SET2, and SET3 are to be listed in unedited form.

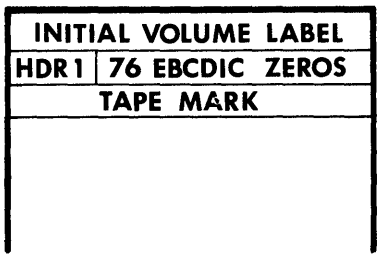
```
//LISTVTOC JOB  09#550,BLUE
//         EXEC PGM=IEHLIST
//SYSPRINT DD   SYSOUT=A
//DD2      DD   UNIT=2311,VOLUME=SER=231100,DISP=OLD
//SYSIN    DD   *
     LISTVTOC  FORMAT,VOL=2311=231100
     LISTVTOC  DUMP,VOL=2311=231100,DSNAME=(SET1,SET2,SET3)
/*
```

IEHLIST Example 4. Listing a Volume Table of Contents

Program Application

The IEHINITT utility program places IBM System/360 Operating System volume label sets onto any number of magnetic tapes mounted on one or more tape drives. Each volume label set created by the program contains:

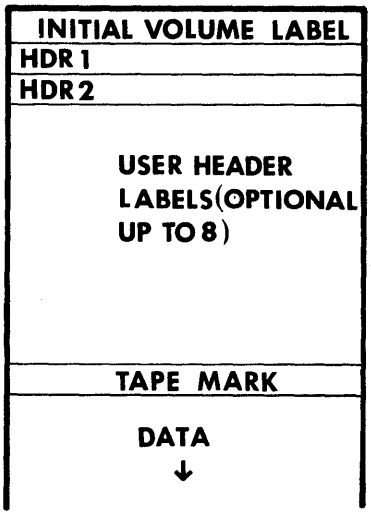
- A standard volume label with user specified serial number and owner identification.
- A dummy header label (an 80-byte record containing HDR1 and 76 EBCDIC zeros).
- A tapemark.



Note: When a labeled tape is subsequently used as a receiving volume:

1. The tapemark created by the IEHINITT program is overwritten.
2. The dummy HDR1 record created by the IEHINITT program 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 by the processing program).
5. A tapemark is written.
6. Data is placed on the receiving volume.

IEHINITT Figure 1 shows a standard label after a volume is first used to receive data. Refer to the publication IBM System/360 Operating System: Supervisor and Data Management Services, GC28-6646, for a discussion on volume labels.



IEHINITT Figure 1. Standard Label After Volume First Receives Data

Placing a Standard Label Set on Magnetic Tape

The IEHINITT program can place labels on 7-track or 9-track magnetic tape volumes. Any number of 7-track and/or 9-track tape volumes can be labeled in a single execution of the program.

Tape volumes are labeled in sequential order by specifying a serial number to be written onto 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.)

The user can provide additional information, such as name and rewind or unload specifications.

The user must supply all tapes to be labeled. Explicit instructions to the operator should be provided with each job request. Otherwise, the operator will not mount the tapes.

If any errors are encountered while attempting to label a tape, the tape is left unlabeled. The program attempts to label any tapes remaining to be processed.

For information on creating routines to write standard or nonstandard labels, refer to the publication IBM System/360 Operating System: System Programmer's Guide, GC28-6550.

Notes: The program writes 7-track magnetic tape labels in even parity. Previously labeled tapes can be overwritten with new labels.

Inputs and Outputs

IEHINITT Table 1 lists the major inputs to and outputs from the IEHINITT program.

IEHINITT Table 1. Data Sets Used (Input) and Produced (Output) by the IEHINITT Program

Input	Control Data Set: This data set contains a utility control statement or statements used to provide control information for the IEHINITT program.
Output	Message Data Set: This data set contains: • Utility program identification. • Initial volume label information for each successfully labeled tape volume. • Contents of the utility control statement or statements that were used. • Error messages, if applicable.

ADDITIONAL OUTPUTS

The IEHINITT program produces a return code to indicate the results of program execution. The return codes and their interpretations are as follows:

00 -- successful completion. A message data set was created.

04 -- successful completion. No message data set was defined by the user.

08 -- the program completed its operation but error conditions were encountered during processing. A message data set was created.

12 -- the program completed its operation but error conditions were encountered during processing. No message data set was defined by the user.

16 -- 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

The IEHINITT program is controlled by job control statements and utility control statements. The job control statements are used to:

- Execute or invoke the program.
- Define the control and message data sets.
- Provide density information for magnetic tapes.

Utility control statements are used to specify applicable label information.

JOB CONTROL STATEMENTS

IEHINITT Table 2 shows the job control statements necessary for executing or invoking the IEHINITT program.

IEHINITT Table 2. Job Control Statements for the IEHINITT Program

Statement	Usage
JOB statement	This statement initiates the job.
EXEC statement	This statement 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 an optional PARM parameter to specify the number of lines to be printed between headings in the message data set. PARM=LINECNT=nn specifies that nn number of lines (00 to 99) are to be printed between headings.[1] If PARM is omitted, 60 lines are printed between headings.
SYSPRINT DD statement	This statement 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.
//name	DD DCB=(DEN=x),UNIT=(xxxx,n,DEFER) This statement defines a magnetic tape drive or drives to be used in a labeling operation. Where: DEN=x specifies the density at which the labels are written. UNIT=(xxxx,n,DEFER) specifies device type, number of drives to be used for the labeling operation, and deferred mounting. "name" must be identical to a name specified in a utility control statement. This relates the specified drive(s) to the appropriate utility control statement.
SYSIN DD statement	This statement defines the control data set. The control data set normally resides in the input stream; however, it can alternatively be defined as a member of a procedure library or as a sequential data set existing somewhere other than the input stream.

The SYSPRINT (message) 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. The SYSIN (control) data set must have a blocksize which is a multiple of 80. Any blocking factor can be specified for these blocksizes.

[1]If the IEHINITT program 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 6-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 the section "Invoking Utility Programs."

UTILITY CONTROL STATEMENTS

The IEHINITT program uses an INITT utility control statement to provide control information for a labeling operation. 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.

The INITT Statement

The INITT statement provides control information for the IEHINITT program.

```
┌──────┬─────────┬──────────────────────────────────────────────────┐
│Name  │Operation│Operand                                           │
├──────┼─────────┼──────────────────────────────────────────────────┤
│name  │INITT    │SER=xxxxxx                                        │
│      │         │[OWNER='cccccccccc']                              │
│      │         │[NUMBTAPE=n]                                      │
│      │         │┌DISP=REWIND┐                                     │
│      │         │└DISP=UNLOAD┘                                     │
└──────┴─────────┴──────────────────────────────────────────────────┘
```

name
: specifies a name that is identical to a ddname in the name field of a DD statement defining a tape drive or drives. This name must begin in column 1 of the utility control statement.

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. 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.

OWNER='cccccccccc'
: specifies the owner's name or similar identification. The information is specified as character constants, and can be up to 10 bytes in length. 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, it must be written as two consecutive apostrophes.

NUMBTAPE=n
: 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 NUMBTAPE is omitted, one tape volume is labeled.

DISP=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.

DISP=UNLOAD
: specifies that a tape is to be unloaded after the label has been written.

IEHINITT Figure 2 shows a printout of a message data set including the INITT control statement and initial volume label information. In this example, one INITT statement was used to place serial numbers of 001122 and 001123 onto two magnetic tape volumes. VOL100112200 and VOL10011230 are interpreted as follows:

- VOL1 indicates that an initial volume label was successfully written onto a tape volume.
- 001122 and 001123 are the serial numbers that were written onto the volumes.
- 0 is unused.

No errors occurred during processing.

```
SYSTEM SUPPORT UTILITIES      IEHINITT

ALL     INITT   SER=001122,NUMBTAPE=2,OWNER='P.T.BROWN',          C
                DISP=REWIND

VOL10011220                   P.T.BROWN
VOL10011230                   P.T.BROWN
```

IEHINITT Figure 2. A Printout of INITT Statement Specifications and Initial Volume Label Information

IEHINITT Examples

The following examples illustrate some of the applications of the IEHINITT program.

IEHINITT Example 1

Operator	Number of Tape Devices Used	Comments
Label 3 tape volumes	1 9-track (2400) tape drive	1. Labels are written at a density of 800 bits per inch.

In this example, serial numbers 001234, 001235, and 001236, are to be placed onto three tape volumes. Each volume to be labeled is mounted, when it is required, on a single 9-track magnetic tape volume.

```
//LABEL1   JOB  09#990,BROWN
//         EXEC PGM=IEHINITT
//SYSPRINT DD   SYSOUT=A
//LABEL    DD   DCB=(DEN=2),UNIT=(2400,1,DEFER)
//SYSIN    DD   *
LABEL   INITT   SER=001234,NUMBTAPE=3
/*
```

IEHINITT Example 1. Labeling Three 9-Track Magnetic Tape Volumes

IEHINITT Example 2

Operation	Number of Tape Devices Used	Comments
Label 2 groups of tape volumes	1 9-track (2400) tape drive	1. Both groups of labels are written at a density of 800 bits per inch.

In this example, two groups of serial numbers (001234, 001235, 001236, and 001334, 001335, 001336) are placed onto six tape volumes. Each volume to be labeled is mounted, when it is required, on a single 9-track magnetic tape volume.

```
//LABEL2   JOB  09#990,BROWN
//         EXEC PGM=IEHINITT
//SYSPRINT DD   SYSOUT=A
//LABEL    DD   DCB=(DEN=2),UNIT=(2400,1,DEFER)
//SYSIN    DD   *
LABEL   INITT   SER=001234,NUMBTAPE=3
LABEL   INITT   SER=001334,NUMBTAPE=3
/*
```

IEHINITT Example 2. Labeling Two Groups of 9-Track Magnetic Tape Volumes

IEHINITT Example 3

Operation	Number of Tape Drives Used	Comments
Label -- increment sequence numbers by 10	4 9-track (2400) tape drives	1. Labels are written at a density of 800 bits per inch.

In this example, serial numbers 001234, 001244, 001254, 001264, 001274, etc., are to placed onto 8 magnetic tape volumes. Each volume to be labeled is mounted, when it is required, on one of four 9-track magnetic tape volumes.

```
//LABEL3   JOB  09#990,BROWN
//         EXEC PGM=IEHINITT
//SYSPRINT DD   SYSOUT=A
//LABEL    DD   DCB=(DEN=2),UNIT=(2400,4,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 3. Labeling 9-Track Volumes -- Serial Numbers Incremented by 10

IEHINITT Example 4

Operation	Number of Tape Drives Used	Comments
Label 3 tape volumes	1 9-track (2400) tape drive 1 9-track (2400-4) tape drive	1. An alphameric label is placed on the 2400 volume. The label is written at a density of 800 bits per inch. 2. Numeric labels are placed on two 2400-4 tape volumes. The labels are written at a density of 1600 bits per inch.

In this example, serial number TAPE1, is to be placed on a 2400 tape volume and serial numbers 001234 and 001235 are to be placed on two 2400-4 tape volumes.

```
//LABEL4   JOB  0.#990,BROWN
//         EXEC PGM=IEHINITT
//SYSPRINT DD   SYSOUT=A
//LABEL1   DD   DCB=(DEN=2),UNIT=(2400,1,DEFER)
//LABEL2   DD   DCB=(DEN=3),UNIT=(2400-4,1,DEFER)
//SYSIN    DD   *
LABEL1  INITT  SER=TAPE1
LABEL2  INITT  SER=001234,NUMBTAPE=2
/*
```

IEHINITT Example 4. Alphameric and Numeric Labeling

Program Applications

The IEHIOSUP program is used to update TTR entries in the transfer control tables of the supervisor call library (SVC library). This program must be used after:

- The SVC library is moved.
- The OPEN, CLOSE, TCLOSE, EOV, FEOV, SCRATCH, ALLOCATE, IEHATLAS, SETPRT, or STOW or any Machine-Check Handler (MCH) recovery management module is changed or replaced in the SVC library.

Updating TTR Entries in the SVC Library

Due to the manner in which the SVC routines are loaded, it is necessary to update TTR entries after changing or replacing a module. This program automatically updates the TTR entries.

Inputs and Outputs

IEHIOSUP Table 1 lists the major input to and output from the IEHIOSUP program.

IEHIOSUP Table 1. Data Sets Used and Produced by the IEHIOSUP Program

Object Data Set: This data set (the SYS1.SVCLIB data set) contains the transfer control tables that are to be updated.
Message Data Set: This data set contains error messages (if any) generated during the execution of the program.

ADDITIONAL OUTPUTS

The IEHIOSUP program produces a return code to indicate the results of program execution. The return codes and their interpretations are as follows:

00 -- successful completion.

12 -- an unrecoverable error has occurred. The job step is terminated.

Control

The IEHIOSUP program is executed or invoked with job control statements. The program function (i.e., the updating of the TTR entries) is performed automatically. (No utility control statements are required.)

JOB CONTROL STATEMENTS

IEHIOSUP Table 2 shows the job control statements necessary for executing or invoking the IEHIOSUP utility program.

IEHIOSUP Table 2. Job Control Statements for the IEHIOSUP Program

Statement	Usage
JOB statement	This statement initiates the job.
EXEC statement	This statement specifies the program name (PGM=IEHIOSUP) or, if the job control statements for the IEHIOSUP program reside in a procedure library, the procedure name.
SYSPRINT DD statement	This statement 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 statement	This statement defines the object (SYS1.SVCLIB) data set. The DSNAME, DISP, UNIT, and VOLUME parameters should be included.
Note: If the SYS1.SVCLIB data set is cataloged, the UNIT and VOLUME parameters are not required in the SYSUT1 DD statement. The blocksize for the SYSPRINT (message) data set must be a multiple of 121. Any blocking factor can be specified.	

IEHIOSUP Examples

The following examples illustrate some of the uses of the IEHIOSUP program.

IEHIOSUP Example 1

Operation	Data Set Organization	I/O Device	Comments
UPDATE the SVC LIBRARY (TTRs)	Object-PARTITIONED Message-SEQUENTIAL	DISK--2311 and System Output Device (Printer Assumed)	1. The SYS1.SVCLIB data set is not cataloged.

In this example, the TTR entries in the SVC library are to be updated.

- The SYSUT1 DD Statement: defines the object data set (the SYS1.SVCLIB data set).
- The SYSPRINT DD Statement: defines the message data set.

```
//TTRUPDTE JOB
//        EXEC PGM=IEHIOSUP
//SYSUT1   DD   DSNAME=SYS1.SVCLIB,DISP=OLD,UNIT=2311,
//             VOLUME=SER=111111
//SYSPRINT DD   SYSOUT=A
//
```

IEHIOSUP Example 1. Updating TTR Entries -- SVC Library is Not Cataloged

IEHIOSUP Example 2

Operation	Data Set Organization	I/O Device	Comments
UPDATE the SVC LIBRARY (TTRs)	Object-PARTITIONED Message-SEQUENTIAL	DRUM--2301 and System Output Device (Printer Assumed)	1. The SYS1.SVCLIB data set is cataloged.

In this example, the TTR entries in the SVC library are to be updated.

- The SYSUT1 DD Statement: defines the object data set (the SYS1.SVCLIB data set). The data set is cataloged; therefore, no UNIT or VOLUME parameter is required.
- The SYSPRINT DD Statement: defines the message data set.

```
//SVCUPDTE JOB
//        EXEC PGM=IEHIOSUP
//SYSUT1   DD   DSNAME=SYS1.SVCLIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//
```

IEHIOSUP Example 2. Updating TTR Entries -- SVC Library is Cataloged

Program Applications

The IFCEREP0 program edits and writes records contained in the SYS1.LOGREC data set. The records were generated by error environment recording programs OBR and SDR, and by recovery management programs SER0, SER1, MCH, TPER, and CCH. The records contain environmental data (machine indicator contents, register contents, etc.) that was stored when an error occurred.

The program can accumulate a history of malfunctions by selecting records from the SYS1.LOGREC data set and placing them in an output (history) data set, or by selecting records from an existing history data set and placing them in a second history data set. In addition, the program can summarize certain types of environmental records; that is, pertinent information can be extracted from selected records and written on an output device (usually the printer).

The program can process five types of environment records:

- Machine-check (CPU) records, produced and stored as a result of machine-check interruptions caused by malfunctions in the central processing unit.
- Channel-check (inboard) records, produced and stored as a result of input/output interruptions caused by specific channel failures.
- I/O device (outboard) records, produced and stored as a result of permanent device errors.
- Statistical data records, produced and stored to maintain a count of input/output device errors.
- T-type records (bulk data records), produced and stored as a result of error data not reflected in any of the other type records.

The program is used to:

- Edit and write selected records.
- Accumulate machine-check, channel inboard, or I/O outboard records and place them in a new or updated history data set.
- Summarize machine-check, channel inboard, or I/O outboard records contained in the SYS1.LOGREC data set or in a history data set.
- Process (edit and write, accumulate, and/or summarize) records produced on a different machine model.

Editing and Writing Selected Records

The IFCEREP0 program retrieves selected environment records contained in the SYS1.LOGREC data set or in a history data set and writes an edited version of the selected records. Editing can be specified as follows:

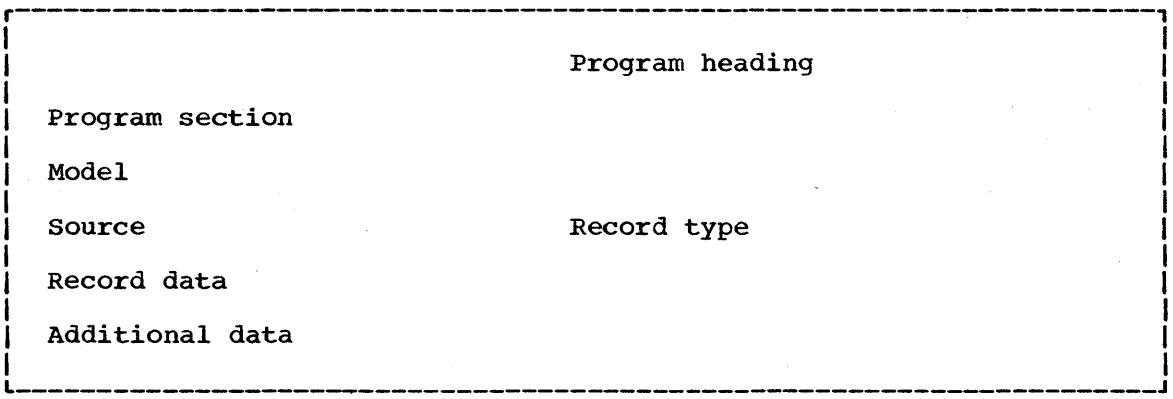
- Machine-check, channel inboard, I/O outboard, or statistical data records (or combinations of these records) can be selected.

- Records can be selected by the model number of the system which created them.
- Records that were written within a specific period of calendar time can be selected.
- I/O outboard or statistical data records related to a specific unit address or device type can be selected.
- Input records can remain uncleared after processing. (The program normally clears each selected record to hexadecimal zeros in the SYS1.LOGREC data set when processing of that record is complete. However, an option can be specified to prevent the clearing of selected records.)

Notes: Those records that are cleared in the SYS1.LOGREC data set cannot be reused by the recording programs until the entire SYS1.LOGREC data set is cleared.

The IFCEREP0 program can write its output on any IBM output device supported by the BSAM access method. The output is written as 121-byte unblocked records with an ASA control character as the first character in each record.

The program can produce CPU (machine-check), channel inboard, I/O outboard, and statistical data printouts on all supported models. A printout for an edited record has the format shown in IFCEREP0 Figure 1.



IFCEREP0 Figure 1. Output Record Printout Structure

Program heading
identifies the IFCEREP0 program on the first page of the listing:

- ENVIRONMENT RECORD EDITING AND PRINTING PROGRAM

Program section
identifies the program section that is generating the printout. Valid program sections are:

- CPU (MC) DATA EDITING AND PRINTING SECTION
- INBOARD DATA EDITING AND PRINTING SECTION
- OUTBOARD DATA EDITING AND PRINTING SECTION
- STATISTICAL DATA EDITING AND PRINTING SECTION
- T-TYPE DATA EDITING AND PRINTING SECTION

Model
identifies the IBM System/360 Model for which the printout is applicable. Valid entries are:

- Model 40, 50, 65, 67, 75, 85, 91, 95, or 195 for machine-check records.
- Model 40, 50, 65, 75, 85, 91, 95, or 195 for channel inboard records. (Model 67 and 95 channel inboard records appear as Model 65 and 91 records, respectively.)
- UNIVERSAL for statistical data or I/O outboard printouts produced by Model 30, 40, 50, 65, 67, 75, 85, 91, 95, or 195.

Source
identifies the error environment or recovery management program that generated the record placed in the SYS1.LOGREC data set. Valid sources are:

- RECORD ENTRY SOURCE - OBR
- RECORD ENTRY SOURCE - SDR
- RECORD ENTRY SOURCE - SER0
- RECORD ENTRY SOURCE - SER1
- RECORD ENTRY SOURCE - MCH
- RECORD ENTRY SOURCE - CCH
- RECORD ENTRY SOURCE - TPER

Record type
indicates the type of printout. Valid types are:

- TYPE - CPU
- TYPE - INBOARD
- TYPE - OUTBOARD
- TYPE - STATISTICAL DATA
- TYPE - T-TYPE

Record data
is a listing of the edited record from the input data set. This data, which constitutes the bulk of the printout, is the programming data and machine data collected at the time of the error.

Additional data
is a listing of records that were recorded in the SYS1.LOGREC data set while the program was being executed.

The heading:

- THE FOLLOWING RE'S WERE GENERATED WHILE EXECUTING EREP

is followed by a printout of the records.

IFCEREP0 Figure 2 shows a sample outboard printout of an environment record that was processed by the outboard data editing and printing section of the utility program. The record was generated by the OBR program on an IBM System/360 Model 30, 40, 50, 65, 67, 75, 85, 91, 95, or 195 (indicated by UNIVERSAL in the printout). The device failure occurred on a 2311 disk with a channel and unit address of 190.

IFCEREP0 Figure 3 shows a sample statistical data printout of an environment record that was processed by the statistical data editing and printing section of the utility program. The record was generated by the SDR recording program on an IBM System/360 Model 30, 40, 50, 65, 67, 75, 85, 91, 95, or 195 (indicated by UNIVERSAL in the printout).

Note: The format for the T-type record is variable and requires special editing modules from the specific sub-types. Because of this variation, no sample printouts are shown for T-type record editing.

```
ENVIRONMENT RECORD EDITING AND PRINTING PROGRAM
OUTBOARD DATA EDITING AND PRINTING SECTION

MODEL-UNIVERSAL
--- RECORD ENTRY SOURCE - OBR  TYPE - OUTBOARD
CHANNEL/UNIT ADDRESS 0190      DEVICE TYPE  2311
PROGRAM IDENTITY IFCEP000      VOLUME LABEL  111111
      DAY  YEAR                      HH MM SS TH
DATE -024   66                TIME- 00 00 38.70
            CC  DA     FL     CT
FIRST CCW   31 0061DB 40 00 0005
FAILING CCW 31 0061DB 40 00 0005
            K   CA     US CS  CT
CSW         00 0061E8 OE 40 0005
 UNIT STATUS                   CHANNEL STATUS
  ATTENTION          0          PRGM-CTLD IRPT     0
  STATUS MODIFIER    0          INCORRECT LENGTH   1
  CONTROL UNIT END   0          PROGRAM CHECK      0
  BUSY               0          PROTECTION CHECK   0
  CHANNEL END        1          CHAN DATA CHECK    0
  DEVICE END         1          CHAN CTL CHECK     0
  UNIT CHECK         1          I/F CTL CHECK      0
  UNIT EXCEPTION     0          CHAINING CHECK     0
LAST SEEK ADDRESS
      M       B        B        C        C        H        H        R
00000000 00000000 00000000 00000000 11100010 00000000 00000100 00000001
SENSE BYTE DATA
   BYTE 0   BYTE 1   BYTE 2   BYTE 3   BYTE 4   BYTE 5
  00000001 00110111 11110000 00010011 01111111 00000001

BYTE 0        BYTE 1        BYTE 2        BYTE 3       BYTE 4    BYTE 5

COMND REJ 0   DATA CHK  0   UNSAFE    1   READY     0 BIT 0 0   COMMAND    0
INTV REQ  0   TRK OVERF 0   BIT 1      1   ON LINE   0 BIT 1 1   IN         0
BUS OUT   0   CYL END   1   SERIAL CH  1   UNSAFE    0 BIT 2 1   PROGRESS   0
EQUIP CHK 0   INV SEQ   1   TAG LINE   1   BIT 3     1 BIT 3 1   WHEN       0
DATA CHK  0   REC UNFND 0   ALU CHK    0   ON LINE   0 BIT 4 1   OVERFLOW   0
OVERRUN   0   FILE PROT 1   UNSEL STA  0   CYL END   0 BIT 5 1   INCMPLETE  0
TRK COND  0   MISG A MK 1   BIT 6      0   BIT 6     1 BIT 6 1   OCCURS     0
SEEK CHK  1   OVFL INC  1   BIT 7      0   SEEK INCR 1 BIT 7 1               1
```

IFCEREP0 Figure 2. Sample Printout -- Outboard Data Editing and Printing Section

```
ENVIRONMENT RECORD EDITING AND PRINTING PROGRAM
STATISTICAL DATA EDITING AND PRINTING SECTION

MODEL-UNIVERSAL
--- RECORD ENTRY SOURCE - SDR ---     TYPE - STATISTICAL DATA
CHANNEL/UNIT ADDRESS  0190            DEVICE TYPE 2311
TEMPY RDS          00003              TEMPY WRT          00000
INTRVN REQD        00000              BUS OUT CHK        00000
EQUIP CHK          00000              OVERRUN            00000
TRK CND            00000              SEEK CHK           00000
UNSAFE             00000
SER/DESER          00000              CHAN TAG LINE      00000
ALU                00000
MISG ADR MKR       00000
```

IFCEREP0 Figure 3. Sample Printout -- Statistical Data Editing and Printing Section

Accumulating Selected Records

Selected machine-check, channel inboard, or I/O outboard records can be moved from the SYS1.LOGREC data set to a history data set or from a history data set to a second history data set. In this manner, an accumulation of specific error conditions can be retained on a selected volume.

Notes: The format of a record in a history data set is identical to the format of a record in the SYS1.LOGREC data set; that is, a record is not altered when it is moved from the SYS1.LOGREC data set to a history data set.

Selected records from an input data set can be written and accumulated in one execution of the program. If the SYS1.LOGREC data set is used as the input data set, those input records that are accumulated in the output are cleared to hexadecimal zeros in the SYS1.LOGREC data set.

Summarizing Selected Records

Selected machine-check, channel inboard, or I/O outboard records from the SYS1.LOGREC data set or from a history data set can be summarized. A summary printout consists of a list of pertinent items extracted from selected records. The frequency of occurrence of each item is included in the summary printout.

Machine-check and channel inboard summaries differ from model to model, due to different hardware configurations; I/O outboard summaries do not differ from model to model.

The program summarizes items that provide, in themselves, clues as to the type of machine malfunction. In general, registers are not summarized; however, if, in themselves, they provide clues as to the type of machine malfunction, they are summarized. The format of a summary depends upon the type of summary.

Machine-Check Summary: A machine-check summary can be generated on IBM System/360 Models 40, 50, 65, 67, 75, 85, 91, 95, and 195. A summary consists of:

- Items that provide clues as to the type of machine malfunction.
- Parity information for (1) registers in the diagnostic scan-out area (logout area), (2) general purpose registers, and (3) floating point registers.
- The status of binary triggers recorded in the logout area.

Note: Model 85 summarizes the error triggers only.

IFCEREP0 Figure 4 shows the format of a machine-check summary. Each summarized item is listed with its frequency of occurrence.

```
*** MOD xx MACHINE-CHECK SUMMARY ***
NUMBER OF RECORDS EXAMINED = 10

   TITLE            TOTAL
ROBAR SUMMARY (UP TO FIRST 10)
   0AAAA              3
   1BBBB              4
   1CCCC              3

LOGOUT REG PARITY CHECK SUMMARY
   REG A              5
   REG B              2
   REG C              3

CHECKS AND INDICATORS SUMMARY
   ROAR CHECK         1
   LSAR PTY CHECK     3
   H DECODE CHECK     4
   D/Y8 CHECK         2
```

IFCEREP0 Figure 4. Machine-Check Summary

Channel Inboard Summary: A channel inboard summary can be generated on IBM System/360 Models 40, 50, 65, 75 85, 91, and 195. (Model 67 and Model 95 channel inboard summaries are identified as Model 65 and 91 summaries, respectively.) Channel inboard records are summarized according to channel address. Each channel summary contains:

- The addresses of devices connected to the channel (a maximum of 10 devices).
- The status of hardware elements (pertaining to the channel) in the logout area.
- A summary of failing CCW command codes (a maximum of 24 entries). (The 24th CCW command code entry is a logical OR of the remainder of the failing command codes, if any.)

IFCEREP0 Figure 5 shows the format of a channel inboard summary. Each summarized item is listed with its frequency of occurrence.

I/O Outboard Summary: An I/O outboard summary can be generated on IBM System/360 Models 30, 40, 50, 65, 67, 75, 85, 91, 95, and 195. I/O outboard summaries are organized according to device address; however, the order of appearance of the summaries is determined by the order in which device addresses are encountered in the OBR records selected for summarization. Each I/O outboard summary contains:

- Volume labels (a maximum of 10 entries).
- A summary of failing CCW command codes (a maximum of 24 entries). (The 24th CCW command code entry is a logical OR of the remainder of the failing command codes, if any.)
- The sense bits (a maximum of 6 bytes).

Note: Selected records can be edited and written, accumulated, and/or summarized in one execution of the program.

IFCEREP0 Figure 6 shows the format of an I/O outboard summary. Each summarized item is listed with its frequency of occurrence.

```
*** MOD xx CHANNEL 1 SUMMARY ***
TOTAL NO. OF RECORDS FOR THE CHANNEL = 20

   TITLE            TOTAL
SUMMARY OF DEVICE ADDRESSES
   (MAX 10 ENTRIES)
   180                5
   190                6
   1F0                5
   UNDET.             4

SUMMARY OF CMND CODES
   (MAX 24 ENTRIES)
   CMND CODES       TOTAL
   '01'               7
   '02'               6
   '12'               3
   '14'               4

SUMMARY OF HARDWARE LOGOUT
   IF PARITY          8
   WLR WR             6
   IF TAG CHK         2
   WO PARITY CHK      4
```

IFCEREP0 Figure 5. Channel Inboard Summary

```
SUMMARY OF I/O OUTBOARD ENVIRONMENT RECORDS FOR DEVICE 031
TOTAL NUMBER OF RECORDS 005                          DEVICE TYPE 2311

VOLUME LABELS ENCOUNTERED (MAXIMUM OF 10 ENTRIES)
VOL. LABEL  22222    001
VOL. LABEL  22223    002
VOL. LABEL  22224    002

CCW COMMAND CODES ENCOUNTERED (MAXIMUM OF 24 ENTRIES)
CMND       TOTAL
02         005

SENSE BYTE SUMMARY

BYTE 0       BYTE 1       BYTE 2       BYTE 3       BYTE 4   BYTE 5

CMND REJ  0  DATA CHK  0  UNSAFE    1  READY     0 BIT 0 0  COMMAND   0
INTV REQ  0  TRK OVERF 0  BIT 1     2  ON LINE   0 BIT 1 1  IN        0
BUS OUT   0  CYL END   1  SERIAL CH 3  UNSAFE    0 BIT 2 2  PROGRESS  0
EQUIP CHK 0  INV SEQ   2  TAG LINE  4  BIT 3     1 BIT 3 3  WHEN      0
DATA CHK  0  REC UNFND 0  ALU CHK   0  ON LINE   0 BIT 4 4  OVERFLOW  0
OVERRUN   0  FILE PROT 3  UNSEL STA 0  CYL END   0 BIT 5 5  INCMPLETE 0
TRK COND  0  MISG A MK 4  BIT 6     0  BIT 6     1 BIT 6 1  OCCURS    0
SEEK CHK  1  OVFL INC  5  BIT 7     0  SEEK INCP 1 BIT 7 1             1
```

IFCEREP0 Figure 6. I/O Outboard Summary

Processing Records Produced on a Different Machine Model

Records from the SYS1.LOGREC data set or from a history data set can be edited and written, accumulated, or summarized by IBM System/360 Models 30, 40, 50, 65, 67, 75, 85, 91, 95, and 195. In addition, a data set recorded on one system can be written by a second system of the same or different model number, provided the system residence volumes of both systems are mounted on the second system, and the serial numbers of the volumes differ.

The job control language requirements are similar to those discussed in the section "Job Control Statements;" however, since the applicable IFCEREP0 program resides on the original system residence volume, the input stream must include:

- A JOBLIB DD statement preceding the EXEC statement to make the original system's link library available to the second system. Within the DD statement, the UNIT parameter specifies the unit on the second system that contains the original system residence volume, and the VOLUME parameter specifies the volume serial number of the original system residence volume.

Note: If the original system's link library resides on a volume other than the system residence volume, that volume must also be mounted on the second system.

Inputs and Outputs

IFCEREP0 Table 1 shows the input to and outputs from the IFCEREP0 program.

IFCEREP0 Table 1. Data Sets Used (Input) and Produced (Output) by the IFCEREP0 Program

Input	Input Data Set: This data set contains the error environment records that are to be edited and written, accumulated, and/or summarized. The data set is organized sequentially. It can be either the SYS1.LOGREC data set or an accumulated (history) data set.
Outputs	Edited Output Data Set: This data set contains: • Edited and written records. • Summarized records. • Informational messages. The sample printouts shown in the section "Editing and Writing Selected Records" are examples of edited data sets that have been written onto a printer. Note: Error messages, if applicable, appear on the console typewriter. Accumulated Output Data Set: This data set contains a history of selected error environment records. It is produced or updated when the accumulation function of the program is specified. The data set is organized sequentially.

Control

The IFCEREP0 program is controlled by job control statements. No utility control statements are required.

JOB CONTROL STATEMENTS

IFCEREP0 Table 2 shows the job control statements necessary for executing the IFCEREP0 program.

IFCEREP0 Table 2. Job Control Statements for the IFCEREP0 Program

Statement	Usage
JOB statement	This statement initiates the job.
EXEC statement	This statement specifies the program name (PGM=IFCEREP0) and PARM parameter information used to control the functions of the program. Refer to "PARM Parameter Control Information" for a discussion of program control.
SERLOG DD statement	This statement defines the input data set as being the SYS1.LOGREC data set. Either this DD statement or the ACCIN DD statement is present for each application of the IFCEREP0 program.
ACCIN DD statement	This statement defines the input data set as being a history (previously accumulated) data set. Either this DD statement or the SERLOG DD statement is present for each application of the program.
EREPPT DD statement	This statement defines the edited output data set. It should be included with each application of the program.
ACCDEV DD statement	This DD statement (optional) defines an accumulated output data set. The accumulated data set can reside on magnetic tape or a direct access volume. Space must be allocated for a new output data set that is to reside on a direct access volume. Space cannot be allocated for an existing output data set.
Notes: The SERLOG, ACCIN, EREPPT, and ACCDEV DD statements define sequential data sets. If records produced on a different machine model are to be processed, a JOBLIB DD statement is required to define the original system's link library. Refer to "IFCEREP0 Examples" for typical uses of the job control statements.	

PARM Parameter Control Information

The PARM parameter within the EXEC statement is used to control the functions of the IFCEREP0 program.

```
PARM=(rectype,model,date,cuu,devtype,N,print,A,R,[D])
                                                 [T]
```

rectype
specifies the type of records to be processed.

Code	Meaning
M	Machine-check records
C	Channel inboard records
O	I/O outboard records
S	Statistical data records
TP	T-type records

A combination of record types can be specified. For example, PARM=(MC,...). If no record type is specified, all record types are processed.

model
indicates that all specified records created on the model or models specified are to be processed. Any combination of model numbers 40, 50, 65, 67, 75, 85, 91, 95, and 195 is valid.

Note: If 67 is specified as a model number, either D or T must also be included in the PARM field.

date
indicates that all of the selected record types generated within a specific period of calendar time are to be processed. The date is written $y_1ddd_1y_2ddd_2$, where y_1 and ddd_1 are the year (of the decade) and day (of the year) when the time period begins, and y_2ddd_2 are the year and day when the period ends. For example, PARM=(M,10011007,...) specifies that machine-check records produced in week 1 of the first year of the decade are to be written.

If no date is specified, all selected records are processed, regardless of when they were generated.

cuu
indicates that I/O outboard or statistical data records, or both (as specified in the rectype subparameter), that are related to a specific channel(s) and unit(s) are to be processed. This subparameter is specified as cuu (one channel and unit address) or cuucuu (two channel and unit addresses).

If cuu is not specified, selected I/O outboard or statistical data records are processed, regardless of device address.

devtype
indicates that I/O outboard or statistical data records, or both (as specified in the rectype subparameter), that are related to a specific device type are to be processed. This subparameter specifies a valid device type; for example, 2311 (disk), 2400 (9-track magnetic tape).

If devtype is not specified, selected I/O outboard or statistical data records are processed, regardless of device type.

N
indicates that input records in the SYS1.LOGREC data set are not to be cleared to hexadecimal zeros after they are processed. If this subparameter is omitted, the selected records are cleared after they are processed. N must not be coded if the accumulate function is specified. N is not applicable if the input is a history data set; i.e., records are not cleared to zeros in the history data set.

Note: It is possible that the user use an operating system on several machines (portability). In this portability environment, the operator must copy SYS1.LOGREC (by running EREP program) onto tape before the system packs are moved to another machine so that the environmental data can later be related to the system that generated it. N should not be coded in this portability environment.

print
indicates how the records are to be processed and written.

Code	Meaning
SU	Suppress full printing (print summary only).
PT	Suppress summary printing (print full record only).
Z	Suppress full record printing and summary printing.
PS	Print full record and summary.

If print is not coded, PS is assumed.

A
indicates that all selected records (except statistical data records) are to be accumulated. If A is coded:

- The N subparameter must not be coded.
- Statistical data records are not accumulated.
- A value of S in the rectype subparameter is invalid unless at least one additional value is included (e.g., M, C, or O), in which case the S is valid, but ignored for accumulation.

R
indicates that the input data set is a history data set, rather than the SYS1.LOGREC data set.

If R is coded, the input data set must be defined with an ACCIN DD statement.

D or T
D specifies that mod 1 Model 67 records are to be processed.

T specifies that mod 2 Model 67 records are to be processed.

Note: In order for the program to recognize Model 67 records existing in the SYS1.LOGREC data set, either D or T must be specified. However, when D or T is specified, any Model 65 machine check (CPU) records are handled as though they are Model 67 records. Furthermore, if accumulation is specified, any Model 65 machine check records encountered are accumulated as Model 67 records.

Neither D nor T is specified when the input is a history data set.

If a value is omitted from the PARM parameter, its absence need not be indicated by a comma; e.g., PARM=(M,,,,,,A,R) can be coded as PARM=(M,A,R). Commas following the last value need not be coded; e.g., PARM=(M,PT).

If the PARM parameter is omitted from the EXEC statement:

- The full contents of all the record types are edited and printed.
- Machine-check, channel inboard, and I/O outboard records are summarized and printed.
- No accumulation is performed.
- The input data set must be the SYS1.LOGREC data set.
- Each input record from the SYS1.LOGREC data set is cleared to zeros when the program is executed.

Note: The header record is initialized in only those executions of the program in which all records existing in the SYS1.LOGREC data set are processed.

If any of the selective retrieval options are selected in the PARM field the selected records are cleared to zero (0); however, the header record is not initialized and the space on SYS1.LOGREC is not relinquished for new records.

The selective retrieval options are:

- Rectype
- Model
- Date
- CUU
- Devtype

IFCEREP0 Examples

The following examples show some of the typical uses of the IFCEREP0 program.

Note: The SYS1.LOGREC data set may or may not be cataloged at an installation.

IFCEREP0 Example 1

Operation	Data Set Organization	Input Device	Output Device	Comments
EDIT & WRITE	Input-SEQUENTIAL Output-SEQUENTIAL	N/A	System output device (printer assumed).	1. All machine-check records are cleared to zeros in the SYS1.LOGREC data set.

In this example machine-check records are to be written on the system output device (printer assumed).

- The EXEC Statement: specifies the record type (machine-check) to be processed and indicates that a full record printout is desired. Summary printing is suppressed.
- The SERLOG DD Statement: defines the input (SYS1.LOGREC) data set. The data set is cataloged.
- The EREPPT DD Statement: defines the system output device (printer assumed).

```
//JOBA     JOB
//         EXEC PGM=IFCEREP0,PARM=(M,PT)
//SERLOG   DD   DSNAME=SYS1.LOGREC,DISP=(OLD,KEEP)
//EREPPT   DD   SYSOUT=A
/*
```

IFCEREP0 Example 1. Writing Machine-Check Records on the Printer (Full Record Printout)

IFCEREP0 Example 2

Operation	Data Set Organization	Input Device	Output Device	Comments
EDIT & WRITE, SUMMARIZE	Input-SEQUENTIAL Output-SEQUENTIAL	N/A	TAPE- 7-track, unlabeled, 200 bits-per-inch density, data converter on	1. Machine-check records are cleared to zero in the SYS1.LOGREC data set.

In this example, date-dependent, machine-check records are to be written onto a 7-track magnetic tape volume at a density of 200 bits per inch.

- The EXEC Statement: specifies (1) the record type (machine-check) to be processed, (2) the applicable time period, and (3) the type of printout (full record and summary print).
- The SERLOG DD Statement: defines the input (SYS1.LOGREC) data set. The data set is cataloged.
- The EREPPT DD Statement: defines the edited output data set. The output records are written on a 7-track, unlabeled, magnetic tape volume at a density of 200 bits per inch (with data converter on). The data set is cataloged for ease of retrieval.

```
//JOBA     JOB
//         EXEC PGM=IFCEREP0,PARM=(M,21102117,PS)
//SERLOG   DD   DSNAME=SYS1.LOGREC,DISP=(OLD,KEEP)
//EREPPT   DD   DSNAME=ERRDATA,UNIT=2400-2,LABEL=(,NL),
//            DCB=(DEN=0,TRTCH=C),DISP=(NEW,CATLG)
/*
```

IFCEREP0 Example 2. Writing Date-Dependent Machine-Check Records on 7-Track Tape

IFCEREP0 Example 3

Operation	Data Set Organization	Input Device	Output Device	Comments
EDIT & WRITE	Input-SEQUENTIAL Output-SEQUENTIAL	N/A	TAPE- 9-track, unlabeled, 800 bits-per-inch density	1. Statistical data records are cleared to zeros in the SYS1.LOGREC data set.

In this example, statistical data records generated by a specific channel and unit are to be written onto 9-track magnetic tape.

- The EXEC Statement: specifies (1) the record type (statistical data) to be processed, (2) the applicable channel and unit address, and (3) the type of printout (full record print).
- The SERLOG DD Statement: defines the input (SYS1.LOGREC) data set. The data set is cataloged.
- The EREPPT DD Statement: defines the edited output data set. The output records are written on a 9-track magnetic tape volume at a density of 800 bits per inch. The data set is cataloged for ease of retrieval.

```
//JOBA     JOB
//         EXEC PGM=IFCEREP0,PARM=(S,280,PT)
//SERLOG   DD   DSNAME=SYS1.LOGREC,DISP=(OLD,KEEP)
//EREPPT   DD   DSNAME=ERRDATA,UNIT=2400,LABEL=(,NL),DISP=(NEW,CATLG)
/*
```

IFCEREP0 Example 3. Writing Statistical Data Records Generated by a Specific Channel and Unit Onto 9-Track Tape

IFCEREP0 Example 4

Operation	Data Set Organization	Input Device	Output Device	Comments
EDIT & WRITE, SUMMARIZE, ACCUMULATE	Input-SEQUENTIAL Outputs-SEQUENTIAL	N/A	DISK - 2311	1. The selected records are cleared to zero in the SYS1.LOGREC data set.

In this example, machine-check and channel inboard records are to be moved to an accumulated output data set. In addition the records are summarized and printed in a full record format.

- The EXEC Statement: specifies (1) the record types (machine-check and channel inboard) to be processed, (2) the type of printout (full record and summary), and (3) accumulation.
- The SERLOG DD Statement: defines the input (SYS1.LOGREC) data set. The data set is cataloged.
- The EREPPT DD Statement: defines the edited output data set. The output records are written on the system output device (printer assumed).
- The ACCDEV DD Statement: defines the accumulated output data set. The data set is cataloged for ease of retrieval.

```
//JOBA     JOB
//         EXEC PGM=IFCEREP0,PARM=(MC,PS,A)
//SERLOG   DD   DSNAME=SYS1.LOGREC,DISP=(OLD,KEEP)
//EREPPT   DD   SYSOUT=A
//ACCDEV   DD   DSNAME=ACCUMSET,UNIT=2311,DISP=(NEW,CATLG),
//             VOLUME=SER=111112,SPACE=(TRK,(40,10))
/*
```

IFCEREP0 Example 4. Writing, Summarizing, and Accumulating Machine-Check and Channel Inboard Records

IFCEREP0 Example 5

Operation	Data Set Organization	Input Device	Output Device	Comments
EDIT & WRITE, ACCUMULATE	Input-SEQUENTIAL Outputs-SEQUENTIAL	N/A	1. PRINTER - (for the edited data set) 2. N/A (accumulated data set)	1. The input and output history data sets are cataloged. 2. The accumulated output data set was created prior to this job step.

In this example, machine-check and I/O outboard records contained in a history data set are to be written onto a system output device (printer assumed). In addition, the records are to be moved from the history data set to a second history data set.

- The EXEC Statement: specifies (1) the record types (machine-check and I/O outboard) to be processed, (2) the type of printout (full record), and (3) accumulation. In addition, this statement identifies the input data set as being a history data set.
- The ACCIN DD Statement: defines the input history data set. The data set is cataloged.
- The EREPPT DD Statement: defines the edited output data set. The output records are written on the system output device (printer assumed).
- The ACCDEV DD Statement: defines the accumulated output data set. The data set was cataloged when it was created.

```
//JOBA     JOB
//        EXEC PGM=IFCEREP0,PARM=(MO,PT,A,R)
//ACCIN    DD   DSNAME=HISTRYIN,DISP=(OLD,CATLG)
//EREPPT   DD   SYSOUT=A
//ACCDEV   DD   DSNAME=EXISTACC,DISP=(MOD,CATLG)
/*
```

IFCEREP0 Example 5. Accumulating and Writing Machine-Check and I/O Outboard Records Contained in a History Data Set

IFCEREP0 Example 6

Operation	Data Set Organization	Input Device	Output Device	Comments
EDIT & WRITE, ACCUMULATE	Input-SEQUENTIAL Outputs-SEQUENTIAL	N/A	1. PRINTER (edited data set). 2. N/A (accumulated data set).	1. The accumulated data set is used as input in the second job step.
EDIT & WRITE	Input-SEQUENTIAL Output-SEQUENTIAL	N/A	PRINTER	

This example is a two-step job. Together the job steps are to produce a printout of machine-check records from the SYS1.LOGREC data set and machine-check records from a history data set.

The first job step (STEPA):

- Edits and writes machine-check records contained in the SYS1.LOGREC data set.
- Accumulates machine-check records in an existing accumulated data set. (This data set is used as input in the second job step.)

The Second job step (STEPB):

- Uses the accumulated data set from step 1 as input.
- Edits and writes the machine-check records contained in the input data set.

STEPA:

- The EXEC Statement: specifies (1) the record type (machine-check) to be processed, (2) the type of printout (full record), and (3) accumulation.
- The SERLOG DD Statement: defines the input (SYS1.LOGREC) data set. The data set is cataloged.
- The EREPPT DD Statement: defines the edited output data set. The output records are written on the system output device (printer assumed).
- The ACCDEV DD Statement: defines the accumulated output data set. The data set was cataloged when it was created.

STEPB:

- The EXEC Statement: specifies the record type (machine-check) to be processed, and the type of printout desired. In addition, it identifies the input data set as being a history data set.
- The ACCIN DD Statement: defines the history (input) data set. The data set is cataloged.
- The EREPPT DD Statement: defines the edited output data set. The output records are written on the system output device (printer assumed).

```
//JOBA     JOB
//STEPA    EXEC PGM=IFCEREP0,PARM=(M,PT,A)
//SERLOG   DD   DSNAME=SYS1.LOGREC,DISP=(OLD,CATLG)
//EREPPT   DD   SYSOUT=A
//ACCDEV   DD   DSNAME=HISTORY,DISP=(MOD,CATLG)
/*
//STEPB    EXEC PGM=IFCEREPO,PARM=(M,PT,R)
//ACCIN    DD   DSNAME=HISTORY,DISP=(OLD,CATLG)
//EREPPT   DD   SYSOUT=A
/*
```

IFCEREP0 Example 6. Writing Recently Generated Machine-Check Records and Accumulated Machine-Check Records (for Comparison)

IFCEREP0 Example 7

Operation	Data Set Organization	Input Device	Output Device	Comments
EDIT & WRITE, SUMMARIZE	Input-SEQUENTIAL Output-SEQUENTIAL	N/A	PRINTER	1. Statistical data records are not cleared to zeros in the SYS1.LOGREC data set.

In this example, statistical data records generated by another IBM System/360 Computing System are to be written onto the printer.

- The JOBLIB DD Statement: defines the original system's link library. This statement ensures that the IFECREPO program that belongs to the original system is executed.
- The EXEC Statement: specifies the record type (statistical data) to be processed, and the type of printout (full record) desired. In addition, it specifies that statistical data records are not to be cleared to zeros after they are processed.
- The SERLOG DD Statement: defines the input (SYS1.LOGREC) data set.
- The EREPPT DD Statement: defines the edited output data set. The output records are written on the system output device (printer assumed).

```
//JOBLIB   DD   DSNAME=SYS1.LINKLIB,UNIT=191,DISP=(OLD,KEEP),
//              VOLUME=SER=222222
//         EXEC PGM=IFCEREP0,PARM=(S,N,PT)
//SERLOG   DD   DSNAME=SYS1.LOGREC,UNIT=191,DISP=(OLD,KEEP),
//              VOLUME=SER=222222
//EREPPT   DD   SYSOUT=A
/*
```

IFCEREP0 Example 7. Writing Machine-Check Records Generated by Another IBM System/360 Computing System Onto the Printer

Program Application

The SYS1.LOGREC data set is automatically initialized when the system is generated. The IFCDIP00 system utility program is used to reinitialize the SYS1.LOGREC data set in the event it is destroyed (indicated by an IFB004I or IFB001I message to the operator).

Program Output

The program produces a reinitialized SYS1.LOGREC data set as output.

Program Control

The program is executed and controlled by job control statements. (No utility control statements are required.)

JOB CONTROL STATEMENTS

IFCDIP00 Table 1 shows the job control statements necessary for executing the IFCDIP00 sytem utility program.

IFCDIP00 Table 1. Job Control Statements for the IFCDIP00 Program

Statement	Usage
JOB statement	This statement initiates the job.
EXEC statement	This statement specifies the program name (PGM=IFCDIP00) and PARM=nnnxx where nnn is the number of uniquely addressable devices plus an additional count for each 2314 in the system, and xx is a hexadecimal code number for the system residence device type. Code — Device Type 01 — IBM 2311 Disk Storage Drive 02 — IBM 2301 Drum Storage Drive 03 — IBM 2303 Drum Storage Drive 08 — IBM 2314 Direct Access Storage Facility
SERERDS DD statement	This statement defines the output (SYS1.LOGREC) data set. The statement should include the DSNAME, DISP, UNIT, and VOLUME parameters.
Note: The hexadecimal code (xx) included in the PARM parameter of the EXEC statement must indicate the type of device originally chosen as the system residence device type.	

IFCDIP00 Example

IFCDIP00 Example 1 shows how the SYS1.LOGREC data set is reinitialized in a system containing 21 uniquely addressable devices. An IBM 2311 Disk Storage Drive was originally chosen as the system residence device.

IFCDIP00 Example 1

Operation	Data Set Organization	System Residence Device
INITIALIZE the SYS1.LOGREC data set	SEQUENTIAL	DISK - 2311

- The EXEC Statement: specifies the program name, the number of uniquely addressable devices, and the system residence device type.
- The SERERDS DD Statement: defines the output (SYS1.LOGREC) data set.

```
//INSERLOG JOB
//         EXEC PGM=IFCDIP00,PARM=02101
//SERERDS  DD   DSNAME=SYS1.LOGREC,UNIT=2311,DISP=(OLD,KEEP),
//             VOLUME=SER=111111
/*
```

IFCDIP00 Example 1. Reinitializing the SYS1.LOGREC Data Set

Program Applications

The IEHDASDR program prepares direct access volumes for IBM System/360 Operating System use and ensures that any permanent hardware errors (i.e., defective tracks) incurred on direct access volumes do not seriously degrade the performance of those volumes. In addition, the IEHDASDR program can dump the entire contents or portions of a direct access volume onto a volume or volumes of the same direct access device type, onto a magnetic tape volume or volumes, or onto a system output device. Data that is dumped onto a magnetic tape volume is arranged so that it can subsequently be "restored" to its original organization by the IEHDASDR program. The direct access device types supported by the IEHDASDR program are: 2301, 2302, 2303, 2311, 2314, and 2321.

The program can be used to:

- ANALYZE -- analyze tracks, assign alternate tracks for defective tracks, and perform housekeeping and formatting functions to make a direct access volume suitable for IBM System/360 Operating System use.
- FORMAT -- perform housekeeping and formatting functions without analyzing tracks.
- LABEL -- change the volume serial number of a formatted direct access volume.
- GETALT -- assign alternate tracks for specified defective or questionable tracks on disk or data cell volumes.
- 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 magnetic tape volume onto a direct access volume.

If during an ANALYZE, FORMAT, DUMP, or RESTORE operation, a password protected data set is encountered, a message will be issued to the operator, requesting him to reply with the proper password.

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; 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.

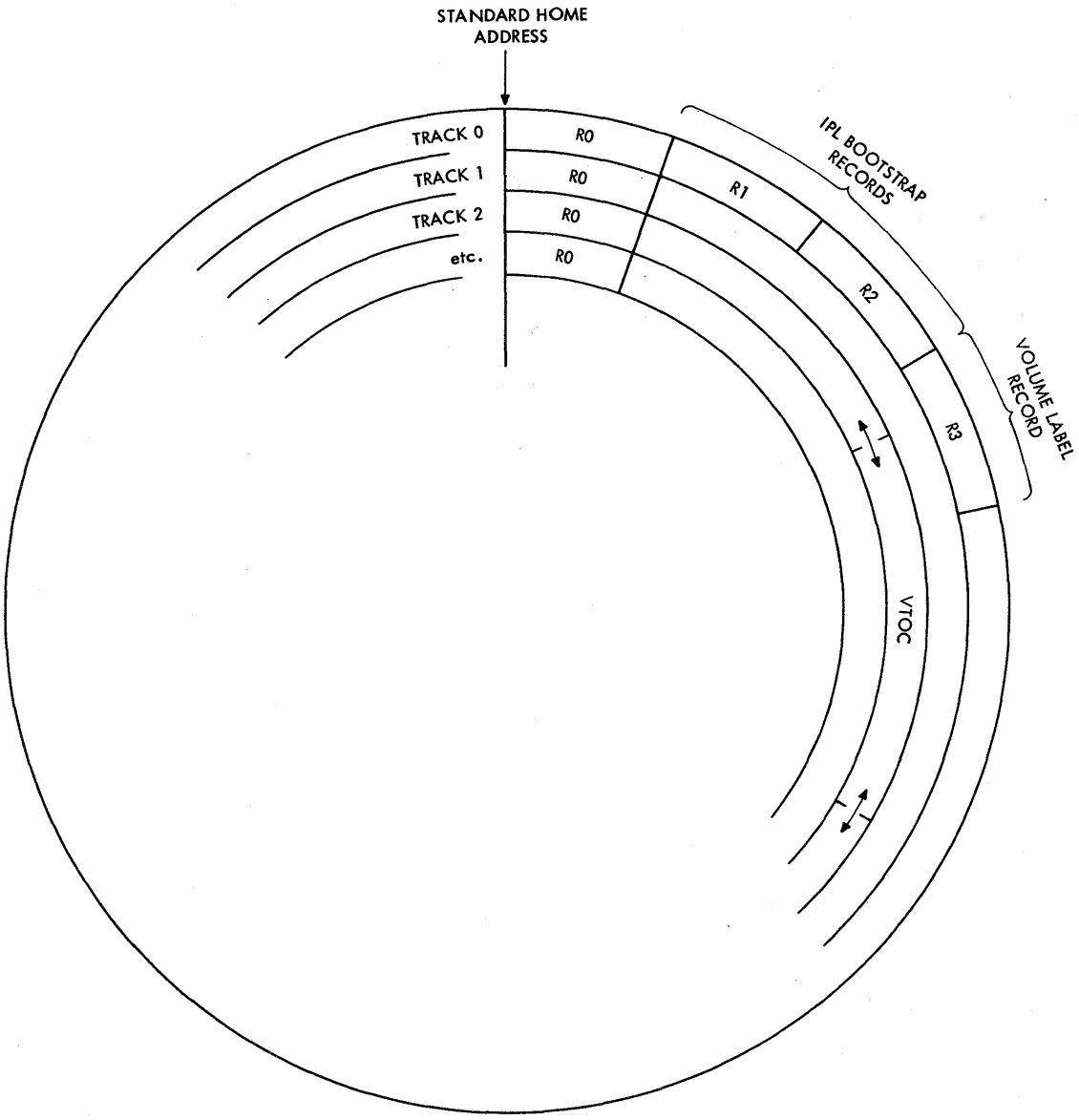
Analyzing the Recording Surface of a Direct Access Volume (ANALYZE)

The ANALYZE function:

- Assigns alternate tracks for any disk or data cell tracks found defective during an analysis, or for any track previously flagged defective. Each track can be analyzed from 1 to 255 times, at the discretion of the user. The test of looking for previously flagged tracks must be suppressed when the ANALYZE function is specified for a new or nonformatted direct access volume.

- "Standardizes" each track by placing a standard home address and a record zero (R0) field on it. The remainder of the track is erased.
- Constructs 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 writes an IPL program record (2301, 2303, 2311, and 2314 volumes only) and provides "owner" information in the volume label record.

IEHDASDR Figure 1 shows a direct access volume after it has been prepared for IBM System/360 Operating System use. A direct access volume can be "initialized" in this manner by either the ANALYZE function or the FORMAT function.



IEHDASDR Figure 1. An Initialized Direct Access Volume

Preparing a Direct Access Volume for System Use (FORMAT)

The FORMAT function prepares a direct access volume for use by the system. This function differs from the ANALYZE function in that it does not perform an analysis or assign alternate tracks. For this reason, the FORMAT function should not be used for the first initialization of a volume.

The FORMAT function:

- Checks a direct access volume for previously flagged tracks (except for drum volumes). No formatting is performed on known defective tracks.
- Standardizes each track by placing a standard home address and a record zero (R0) field on it. The remainder of the track is erased.
- Constructs 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 writes an IPL program record (2301, 2303, 2311, and 2314 devices only) and provides owner information in the volume label record.

Changing the Volume Serial Number of a Direct Access Volume (LABEL)

The LABEL function changes the volume serial number of a direct access volume. Optionally, this function places a 1- to 10-character owner name in the volume label record (record 3 of track 0). If an owner name already exists, it is overwritten with the new name.

Note: The LABEL function can only be used on an initialized volume.

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 (GETALT)

The GETALT function assigns an alternate track for a specified track on a data cell or 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. The GETALT function can be performed only if the data cell or disk volume has been previously initialized.

Notes: If it becomes necessary to assign an alternate track on a drum volume, an IBM customer engineer should be notified.

A list of defective tracks is provided with new IBM Disk Storage Devices. The GETALT function can be used to assign alternate tracks for those tracks mentioned on the list.

Creating a Backup, Transportable, or Printed Copy (DUMP)

The IEHDASDR program can dump a direct access volume or a portion of a volume onto any number of magnetic tape volumes or volumes of the same direct access device type, or onto 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 primary track is found to have an alternate track assigned, the alternate is dumped in place of the primary.

Each track to be dumped will have all of its data except the home address and the count field of record zero (R0) copied from it onto the receiving volume.

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.

Note: Except for a printing operation, only data that is "owned" is dumped; that is, the IEHDASDR program checks the first or only Format 5 data set control block (DSCB) in the volume table of contents. The 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 onto 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.

IEHDASDR Figure 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.

Note: If an alternate track is printed in place of a primary track, it is identified in the printout by the primary track address.

```
*** TRACK  ccchhhh      RO DATA  xxxxxxxxxxxxxxxx

    COUNT  ccchhhhrrkkdddd
                          key and data fields (hexadecimal)                    key and data fields (EBCDIC)

000000   xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx    xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx  *................................*
000032   xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx    xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx  *................................*
                             etc.

    COUNT  ccchhhhrrkkdddd

000000   xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx    xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx  *................................*
000032   xxxxxxxx xxxxxxxx xxxxxxxx etc.


*** TRACK  ccchhhh      RO DATA  xxxxxxxxxxxxxxxx

    COUNT  ccchhhhrrkkdddd

000000   xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx    xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx  *................................*
000032   xxxxxxxx xxxxxxxx etc.
```

IEHDASDR Figure 2. Format of Printed Output When Dumping the Contents of a Direct Access Volume Onto a Printer

Copying Dumped Data From a Magnetic Tape Volume to a Direct Access Volume (RESTORE)

When a direct access volume is dumped onto a magnetic tape volume, the data is placed in a format that is specially suited for the magnetic tape volume. The IEHDASDR program 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 2311 volume can be "restored" to a 2311 volume; data 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 the IBCDMPRS (DUMP/RESTORE) independent utility program 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 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.

Inputs and Outputs

The input to the IEHDASDR program is a control data set containing utility control statements and optionally, IPL text. The utility control statements are used to control the functions of the program and to refer to DD statements defining volumes to be processed by the program.

The primary output or result of executing the program is determined by the function(s) that the user specifies.

A sequential message data set is created to list informational messages (e.g., control statements used), dumped data (for a print operation), and error messages, if any.

ADDITIONAL OUTPUTS

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

00 -- successful completion.

04 -- an unusual condition was encountered; however, the overall result is successful. A warning message is issued.

08 -- a specified operation did not complete successfully. An attempt is made to perform additional specified operations, if any.

16 -- either an error occurred upon invoking the IEHDASDR program or the program was unable to open the input or message data set. The job step is terminated.

Control

The IEHDASDR program is controlled by job control statements and utility control statements. The job control statements are used to:

- Execute or invoke the program.
- Define the control and message data sets.
- Define volumes and/or devices to be used during the course of program execution.

The utility control statements are used to control the functions of the program.

JOB CONTROL STATEMENTS

IEHDASDR Table 1 shows the job control statements necessary for executing or invoking the IEHDASDR program.

IEHDASDR Table 1. Job Control Statements for the IEHDASDR Program (Part 1 of 3)

Statement	Usage
JOB statement	This statement initiates the job.
EXEC statement	This statement specifies the program name (PGM=IEHDASDR) or, if the job control statements reside in a procedure library, the procedure name. This statement can include optional PARM information. For a discussion of the PARM parameter, refer to "PARM information in the EXEC statement."
SYSPRINT DD statement	This statement 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 device.
//anyname* DD UNIT=xxxx,VOLUME=SER=xxxxxx,DISP=OLD This statement defines a direct access device type. One statement must be included for each device to be used in the job step. If more than one volume is to be processed on a single mountable device, deferred mounting can be specified by including: UNIT=(xxxx,,DEFER),VOLUME=(PRIVATE,...),DISP=(NEW,KEEP) in a DD statement defining a mountable device. Notes: This DD statement is not used for an operation that analyzes an offline direct access volume. If an IEHDASDR operation changes a volume serial number and a subsequent operation is performed on the newly labeled volume (in the same job step), two of these DD statements are required. The VOLUME parameter in the first DD statement includes the old volume serial number; the VOLUME parameter in the second statement specifies the new volume serial number. In addition, the second DD statement specifies unit affinity with the first.	

(Part 1 of 3)

IEHDASDR Table 1. Job Control Statements for the IEHDASDR Program (Part 2 of 3)

Statement	Usage
	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: UNIT=(191,,DEFER),VOLUME=PRIVATE,DISP=(NEW,KEEP) In this case, the operator will be asked to mount a scratch (SCRTCH) volume on that unit. It may also be possible to make a nonspecific unit request as shown in Appendix D. The system will inform the operator on which unit to mount the scratch volume. If an IEHDASDR operation produces a serial number(s) that is a duplicate of a serial number already allocated within the system, the IEHDASDR program makes unavailable to the system the volume(s) to which it has assigned the serial number. The operator is asked to remove the applicable volume(s) at the completion of the function.

	//tapename* DD UNIT=xxxx,VOLUME=SER=xxxxxx,LABEL=(...,...), C
	// DISP=(...,KEEP),DCB=(TRTCH=C,DEN=x)
	
	7-track only
	
	This statement defines a magnetic tape drive. A version of this statement must be included for a DUMP operation onto magnetic tape or for a RESTORE operation from magnetic tape.
	
	If more than one magnetic tape volume is to be processed on the same tape drive, deferred mounting can be specified by including:
	
	UNIT=(xxxx,,DEFER),VOLUME=(PRIVATE,...)
	
	in the DD statement defining the magnetic tape drive.

SYSIN DD statement	This statement 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 alternatively as a blocked or unblocked sequential data set or as a member of a procedure library.
	*The "anyname" 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 other than 1. 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 if a block size other than a multiple of 121 is included in the statement, a block size of 121 bytes is assumed. SYSIN data sets can be concatenated, provided that SYSIN attributes are identical. 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.

(Part 2 of 3)

IEHDASDR Table 1. Job Control Statements for the IEHDASDR Program (Part 3 of 3)

Because the IEHDASDR program 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 the IEHDASDR program is run in a multiprogramming environment, the user must choose a combination of DD statements (defining mountable devices) that will ensure that volume integrity is maintained. Refer to Appendix D for information on defining mountable volumes.

The IEHDASDR program always checks the volume table of contents of each volume to be processed for the presence of password protected data sets.

If password protected data sets reside on volumes which are used by the IEHDASDR program, the following considerations must be made:

- When dumping from a volume containing 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 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, the direct access "TO" volume is checked for password protected data sets. At this time the operator must supply the correct password (in answer to a console message) for each password protected data set encountered.

Refer to the publication IBM System/360 Operating System: System Programmer's Guide, GC28-6550, for additional information on password protection facilities.

Concurrent Operations: The IEHDASDR program can perform up to six concurrent operations of the same type (ANALYZE, FORMAT, DUMP, or RESTORE operations only). This feature, which can shorten the amount of 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 DUMP statements permits six concurrent dump operations to take place; a combination of four successive RESTORE statements permits four concurrent restore operations to take place, etc. However, if the utility control statements are arranged so that no operations of the same type appear in succession, e.g.,

```
DUMP
RESTORE
DUMP
RESTORE
  .
  .
  .
```

no operations are performed concurrently, even though many devices might be defined for use.

Note: The concurrent operation feature can be overridden by an EXEC statement PARM value that permits explicit definition of the number of successive like commands to be executed concurrently.

PARM Parameter Information in the EXEC Statement

The EXEC statement for the IEHDASDR program 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='LINECNT=xx'          ]
                            [,PARM='N=n'                 ]
                            [,PARM='LINECNT=xx,N=n'      ]
```

Where:

LINECNT=xx
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.

N=n (applicable to the ANALYZE, FORMAT, DUMP, and RESTORE functions) specifies a decimal number from one to six. The number represents the maximum number of like functions that can be performed concurrently by the IEHDASDR program.

If N is omitted, up to six like functions can be performed concurrently.

IEHDASDR Table 2 shows the effect of the N keyword over concurrent operations. This table assumes that adequate system resources are available for each case.

IEHDASDR Table 2. Effect of the N Keyword Over Concurrent Operations

N not specified	Up to six operations are performed concurrently -- according to the number of successive like commands in the input stream.
N=1 N=2 N=3 N=4 N=5 N=6	The specified number of operations (N=n) is performed concurrently, provided that n number of successive like commands are included in the input stream. No more than n concurrent operations can be performed at any one time in the job step; 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.

Note: System resources permitting, multiple output copies can be specified in any one or all of the commands to be executed concurrently.

Combinations of ANALYZE, FORMAT, LABEL, GETALT, DUMP, and RESTORE utility control statements are used to control the functions of the program. In addition, an IPLTXT utility control statement is used with the ANALYZE or FORMAT control statement when IPL text is included in the input stream.

The ANALYZE Statement

The ANALYZE statement causes bit patterns to be written on a track. They are then read and tested for defects. If no defects are found, the track is formatted to make it ready for system use.

Name	Operation	Operand
[name]	ANALYZE	{TODD=(cuu,...) / TODD=(ddname,...)}
		VTOC=xxxxx
		EXTENT=xxxxx
		[NEWVOLID=serial]
		[IPLDD=ddname]
		[FLAGTEST={YES / NO}]
		[PASSES=n]
		[OWNERID=name]
		[PURGE={YES / NO}]

TODD=(cuu,...) (used only for the analysis of a volume offline, which includes the first analysis of a volume)
specifies the channel and unit address of a direct access device containing a volume to be initialized. If the volume to be processed is a 2321 volume, TODD=cuu/b is specified, where cuu is the channel and unit address of the device, and b is the bin number of the volume. No DD statement defining a mountable device is required if TODD=(cuu,...) or TODD=cuu/b is specified.

When TODD=(cuu,...) or TODD=cuu/b, is coded, the specified devices must be varied offline (by use of the VARY OFFLINE command) prior to the execution of the job step.

TODD=(ddname,...)
specifies the ddname of a job control statement defining a direct access device containing a volume to be analyzed, formatted, and labeled. Multiple ddnames specifying additional job control statements can be included unless the TODD device is a 2321, in which case only one DD statement can be referred to.

Note: If multiple volumes are specified in an ANALYZE statement and an abnormal completion of the ANALYZE operation occurs, the operation is terminated on all volumes.

VTOC=xxxxx
specifies a 1- to 5-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.
- track 1 if IPL text is written (2303 and 2311 volumes only).

EXTENT=xxxxx
specifies the decimal length of the VTOC in tracks. The VTOC cannot extend into the alternate track area or onto a second volume.

NEWVOLID=serial (mandatory for the analysis of a volume offline)
specifies a 1- to 6-character serial number. The serial number is assigned to all direct access volumes processed through the use of this control statement.

If NEWVOLID is omitted, all direct access volumes retain their own serial numbers.

IPLDD=ddname (applicable to 2301, 2303, 2311, and 2314 volumes)
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 utility control statement is used to separate the ANALYZE statement from the IPL program text statements. The format of the IPLTXT utiltiy control statement is:

Name	Operation
[name]	IPLTXT

IPL text need be included only once in the input stream; that is, the IEHDASDR program refers to the first copy of IPL text encountered when performing multiple functions in a single job step. The IPL text must follow the first statement referring to it.

FLAGTEST=NO (applicable to disk and data cell volumes; not applicable to drum volumes)
specifies that the program is not to check for previously flagged track on the volume. When FLAGTEST=YES is specified (or when the FLAGTEST keyword is omitted) the program checks each data cell or disk track prior to writing a bit pattern to see if the track was flagged "defective" previously. (However, when a volume is initialized offline, the program ignores any FLAGTEST keyword which may have been specified. Offline operation of the program does not check for flags.)

PASSES=n(applicable to disk and drum volumes)
specifies that the bit pattern test is to be performed n times, where n is a decimal number from 1 to 255. If PASSES=n is omitted, the bit pattern test will be performed once on each track.

PASSES=0(applicable to all direct access volumes supported by IEHDASDR)
specifies that the "QUICK DASDI" feature of the ANALYZE function is to be performed. The "QUICK DASDI" feature bypasses all surface analysis and track formatting, writing only a VTOC, track zero records (IPL bootstrap and volume label records), and IPL text if requested.

All previously initialized direct access devices supported by IEHDASDR may be "QUICK DASDI'ed" while online. The offline "QUICK DASDI" feature is only supported for factory initialized 2314 volumes. An offline "QUICK DASDI" request will always check for a volume label, and if one is found, the function will be terminated.

The "QUICK DASDI" feature of the ANALYZE function should not be used for initialization of new volumes other than the factory initialized 2314 volumes.

OWNERID=name
specifies a 1- to 10-character name or other identifying information to be placed in the volume label record. The field should be specified as an EBCDIC character string with the exclusion of the blank and the comma characters (these terminate the control card scan of a field or an entire card).

PURGE=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. The operator replies are:

Reply	Meaning
U	All unexpired data sets on this volume can be overwritten. (The ANALYZE operation continues.)
T	This volume contains unexpired data sets that must not be overwritten. (The ANALYZE operation is terminated.)

If PURGE=NO is coded (or the PURGE keyword is omitted) and an unexpired data set is encountered, the ANALYZE operation is terminated.

Note: The PURGE keyword has no control over 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 ANALYZE operation is terminated.

The FORMAT Statement

The FORMAT statement prepares a volume for IBM System/360 Operating System use. Except for flag testing, no analysis is made prior to formatting a track. Previously flagged disk tracks remain flagged and will have alternate tracks assigned, where applicable.

Note: The FORMAT function is not applicable to the 2321 data cell. The ANALYZE function should be used when initializing a 2321 volume. (If FORMAT is specified for a 2321 volume, the ANALYZE function is automatically performed.)

Name	Operation	Operand
[name]	FORMAT	TODD=(ddname,...) VTOC=xxxxx EXTENT=xxxxx [NEWVOLID=serial] [IPLDD=ddname] [OWNERID=name] [PURGE={YES / NO}]

TODD=(ddname,...)
specifies the ddname of a job control statement defining a direct access device containing a volume to be formatted. Multiple ddnames specifying additional job control statements can be included.

Note: If multiple volumes are specified in a FORMAT statement and an abnormal completion of the FORMAT operation occurs, the operation is terminated on all volumes.

VTOC=xxxxx
specifies a 1- to 5-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.
- track 1 if IPL text is written (2303 and 2311 volumes only).

EXTENT=xxxxx
specifies the decimal length of the VTOC in tracks. The VTOC cannot extend into the alternate track area or onto a second volume.

NEWVOLID=serial
specifies a 1- to 6-character serial number. The serial number is assigned to all direct access volumes processed through the use of this control statement.

If NEWVOLID is omitted, the direct access volumes retain their own serial numbers.

IPLDD=ddname (Applicable to 2301, 2303, 2311, or 2314 volumes)
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 utility control statement is used to separate the FORMAT statement from the program text statements. The format of the IPLTXT control statement is:

Name	Operation
[name]	IPLTXT

Note: IPL text need be included only once in the input stream; that is, the IEHDASDR program refers to the first copy of IPL text encountered when performing multiple functions in a single job step. The IPL text must follow the first statement referring to it.

OWNERID=name
specifies a 1- to 10-character name or other identifying information to be placed in the volume label record. The field should be specified as an EBCDIC character string with the exclusion of the blank and the comma characters (these terminate the control card scan of a field or an entire card).

PURGE=YES
indicates that all unexpired data sets on the volume can be over-written provided that the operator signals his concurrence when the first unexpired data set is encountered. The operator replies are:

Reply	Meaning
U	All unexpired data sets on this volume can be overwritten. (The FORMAT operation continues.)
T	This volume contains unexpired data sets that must not be overwritten. (The FORMAT operation is terminated.)

If PURGE=NO is coded (or the PURGE keyword is omitted) and an unexpired data set is encountered, the FORMAT operation is terminated.

Note: The PURGE keyword has no control over 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 FORMAT operation is terminated.

The LABEL Statement

The LABEL statement changes the serial number of a direct access volume and optionally updates the owner field in record 3 of track 0.

Name	Operation	Operand
[name]	LABEL	TODD={cuu / ddname}
		NEWVOLID=serial
		[OWNERID=name]

TODD=cuu (used only for labeling an offline volume)
specifies the channel and unit address of a direct access device containing a volume whose serial number is to be changed. If the volume to be processed is a 2321 volume, TODD=cuu/b is specified, where cuu is the channel and unit address of the device and b is the bin number of the volume. No DD statement defining a mountable device is required if TODD=cuu or TODD=cuu/b is specified.

One LABEL statement must be included for each volume that is to have its serial number changed.

When TODD=cuu or TODD=cuu/b is coded, the specified device must be varied offline (by use of the VARY OFFLINE command) prior to the execution of the job step.

TODD=ddname
specifies the ddname of a job control statement defining a direct access device containing a volume whose serial number is to be changed. One LABEL statement must be included for each volume that is to have its serial number changed.

NEWVOLID=serial
specifies a 1- to 6-character serial number. The serial number is assigned to the direct access volume processed through the use of this control statement.

OWNERID=name
specifies a 1- to 10-character name or other identifying information.

If OWNERID is omitted, the old owner information, if any, is retained. The field should be specified as an EBCDIC character string with the exclusion of the blank and the comma characters (these terminate the control card scan of a field or an entire card).

The GETALT Statement

The GETALT statement assigns an alternate track for a specified data cell or disk track.

Name	Operation	Operand
[name]	GETALT	TODD=ddname TRACK=ccccHHHH

TODD=ddname
specifies the ddname of a job control statement defining a data cell or disk device containing a volume on which an alternate track is to be assigned.

TRACK=ccccHHHH
specifies in hexadecimal the cylinder number (cccc) and head number (hhhh) of a track for which an alternate track is requested. (When referring to a 2321 volume, cccc is the subcell and strip address, and hhhh is the cylinder and head address.

TRACK=ccccHHHH cannot specify track 0 or the first track occupied by the VTOC.

The DUMP Statement

The DUMP statement dumps a single track, a group of tracks, or an entire direct access volume onto one or more direct access volumes of the same device type, onto one or more magnetic tape volumes, or onto a system output device (printer assumed).

Name	Operation	Operand
[name]	DUMP	FROMDD=ddname TODD=(ddname,...) [CPYVOLID={YES / NO}] [BEGIN=ccccHHHH] [END=ccccHHHH] [PURGE={YES / NO}]

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.

TODD=(ddname,...)
specifies the ddname of the system output device (SYSPRINT) or specifies the ddnames of the DD statements defining the devices containing the direct access or magnetic tape volumes on which copies are to be made.

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 will be identified and printed.

An extra I/O 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.
- I/O error on a search command.
- Missing address marker and no address 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 an abnormal completion of the DUMP operation occurs, the operation is terminated on all output volumes.

CPYVOLID=YES
specifies that all receiving direct access volumes are to be assigned the serial number of the dumped volume.

If CPYVOLID=NO is coded or the CPYVOLID keyword is omitted, all receiving volumes retain their own serial numbers.

BEGIN=ccccchhhh
specifies in hexadecimal a cylinder number (cccc) and head number (hhhh) that identify the first track to be dumped. (When referring to a 2321 volume, cccc is the subcell and strip address, and hhhh is the cylinder and head address.)

If BEGIN is omitted, the DUMP operation begins with track 0.

END=ccccchhhh
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 the BEGIN and the END keywords specify that track address. (When referring to a 2321 volume, cccc is the subcell and strip address, and hhhh is the cylinder and head address.)

If END is omitted, 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.)

PURGE=YES
indicates that all unexpired data sets on a receiving direct access volume can be overwitten, provided that the operator signals his concurrence when the first unexpired data set is encountered. The operator replies are:

Reply	Meaning
U	All unexpired data sets on the receiving direct access volume can be overwritten. (The DUMP operation continues.)
T	The receiving direct access volume contains unexpired data sets that must not be overwitten. (The DUMP operation is terminated.)

If PURGE=NO is coded (or the PURGE keyword is omitted) and an unexpired data set is encountered on a receiving direct access volume, the DUMP operation is terminated. The PURGE keyword is not applicable when dumping onto a restore tape.

The PURGE keyword has no control over 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 DUMP operation is terminated.

CAUTIONS: The user is cautioned against performing such operations as dumping a volume and restoring new data to that volume in the same job step. The IEHDASDR program 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. Since only those tracks that are dumped are placed on the receiving volume, the partially dumped data may not be usable under the operating system. When partially dumped data is subsequently restored, it is placed on the same tracks as it originally occupied.

When space permits, more than one direct access volume can be dumped onto a restore tape. However, the IEHDASDR program creates two files for each volume of data that is dumped. Therefore, the LABEL keyword sequence number in the DD statement defining the restore volume must be coded as follows:

```
LABEL=(3,...)   for the second volume dumped onto the restore tape.
LABEL=(5,...)   for the third volume dumped onto the restore tape.
LABEL=(7,...)   for the fourth volume dumped onto the restore tape.
etc.
```

or, in the case of an IPL restore tape, as follows:

```
LABEL=(2,...)   for the first volume dumped onto the restore tape.
LABEL=(4,...)   for the second volume dumped onto the restore tape.
LABEL=(6,...)   for the third volume dumped onto the restore tape.
etc.
```

The files are referred to in the same manner when restoring data onto a direct access device.

When processing an unlabeled tape prior to a DUMP operation, the IEHDASDR program will first write an end-of-file record (tapemark) then continue processing.

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. Nonlabeled 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 RESTORE Statement

The RESTORE statement restores a direct access volume or volumes from a magnetic tape volume on which a dumped copy was previously placed.

Note: When a standard label restore tape created by IBCDMPRS is restored by IEHDASDR, the DD card describing the tape for IEHDASDR can specify LABEL=(,BLP). Bypass-label-processing must have been sysgened by specifying OPTIONS=BYLABEL on the SCHEDULR control card. If bypass-label-processing is not available, any standard label tape created by IBCDMPRS cannot be restored by IEHDASDR, by providing appropriate DCB parameters on the DD statement for the tape (RECFM=U,BLKSIZE=track length).

Name	Operation	Operand
[name]	RESTORE	TODD=(ddname,...) FROMDD=ddname [CPYVOLID={YES / NO}] [PURGE={YES / NO}]

TODD=(ddname,...)
specifies the ddnames of the DD statements defining the devices containing the direct access volumes to be restored.

Note: If multiple output volumes are specified in a RESTORE statement and an abnormal completion of the RESTORE operation occurs, the operation is terminated on all output volumes.

FROMDD=ddname
specifies the ddname of the DD statement defining the magnetic 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 multivolumes.

CPYVOLID=YES
specifies that all restored direct access volumes are to be assigned the serial number of the dumped direct access volume.

If CPYVOLID=NO is coded or the CPYVOLID keyword is omitted, all receiving volumes retain their own serial numbers.

PURGE=YES
indicates that all unexpired data sets on the receiving direct access volume can be overwritten provided that the operator signals his concurrence when the first unexpired data set is encountered. The operator replies are:

Reply	Meaning
U	All unexpired data sets on this volume can be overwritten. (The RESTORE operation continues.)
T	This volume contains unexpired data sets that must not be overwritten. (The RESTORE operation is terminated.)

If PURGE=NO is coded (or the PURGE keyword is omitted) and an unexpired data set is encountered on the receiving direct access volume, the RESTORE operation is terminated.

Note: The PURGE keyword has no control over 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 RESTORE operation is terminated.

IEHDASDR Examples

The following examples show some of the uses of the IEHDASDR program.

IEHDASDR Example 1

Operation	Comments
ANALYZE (Initialize) a 2311 disk volume	1. The volume is to be initialized for the first time. 2. IPL text is included in the input stream.

In this example, a blank 2311 disk volume is to be analyzed and formatted for the first time. Since this example deals with a blank volume, two considerations must be made:

1. The TODD keyword in the ANALYZE statement must specify a channel and unit address, rather than a ddname.
2. The selected device (in this example, unit 190) must be varied offline by the operator; that is, before the job is executed, the operator must use the VARY OFFLINE command to place unit 190 offline.

- The SYSIN DD Statement: defines the control data set, which follows in the input stream.
- The ANALYZE Utility Control Statement: defines a mountable device on which a blank 2311 volume is to be mounted. This statement defines the starting location and extent of a volume table of contents, specifies a serial number and owner identification, indicates that no flag testing is to be performed, and indicates that IPL text is included in the input stream.
- The IPLTXT Utility Control Statement: signals the start of IPL text.

```
//DASDR1   JOB
//         EXEC PGM=IEHDASDR
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
      ANALYZE   TODD=190,VTOC=00004,EXTENT=00010,FLAGTEST=NO,         C
                NEWVOLID=231100,OWNERID=SMITH,IPLDD=SYSIN
       IPLTXT
 TXT
 .
 .      (IPL text)
 .
 TXT
 END             (IPL text END statement)
/*
```

IEHDASDR Example 1. Initializing a New Direct Access Volume

IEHDASDR Example 2

Operation	Comments
ANALYZE (Initialize) three volumes and change their serial numbers	1. All of the volumes have been previously initialized. 2. The ANALYZE functions are performed concurrently.

In this example, three previously initialized 2311 volumes are to be initialized and assigned new serial numbers.

- The VOL1, VOL2, and VOL3 DD Statements: define three 2311 devices on which the volumes to be initialized are mounted.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream.
- The ANALYZE Utility Control Statements: indicate the ddnames of DD statements defining devices on which the three 2311 volumes (231100, 231101, and 231102) are to be mounted. The ANALYZE statements also define starting locations and extents of the three volume tables of contents, specify new owner names and serial numbers (DISK01, DISK02, and DISK03), and indicate that no flag testing is to be performed on these volumes.

```
//DASDR2   JOB
//         EXEC PGM=IEHDASDR
//SYSPRINT DD   SYSOUT=A
//VOL1     DD   UNIT=(2311,,DEFER),DISP=OLD,
//             VOLUME=(PRIVATE,,SER=(231100))
//VOL2     DD   UNIT=(2311,,DEFER),DISP=OLD,
//             VOLUME=(PRIVATE,,SER=(231101))
//VOL3     DD   UNIT=(2311,,DEFER),DISP=OLD,
//             VOLUME=(PRIVATE,,SER=(231102))
//SYSIN    DD   *
      ANALYZE   TODD=VOL1,VTOC=00003,EXTENT=00010,FLAGTEST=NO,        C
                OWNERID=SMITH,NEWVOLID=DISK01
      ANALYZE   TODD=VOL2,VTOC=00006,EXTENT=00010,FLAGTEST=NO,        C
                OWNERID=SMITH,NEWVOLID=DISK02
      ANALYZE   TODD=VOL3,VTOC=00004,EXTENT=00010,FLAGTEST=NO,        C
                OWNERID=SMITH,NEWVOLID=DISK03
/*
```

IEHDASDR Example 2. Initializing and Assigning New Serial Numbers to Previously Initialized Volumes

IEHDASDR Example 3

Operation	Comments
GETALT (get alternate tracks on a 2321 volume) and LABEL (change the serial number of the 2321 volume)	1. This example assumes that the 2321 volume has been previously initialized.

In this example, alternate tracks are to be assigned for three suspected defective tracks on a 2321 volume.

- The VOLUME1 DD Statement: defines a device that is to contain the 2321 volume (232100).
- The SYSIN DD Statement: defines the control data set, which follows in the input stream.
- The GETALT Utility Control Statements: specify the ddname of the DD statement defining the device on which the 2321 volume is mounted. The GETALT statements specify the relative track addresses of the tracks for which alternates are to be assigned.
- The LABEL Utility Control Statement: specifies the ddname of the DD statement defining the device on which the 2321 volume is mounted. The LABEL statement changes the serial number of the 2321 volume from 232100 to DISK00.

```
//DASDR3   JOB
//         EXEC PGM=IEHDASDR
//SYSPRINT DD   SYSOUT=A
//VOLUME1  DD   UNIT=(2321,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(232100))
//SYSIN    DD   *
       GETALT   TODD=VOLUME1,TRACK=05070310
       GETALT   TODD=VOLUME1,TRACK=0A06020F
       GETALT   TODD=VOLUME1,TRACK=0A070311
        LABEL   TODD=VOLUME1,NEWVOLID=DISK00,OWNERID=SMITH
/*
```

IEHDASDR Example 3. Getting Alternate Tracks and Assigning a New Serial Number on a 2321 Volume

IEHDASDR Example 4

Operation	Number of Devices Required
DUMP onto three volumes	4 2311 DISK STORAGE DEVICES

In this example, a copy of an entire volume (231100) is to be dumped onto three volumes (231101, 231102, and 231103).

- The DUMPFROM DD Statement: defines a mountable device that is to contain a source volume.
- The DUMPTO1, DUMPTO2, and DUMPTO3 DD Statements: each defines one of three mountable devices that is to contain one of the three receiving volumes.
- The DUMP Utility Control Statement: specifies the dump operation and identifies the DD statements defining the applicable devices. All receiving volumes are to retain their own serial numbers.

```
//DASDR4   JOB
//         EXEC PGM=IEHDASDR
//SYSPRINT DD   SYSOUT=A
//DUMPFROM DD   UNIT=(2311,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(231100))
//DUMPTO1  DD   UNIT=(2311,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(231101))
//DUMPTO2  DD   UNIT=(2311,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(231102))
//DUMPTO3  DD   UNIT=(2311,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(231103))
//SYSIN    DD   *
         DUMP   FROMDD=DUMPFROM,TODD=(DUMPTO1,DUMPTO2,DUMPTO3)
/*
```

IEHDASDR Example 4. Dumping a Copy of an Entire Volume Onto Three Volumes

IEHDASDR Example 5

Operation	Number of Devices Required
DUMP a group of tracks onto a receiving volume	1 2311 DISK STORAGE DEVICE 1 system output device (printer assumed)

In this example, a copy of tracks 0 through 60 is to be dumped from a disk volume (231100) onto a system output device.

- The DUMPFROM DD Statement: defines a mountable device that is to contain the source volume.
- The DUMP Utility Control Statement: specifies the DUMP operation, identifies the DD statements defining the source and receiving devices, and identifies the tracks (0 through 60) that are to printed.

```
//DASDR5   JOB
//         EXEC PGM=IEHDASDR
//SYSPRINT DD   SYSOUT=A
//DUMPFROM DD   UNIT=(2311,,DEFER),DISP=OLD,
//             VOLUME=(PRIVATE,,SER=(231100))
//SYSIN    DD   *
         DUMP  FROMDD=DUMPFROM,TODD=SYSPRINT,BEGIN=00000000,END=00050009
/*
```

IEHDASDR Example 5. Dumping a Group of Tracks Onto a System Output Device

IEHDASDR Example 6

Operation	Number of Devices Required	Comments
DUMP a 2311 disk volume onto a magnetic tape volume	1 2311 DISK STORAGE DEVICE 1 2400 MAGNETIC TAPE DRIVE	1. This example, coded as shown, assumes that only one magnetic tape volume is required. For those applications in which additional magnetic tape volumes are required (e.g., when dumping a 2321 volume), code the serial numbers of the additional volumes in the VOLUME parameter of the DD statement defining the magnetic tape device. (For unlabeled tapes, include a volume count in the DD statement.)

In this example, a 2311 disk storage volume (231100) is to be dumped onto a 9-track, 800 bpi, magnetic tape volume (240000).

- The SOURCE DD Statement: defines a mountable device that is to contain the source volume.
- The RECEIVE DD Statement: defines a 9-track magnetic tape drive that is to contain the receiving tape volume.
- The DUMP Utility Control Statement: specifies the dump operation and identifies the DD statements defining the source and receiving devices.

```
//DASDR6   JOB
//         EXEC PGM=IEHDASDR
//SYSPRINT DD   SYSOUT=A
//SOURCE   DD   UNIT=(2311,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(231100))
//RECEIVE  DD   UNIT=(2400,,DEFER),DISP=NEW,DSNAME=TAPE1,
//              VOLUME=(PRIVATE,,SER=(240000))
//SYSIN    DD   *
         DUMP   FROMDD=SOURCE,TODD=RECEIVE
/*
```

IEHDASDR Example 6. Dumping Onto a 9-Track Magnetic Tape Volume

IEHDASDR Example 7

Operation	Number of Devices Required	Comments
RESTORE three direct access volumes from a 7-track tape volume	1 2400-2 MAGNETIC TAPE DRIVE 3 2311 DISK DEVICES	1. A 2311 disk volume was previously dumped onto the 7-track magnetic tape volume.

In this example, three disk volumes (231100, 231101, and 231102) are to be restored from a 7-track, 556 bpi, standard labeled, magnetic tape volume.

- The TAPE DD Statement: defines a 7-track magnetic tape drive that is to contain the source tape volume.
- The DIRACC1, DIRACC2, and DIRACC3 DD Statements: each defines one of three mountable devices that is to contain one of the three receiving volumes.
- The RESTORE Utility Control Statement: specifies the restore operation and identifies the DD statements defining the source and receiving devices. The receiving volumes retain their own serial numbers.

```
//DASDR7   JOB
//         EXEC PGM=IEHDASDR
//SYSPRINT DD   SYSOUT=A
//TAPE     DD   UNIT=(2400-2,,DEFER),DISP=OLD,DCB=(TRTCH=C,DEN=1),
//              VOLUME=(PRIVATE,,SER=(240000)),DSNAME=TAPE1
//DIRACC1  DD   UNIT=(2311,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(231100))
//DIRACC2  DD   UNIT=(2311,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(231101))
//DIRACC3  DD   UNIT=(2311,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(231102))
//SYSIN    DD   *
     RESTORE   TODD=(DIRACC1,DIRACC2,DIRACC3),FROMDD=TAPE
/*
```

IEHDASDR Example 7. Restoring Three Direct Access Volumes From 7-Track Magnetic Tape

IEHDASDR Example 8

Operation	Number of Devices Required	Comments
1. DUMP two direct access volumes onto two receiving direct access volumes. 2. RESTORE two direct access volumes from two magnetic tape volumes	2 2400 MAGNETIC TAPE DRIVES 4 2311 DISK STORAGE DEVICES	1. The DUMP operations are performed concurrently to minimize I/O time. 2. The RESTORE operations are performed concurrently to minimize I/O time.

In the first operation performed by this example, two direct access volumes are to be dumped concurrently onto two receiving volumes. In the second operation, two direct access volumes are to be restored concurrently from two 9-track, 800 bpi, standard labeled, magnetic tape volumes.

- The SOURCE1 and SOURCE2 DD Statements: define devices on which the source volumes for the dump operation are to be mounted.
- The TO1 and TO2 DD Statements: define devices on which the receiving volumes for the dump operation are to be mounted.
- The SOURCE3 and SOURCE4 DD Statements: define devices on which the source tape volumes for the restore operation are to be mounted.
- The TO3 and TO4 DD Statements: 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.

```
//DASDR8   JOB
//         EXEC PGM=IEHDASDR
//SYSPRINT DD   SYSOUT=A
//SOURCE1  DD   UNIT=(2311,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(231100))
//SOURCE2  DD   UNIT=(2311,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(231101))
//TO1      DD   UNIT=2311,VOLUME=SER=231102,DISP=OLD
//TO2      DD   UNIT=2311,VOLUME=SER=231103,DISP=OLD
//SOURCE3  DD   UNIT=(2400,,DEFER),DISP=OLD,LABEL=(,NL),
//              VOLUME=(PRIVATE,,SER=(240000))
//SOURCE4  DD   UNIT=(2400,,DEFER),DISP=OLD,LABEL=(,NL),
//              VOLUME=(PRIVATE,,SER=(240001))
//TO3      DD   UNIT=AFF=TO1,VOLUME=SER=231104,DISP=OLD
//TO4      DD   UNIT=AFF=TO2,VOLUME=SER=231105,DISP=OLD
//SYSIN    DD   *
         DUMP   FROMDD=SOURCE1,TODD=TO1
         DUMP   FROMDD=SOURCE2,TODD=TO2
      RESTORE   TODD=TO3,FROMDD=SOURCE3
      RESTORE   TODD=TO4,FROMDD=SOURCE4
/*
```

IEHDASDR Example 8. Performing Concurrent Operations -- Minimizing I/O Time and Minimizing the Number of Required Devices

Program Applications

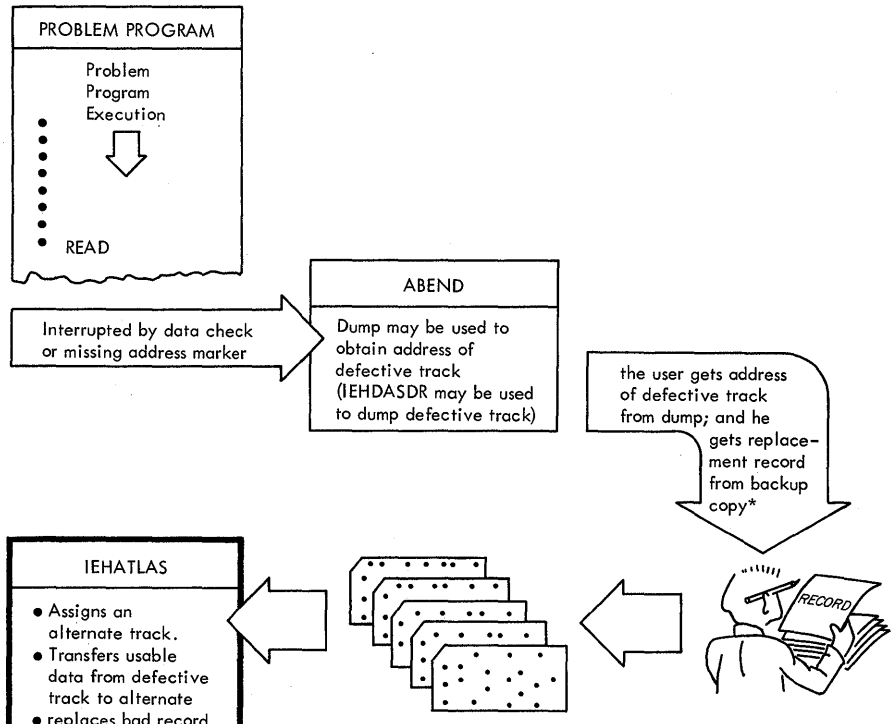
The IEHATLAS utility program is used when a defective track is indicated by a data check or missing address marker condition. It locates and assigns an alternate track to replace the defective track. Usable data records on the defective track are retrieved and transferred to the alternate track. The bad record from the defective track is then replaced on the alternate by a correct copy. (The correct copy must be provided by the user.)

The IEHATLAS utility operates under the control of the System/360 Operating System. It can be executed as a separate job, as a job step, or as a part of a user's error recovery routine. Its simplest usage will be as a separate job after an abnormal termination of a problem program (see Figure 1). Input data necessary for execution of IEHATLAS may be obtained from the dump and from backup data. A more complex usage may involve the preparation of a user's SYNAD routine which will reconstruct the necessary input data and invoke the IEHATLAS utility dynamically.

When the program is invoked, it attempts to write on the defective track. If the subsequent readback check indicates that the attempt was successful, an appropriate message is issued on the SYSOUT device. If not, a supervisor call routine (SVC 86) is entered automatically.

The SVC routine then 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, the IEHDASDR DUMP function may be used to produce a listing of error data on a track (refer to IEHDASDR for a description of the DUMP function). Using this data, the input data record for IEHATLAS can be created. The 'replace' function can then be performed under the control of the operating system by executing the IEHATLAS utility program.




```
//ATLAS   JOB    Accounting,MSGLEVEL=1
//STEP    EXEC   PGM=IEHATLAS
//SYSABEND DD    SYSOUT=A
//SYSPRINT DD    SYSOUT=A
//SYSUT1   DD    DSNAME=DATASET,UNIT=2311
//              VOLUME=SER=222222,DISP=OLD
//SYSIN    DD    *
 TRACK=BBCCHHRKDDS
INPUT(replacement record)
/*
```

*Backup copy may be a listing, punched cards, tape, etc. containing records in reserve

IEHATLAS Figure 1. Example of a Simple Application for IEHATLAS

Inputs and Outputs

Input to the IEHATLAS program consists of:

1. A description of the defective track, specifying the bin (or cell), cylinder, track, record, key, and data length (in hexadecimal notation).

2. An indication if WRITE special is needed.

3. A valid copy (in hexadecimal notation) of the bad record.

Output consists of:

1. A message, issued on the SYSOUT device, containing the user's control information, the input record, and diagnostics.

2. The input record, written onto either the original (defective) track or onto an alternate track containing the usable data taken from the defective track.

3. The return parameter list (specifying a maximum of three error record numbers in hex when an unrecoverable error occurs).

Control

The IEHATLAS program is controlled by job control statements and two utility control statements. The job control statements are used to:

- Execute or invoke the program.
- Define the work and control data set.
- Define the volumes and/or devices to be used during the course of program execution.

The utility control statement specifies whether the bad record is a member of the Volume Table of Contents or a member of some other data set. It also indicates whether or not the WRITE special command is to be used.

IEHATLAS Table 1 shows the job control statements necessary for executing or invoking the IEHATLAS program.

IEHATLAS Table 1. Job Control Statements for the IEHATLAS Program

Statement	Usage
JOB statement	This statement initiates the job.
EXEC statement	This specifies the program name (PGM=IEHATLAS) or, if the job control statements reside in a procedure library, the procedure name.
SYSABEND DD statement	This statement defines a dump data set. It must include appropriate parameters for a basic sequential (BSAM) data set. The data set can be written onto a system output device, a magnetic tape volume, or a direct access volume.
SYSPRINT DD statement	This statement defines a sequential data set which contains the output messages issued by the utility. The data set can be written onto a system output device, a magnetic tape volume, or a direct access volume.
SYSUT1 DD statement	This statement defines the data set which contains the bad record. (RESTRICTION: DISP=SHR must not be used for the SYSUT1 DD Statement
SYSIN DD statement	This statement defines the control data set. It contains the utility control statement and a copy of the bad record. Any input device supported by the system may be specified.
Note: The blocksize for the SYSPRINT (message) data set must be a multiple of 121. The blocksize for the SYSIN (control) data set must be a multiple of 80. Any blocking factor can be specified for these blocksizes.	

Utility Control

The utility control statement consists of either:

```
TRACK=bbbbccccchhhhrrkkdddd[S]
```

or

```
VTOC=bbbbccccchhhhrrkkdddd
```

The parameter bbbbccccchhhhrrkkdddd represents ten bytes of hexadecimal information.

bbbb
is the bin (or cell) number when the device specified in the SYSUT1 DD Statement is s 2321 data cell; if the device is other than a 2321 data cell, the number must be padded with zeros.

cccc
is the number of the cylinder in which the defective track was found.

hhhh
is the defective track number.

rrkk
is the record number and key length for the bad record.

dddd
is the data length of the bad record. (When a WRITE Special command is used, dddd is the length of the record segment.

S
is an optional byte of EBCDIC information which specifies that the WRITE Special command is to be used (when the last record on the track overflows and must be completed elsewhere).

Note 1: Care should always be taken to insure that the input record data length does not exceed track size. This is especially important when the WRITE Special command is specified because the error may not be recognized immediately by the system.

Note 2: The utility control 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 (1-80) are used on the additional cards.

IEHATLAS Examples

The following examples illustrate some of the uses of the IEHATLAS program.

```
//JOBATLAS JOB  06#990,SMITH,MSGLEVEL=1
//STEP     EXEC PGM=IEHATLAS
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=NEWSET,UNIT=2311,VOLUME=SER=231111,DISP=OLD
//SYSIN    DD   *
 TRACK=00000002000422020006S
F3F1C2C2F0F00000
/*
```

IEHATLAS Example 1. Assigning an Alternate Track on a Direct Access Device-Write Special Included Because of a Track Overflow Condition

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.

- The SYSPRINT DD Statement: defines the device onto which the output messages can be written.
- The SYSUT1 DD Statement: defines the data set which contains the bad record.
- The SYSIN DD Statement: defines the control data set which follows in the input stream.
- The TRACK Utility Control Statement: represents the bin, cylinder, and track number for the defective track as well as the record number, key length and data length of the bad record. In this example, the input record has a bin number of zero (since the device is a 2311); it is to be placed on cylinder two, track four, record 22; and 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 the TRACK utility control statement. The input record contains eight bytes (data length = 6, key length = 2).

```
//JOBATLAS JOB  06#990,SMITH,MSGLEVEL=1
//STEP     EXEC PGM=IEHATLAS
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=EOFSET,UNIT=2311,VOLUME=SER=33333,DISP=OLD
//SYSIN    DD
 TRACK=00000001000003000000
/*
```

IEHATLAS Example 2. Assigning as Alternate Track for a Bad End-of-File Record

- The SYSPRINT DD Statement: defines the device onto which the output messages can be written.
- The SYSUT1 DD Statement: defines the data set which contains the bad record.
- The SYSIN DD Statement: defines the control data set which follows in the input stream.
- The TRACK Utility Control Statement: defines an End-of-file record on cylinder one, track zero, record three, Input data -- other than the utility control statement -- is not required.

```
//JOBTALAS JOB  06#990,SMITH,MSGLEVEL=1
//STEP     EXEC PGM=IEHATLAS
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=2311,VOLUME=SER=22222,DISP=OLD
//SYSIN    DD   *
 VTOC=000000000005022C0060
D6C2D1C5C3E340404040404040404040404040404040404040404040404040404040404040404040
40404040F1F2F3F1F1F0F000014401360000000100000000000000000000000000000000000000000
00004000800000F000F0003333333333333333310DDD00000100000A0000000A00000000000000
00000000000000000000000000000000000000000
/*
```

IEHATLAS Example 3. Assigning an Alternate Track for a Bad VTOC Record

- The SYSPRINT DD Statement: defines the device onto which the output messages can be written.
- The SYSUT1 DD Statement: defines the data set which contains the bad record.
- The SYSIN DD Statement: defines the control data set which follows in the input stream.
- The VTOC Utility Control Statement: defines the location of the bad VTOC record as being on track five of cylinder zero. The record number is two 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 utility control statement. The input record contains 140 bytes (data length = 96, key length = 44).

Program Applications

The IFHSTATR program selects, formats, and writes information from type 21 (error statistics by volume) records. Error statistic by volume (ESV) records should be retrieved from the IFASMFDP 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, the IFHSTATR program does not clear the SYS1.MANX (or SYS1.MANY) data set and make it available for additional records.

Inputs and Outputs

IFHSTATR Figure 1 shows the structure of the type 21 (ESV) record, which contains information about errors on magnetic tape. Only ESV records are processed by the IFHSTATR program; if none are found, an appropriate message is written to SYSUT2.

0	TOTAL RECORD LENGTH		'00'	
4	RESERVED	RECORD TYPE X'15'	TIME OF DAY	
8	TIME OF DAY (continued)		CURRENT DATE	
12	CURRENT DATE (continued)		CPU ID	
16	MODEL NO.		DATA LENGTH	
20	VOLUME SERIAL NO.			
			CHANNEL/UNIT ADDRESS	
28	UCB TYPE			
32	TEMPORARY READ ERRORS	TEMPORARY WRITE ERRORS	START I/O'S	
36	PERMANENT READ ERRORS	PERMANENT WRITE ERRORS	NOISE BLOCKS	ERASE GAPS
40	ERASE GAPS (continued)	CLEANER ACTIONS		TAPE DENSITY
44	BLOCK SIZE		RESERVED	

IFHSTATR Figure 1. Type 21 (ESV) Record Format

The IFHSTATR program can write on any output device supported by BSAM. The output takes the form of 121-byte unblocked records, with an ASA control character is in the first byte of each record. IFHSTATR Figure 2 shows a sample of printed output from the IFHSTATR program.

VOLUME SERIAL	DATE	CPU ID	MOD NO	TIME OF DAY	CHANNEL / UNIT	TEMP READ	TEMP WRITE	PERM READ	PERM WRITE	NOISE BLOCKS	ERASE GAPS	CLEANER ACTIONS	USAGE (SIO'S)	TAPE DENSITY	BLOCK 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

IFHSTATR Figure 2. Sample Output from the IFHSTATR Program

Control

The IFHSTATR program is controlled by the job control statements shown in IFHSTATR Table 1.

IFHSTATR Table 1. Job Control Statements for the IFHSTATR Program.

Statement	Usage
JOB	This statement initiates the job.
EXEC	This statement specifies the program name (PGM=IFHSTATR).
SYSUT1 DD	This statement defines the input data set and the device on which it resides. The DSNAME, UNIT, VOLUME, LABEL and DISP parameters should be included.
SYSUT2 DD	This statement defines the sequential data set on which the output is to be written. The blocksize for SYSUT2 must be a multiple of 121. Any blocking factor can be specified.

IFHSTATR Example

The following is an example of the JCL needed to produce a report such as that shown in IFHSTATR Figure 2.

```
//          JOB
//          EXEC PGM=IFHSTATR
//SYSUT1    DD   UNIT=2400,DSNAME=SYS1.MAN,LABEL=(,SL),                    X
//               VOLUME=SER=VOLID,DISP=OLD
//SYSUT2    DD   SYSOUT=A
/*
```

Section 2: Data Set Utilities

Data set utility programs 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. This section describes the capabilities, requirements for execution, and examples of the use of each data set utility program.

- IEBCOPY -- a program that copies or merges partitioned data sets, selects or excludes specified members in a copy operation/step, renames and/or replaces selected members of a partitioned data set(s).
- IEBGENER -- a program that copies records from a sequential data set or converts a data set from sequential to partitioned organization.
- IEBCOMPR -- a program that compares records in sequential or partitioned data sets.
- IEBPTPCH -- a program that prints or punches records that reside in a sequential or partitioned data set.
- IEBTCRIN -- a program that constructs records from the input data stream read from the 2495 Tape Cartridge Reader.
- IEBUPDTE -- a program that incorporates changes to sequential or partitioned data sets.
- IEBISAM -- a program that places source data from an indexed sequential data set into a sequential data set in a format suitable for subsequent reconstruction.
- IEBEDIT -- a program that creates an input stream.
- IEBUPDAT -- a program that incorporates changes to symbolic libraries.
- IEBDG -- a program that creates a test data set consisting of patterned data.

A data set utility program is executed or invoked through the use of job control statements and utility control statements.

Job Control Statement Requirements

Data set utility programs are introduced as jobs or job steps, and are executed in response to job control statements and utility control statements. In general, a program to be executed requires:

- A JOB statement.
- An EXEC statement that identifies the program to be executed.
- DD statements that define data sources and destinations.

METHODS OF EXECUTION

A set of job control statements for a utility program can be introduced to the operating system in different ways. The statements can be included in the input stream, or they can be placed in a procedure library, or the job can be invoked by a calling program. Refer to Methods of Execution in "Section 1: System Utilities" for a complete discussion of these methods.

MVT CONSIDERATIONS (MULTIPROGRAMMING WITH A VARIABLE NUMBER OF TASKS)

In an MVT environment, a region size should be specified for each application of the program. The region size is determined by the number of bytes in the utility program and by the block size of an object data set (i.e., the largest block size in the job step). A region size can be specified as a parameter in the EXEC statement specifying the utility program name. The minimum region sizes are:

- IEBCOPY -- REGION=28K + 2(largest blocksize in the job step) rounded to the next highest 2K. Also see IBM System/360 Operating System: Storage Estimates, GC28-6551.
- IEBGENER -- REGION=14K + b
- IEBCOMPR -- REGION=14K + 2b
- IEBPTPCH -- REGION=16K + b
- IEBTCRIN -- REGION=12K + 2(BUFL on SYSUT1) + maximum logical record length (rounded to next highest 2K) + sum of all user exit routines (each rounded to next highest 2K).
- IEBUPDTE -- REGION=14K + 2b
- IEBISAM -- REGION=8K
- IEBEDIT -- REGION=10K
- IEBUPDAT -- REGION=10K + 2b

  where b is the largest block size in the job step rounded to the next highest 2K.

- IEBDG -- varies, see IBM System/360 Operating System: Storage Estimates, GC28-6551, for information on calculating REGION size required for a particular application.

NOTE: A job that modifies a system data set (i.e., a SYS1. data set) 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 an MVT environment. The operator should be informed of all jobs that modify system data sets.

DD statements defining object data sets should ensure that the volumes on which the data sets reside are nonsharable.

Utility Control Statement Specifications

All of the data set utility programs except IEBISAM use utility control statements to control the functions of the programs. (The IEBISAM program is controlled solely by job control statements.)

Within a job step, the utility control statements normally follow the job control statements in the input stream; however, the utility control statements can alternatively be placed in a sequential data set or in a partitioned data set, as a member of that data set.

Refer to each utility program for a discussion of the functions performed by the utility control statements.

Program Description

(Note: This is a description of the new version of the IEBCOPY program. The program is designed to accept the job and control statements written for the earlier version. However, it is recommended that any future user applications be written to the specifications discussed in this section.)

The IEBCOPY program can copy one or more partitioned data sets or merge partitioned data sets. Specified members of a partitioned data set(s) can be selected for, or excluded from a copy operation.

The program can be used to:

- Create a back-up copy.
- Copy one or more data sets per copy operation.
- Select members, from one or more data sets, to be copied.
- Exclude members, from one or more data sets, from being copied.
- Compress a data set in place.
- Merge data sets.
- Optionally replace identically named members on data set(s).
- Optionally rename selected members.
- Recreate a data set that has exhausted its primary, secondary, or directory space allocation.

In addition, the IEBCOPY program automatically lists the number of unused directory blocks and the number of unused tracks available for member records in the output partitioned data set. By using the LIST=NO operand the program also suppresses the names of copied members listed by input 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.

Inputs and Outputs

IEBCOPY Table 1 lists the major inputs to and outputs from the IEBCOPY program.

Acceptable Devices

All input, output, and utility data sets must be on direct access devices. The following devices may be used:

- 2311 Disk Storage Drive.
- 2314 Direct Access Storage Facility.
- 2302 Disk Storage.
- 2303 Drum Storage.
- 2301 Drum Storage.
- 2321 Data Cell Drive.

Any combination of these devices is acceptable to the IEBCOPY program.

IEBCOPY Table 1. Data Sets Used (Input) and Produced (Output) by the IEBCOPY Program

Inputs	Input Data Set: This data set contains the members to be copied or merged into a partitioned data set. Control Data Set: This data set contains utility control statements. The data set is required if selected members are to be copied, merged into a partitioned data set, or omitted from the copy or merge operation. If the data set is null, a full copy from SYSUT1 to SYSUT2 is attempted. (Note: In this case, SYSUT1 and SYSUT2 are required ddnames for the input partitioned data set and output partitioned data set, respectively.)
Outputs	Output Data Set: This data set contains the copied or merged data. It is either a new data set (copy operation) or an old data set (merge or compress in place operation). Message Data Set This data set contains informational messages (e.g., the names of copied members; the contents of applicable utility control statements) and error messages, if applicable.
Temporary Utility Outputs	Spill Data Set: These data sets are temporary utility data sets used to provide space when there is not enough core available for the input and/or output partitioned data set directories. These data sets are opened only when needed. (Note: Refer to IBM System/360 Operating System: Storage Estimates, GC28-6551, to determine when spill data sets are required; see "Space Allocation" in this section for a description of how to determine the amount of space to allocate.)

Additional Outputs

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

00 -- successful completion.
04 -- a condition has occurred from which recovery may be possible.
08 -- an unrecoverable error has occurred. The job step is terminated.

Control

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

JOB CONTROL STATEMENTS

(Note: This is a description of the new version of the IEBCOPY program. The program is designed to accept the job control statements written for the earlier version. However, it is recommended that all future user applications be written with the new job control statements.) IEBCOPY Table 2 shows the job control statements necessary for executing or invoking the IEBCOPY program.

IEBCOPY Table 2. Job Control Statements for the IEBCOPY Program (Part 1 of 2)

Statement	Usage
JOB Statement	This statement initiates the job.
EXEC Statement	This statement specifies the program name (PGM=IEBCOPY), or if the job control statements for the IEBCOPY program reside in the procedure library, the procedure name.
SYSPRINT DD Statement	This statement defines the sequential message data set used for listing statements and/or messages. This data set can be written onto a system output device, a magnetic tape volume, or a direct access volume, and must be in fixed blocked or fixed format.
Input/Output DD Statement	Each statement defines input/output partitioned data sets. The statements must describe partitioned data sets on direct access devices. Each data set can be defined by a data set name, or as being a cataloged data set, or as a data set passed from a previous job step. User Option: These statements define either input or output data sets. A minimum of two statements is required (except for a compress in place): one describing the input partitioned data set, and one describing the output partitioned data set. In addition, there must be a DD statement for each unique partitioned data set used in the job step. Input partitioned data sets cannot be concatenated.
SYSUT3 DD Statement	This statement defines a spill utility data set. It must define a data set on a direct access device. SYSUT3 is used when there is no space in core for some or all of the 'current' input partitioned data set's directory entries. Note: Refer to IBM System/360 Operating System: Storage Estimates, GC28-6551, to determine when spill data sets are required; see "Space Allocation" in this section for a description of how to determine the amount of space to allocate.
SYSUT4 DD Statement	This statement defines a spill utility data set. It must define a data set on a direct access device. SYSUT4 is used when there is no space in core for the current output partitioned data set's merged directory and the output partitioned data set is not 'new'. Note: Refer to IBM System/360 Operating System: Storage Estimates, GC28-6551, to determine when spill data sets are required; see "Space Allocation" in this section for a description of how to determine the amount of space to allocate.
SYSIN DD Statement	This statement 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 magnetic tape volume, or a direct access volume, and must be in fixed blocked or fixed format.

IEBCOPY Table 2. Job Control Statements for the IEBCOPY Program (Part 2 of 2)

Notes: SYSPRINT and SYSIN are mandatory DD statements. The blocksize for the SYSPRINT (message) data set must be a multiple of 121. The blocksize for the SYSIN (control) data set must be a multiple of 80. Any blocking factor may be specified for these data sets, with a maximum allowable blocksize of 32,767 bytes.

Fixed length or variable length records can be reblocked. Reblocking or deblocking is done if the blocksize of the input partitioned data set is not equal to the blocksize of the output partitioned data set. Reblocking or deblocking cannot be done if either the input or the output data set has undefined format (U-format) records, keyed records, track overflow records, note lists and/or user TTRN's, or if compress in place is specified. (Earlier versions allowed reblocking or deblocking with track overflow output records.) The following chart shows the only allowable ways to change input record formats:

INPUT	OUTPUT
Fixed	Fixed Blocked
Fixed Blocked	Fixed
Variable	Variable Blocked
Variable Blocked	Variable

In addition, any record format can be changed to the Undefined format (in terms of its description in the DSCB).

System data sets should not be compressed in place in a multiprogramming environment unless the subject partitioned data set is made "nonsharable."

Refer to the publication IBM System/360 Operating System: Supervisor and Data Management Services, GC28-6646 for information on estimating space allocations.

Refer to "IEBCOPY Examples" in this section for typical uses of the job control statements.

Space Allocation

Sometimes it is necessary to allocate space on the spill utility data sets (SYSUT3 and SYSUT4).

To conserve space on the direct access volume, an initial quantity and a secondary quantity for space allocation may be used as follows:

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

- c = { 80 for SYSUT3 / 256 for SYSUT4 }
- x = primary allocation
- y = secondary allocation
- (x + 15y) ≥ { e for SYSUT3 / b for SYSUT4 },where
- e = the number of entries in the largest input partitioned data set in the copy operation.
- b = the number of blocks allocated to the largest output partitioned data set directory in the IEBCOPY job step.

In the worst possible case, SYSUT3 requires a space allocation of SPACE=(80,e).

SYSUT4, in the worst possible case, requires a space allocation of SPACE=(256,b).

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

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

However, the total amount of space required for SYSUT3 in the worst case situations would be 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)...", for use in allocation of space by the operating system. Note that the IEBCOPY utility program ignores all other DCB information specified for SYSUT3 and/or SYSUT4. Multi-volume SYSUT3 and SYSUT4 data sets are not supported.

Alias Processing

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

- When the main member and its alias(es) are copied, they exist on the output partitioned data set in the same relationship which they had on the input partitioned data set.
- When one alias is copied without its main member it becomes a main member.
- When two or more aliases are copied without the main member, the lowest alias name (in alphameric collating sequence) becomes the main member; the remaining alias(es) becomes an alias(es) of the 'new' main member. However, it must be noted that if the 'old' main member name is present in the alias entry(s), it remains there.

The rules for replacing or renaming members apply to both aliases and members without making a distinction between them.

UTILITY CONTROL STATEMENTS

(Note: This is a description of the new version of the IEBCOPY program. The program is designed to accept the control statements written for the earlier version. Although the earlier version are acceptable, the control statements used in different versions must not be mixed. It is recommended that all future user applications be written with the new control statements.)

The IEBCOPY program uses three control statements:

- The COPY Statement.
- The SELECT Statement.
- The EXCLUDE Statement.

In addition, when the INDD 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 placed in columns 2-71; i.e., no data may be placed in column 1, or in columns 72-80. All other rules discussed in Appendix C apply.

The COPY Statement

The COPY statement is required for all copy operations in the IEBCOPY program. The following list of abbreviations are acceptable to the IEBCOPY program on the COPY statement:

WORD	ABBREVIATION
COPY	C
INDD	I
OUTDD	O
LIST	L

The statement contains:

Name	Operation	Operand(s)
[name]	COPY	OUTDD=ddname,INDD={ddname1[,ddname2]... / ddname1[,ddname2][,(ddname3,R)]... / ((ddname1,R)[,ddname2]...)}[1] [,LIST=NO]
		[1]The INDD parameter may appear on a separate card; if this option is selected, the INDD parameter is not preceded by a comma (,).

OUTDD=ddname (one ddname per copy operation; the ddname used must be specified on a DD statement)
specifies the name of the output partitioned data set.

LIST=NO
specifies that names of members copied are not to be listed on SYSPRINT at the end of each input data set. 'NO' is the only valid parameter with this optional keyword.

INDD=ddname1 [,ddname2]... (each ddname referenced must be specified on a DD statement)
specifies the name(s) of the input partitioned data set(s). Any number of ddnames may be specified, but only one INDD keyword may be specified per card. If more than one ddname is specified, the input data sets are processed in the same sequence as the ddnames. The INDD operand may, optionally, be placed on a separate card following:

- A COPY statement containing the OUTDD keyword.
- Another INDD statement.
- A SELECT statement.
- An EXCLUDE statement.

INDD={ddname1[,(ddname2,R)]... / ((ddname1,R),(ddname2[,R]))...} (each ddname must be specified on a DD statement)
the 'R' parameter specifies replace. It is an optional feature that specifies that all members to be copied from this input partitioned 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 as usual.) When this option is specified with the INDD operand it does not have to appear with the MEMBER parameter (discussed in "The SELECT Statement" in this section) 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 the INDD field the entire field, exclusive of the INDD keyword, must be enclosed in a second set of parenthseses.

Notes: Only one INDD and one OUTDD keyword may be placed on a single card. The OUTDD operand must appear on the COPY statement. These two keywords may appear in any order on the statement. When the INDD operand appears on a separate card no other operands may be specified on that card.

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 in which the ddnames are specified.

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

The compress in place function is normally invoked by specifying the same ddname (with the same dsname and volume serial specified on the DD statement) 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 any 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. The compress in place operation cannot be performed for the following:

- 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.

The SELECT Statement

The SELECT statement specifies which members are to be selected from the input partitioned data set(s) to be copied onto an output partitioned data set. This statement is also used to rename and/or replace selected members on the output partitioned 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. One or more INDD statements must precede the SELECT statement. A SELECT statement cannot appear with an EXCLUDE statement in the same copy step; however, both may be used in a single copy operation.

It must be remembered that when a selected member is found on an input partitioned data set it is not searched for again, regardless of whether or not it is copied. 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 INDD level or the MEMBER level. (For a description of replacing identically named members see "Replacing Identically Named Data Set Members," and" Replacing Selected Members" in the "Program Applications" portion of this section.) 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.

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' membername and 'new' membername must be enclosed in a set of parentheses; if it is specified for the first membername in the MEMBER field, the entire field, exclusive of the MEMBER keyword, must be enclosed in a second set of parentheses.

The following list of abbreviations are acceptable to the IEBCOPY program on the SELECT statement:

WORD	ABBREVIATION
SELECT	S
MEMBER	M

The statement contains:

Name	Operand	Operand(s)
[name]	SELECT	MEMBER={[(]membername1[,membername2 / ,(membername2,,R) / ,(membername2,newname[,R])]...[)] / ((membername1{,newname[,R]) / ,,R}[membername2]...)}

MEMBER=(list of member names and/or aliases)
specifies which members are to be selected from the input partitioned data set. Each member name specified within one copy step must be unique; i.e., duplicate names cannot be specified as either old names, or new names, or both, under any circumstances.

It also allows the user to rename selected members. The member is copied onto 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 replace is also specified.

In addition, it allows the user to replace an identically named member(s) that exist on the output partitioned data set.

The EXCLUDE Statement

The EXCLUDE statement specifies members which are to be excluded from the copy step. Unlike the selective copy, the exclusive copy causes all specified members on each input partitioned data set to be excluded from the copy. 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. An INDD statement must precede the first EXCLUDE statement. An EXCLUDE statement cannot appear with a SELECT statement in the same copy step; however, both may be used in a single copy operation.

The following list of abbreviations are acceptable to the IEBCOPY program on the EXCLUDE statement:

WORD	ABBREVIATION
EXCLUDE	E
MEMBER	M

The statement contains:

Name	Operation	Operand(s)
[name]	EXCLUDE	MEMBER=[(]membername1 [,membername2]...[)]

MEMBER=(list of member names and/or aliases)
specifies member(s) on the input partitioned data set(s) that are not to be copied onto the output partitioned data set. The members are not deleted from the input partitioned 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.

Control Statement Sequence

A COPY statement must precede a SELECT and/or EXCLUDE statement when members are selected for or excluded from the copy operation/step. In addition, if the input ddname(s) is specified on a separate INDD statement(s), 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 a /* card or another COPY statement, a full copy is invoked onto the most recent output partitioned data set previously specified.

Using the Utility Control Statements

IEBCOPY Table 3 shows some sample uses of the utility control statements. The samples show the minimum required statement(s) for each application.

COPY OPERATION/COPY STEP CONCEPT

The IEBCOPY program introduces a concept that is similar to the job step concept in job control: the copy operation/copy step concept. The unit of work that begins with a COPY utility control statement and continues until a delimiter statement (/*) or another COPY utility control statement is found is called a copy operation. Thus, the COPY utility control statement is the first control statement in every copy operation.

Within a copy operation are one or more copy steps. The unit(s) of work that is delimited by a COPY utility control statement, or an INDD statement following a SELECT or EXCLUDE statement, or a delimiter statement (/*) is a copy step.

IEBCOPY Table 3. Use of the COPY, SELECT, and EXCLUDE Statements in the IEBCOPY Program (Note: Cards are read from bottom to top within each sample.)

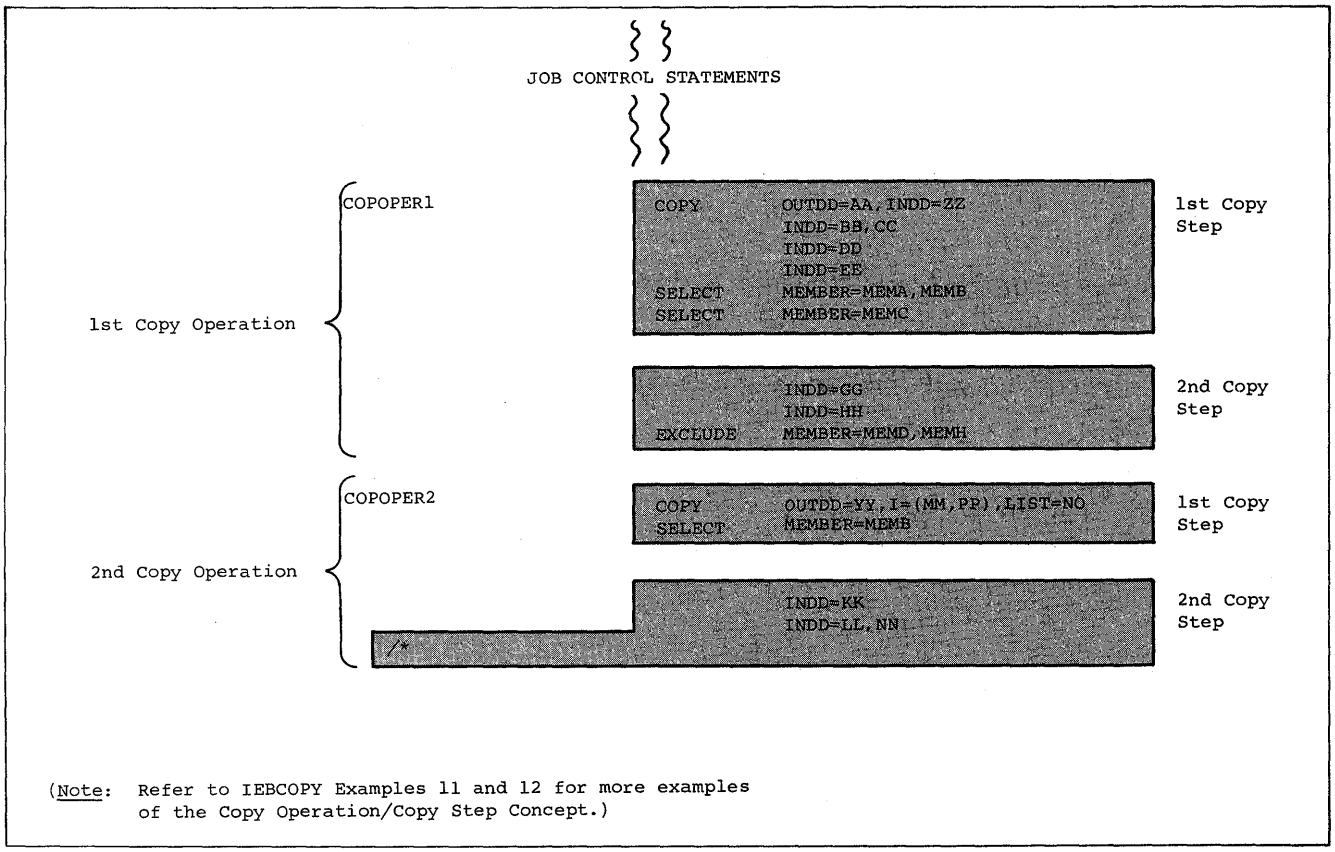
To perform a	The SYSIN Data Set Contains
Full Copy	[name] COPY INDD=ddname,OUTDD=ddname2
	INDD=ddname1 [name] COPY OUTDD=ddname2 The INDD operand may be placed on a separate card, as shown here, in any application.
Replace (INDD level)	[name] COPY OUTDD=ddname1,INDD=ddname2,(ddname3,R)
	[name] COPY OUTDD=ddname2,INDD=((ddname1,R))
Selective Copy	[name] SELECT MEMBER=name1 [name] COPY OUTDD=ddname1,INDD=ddname2
Selective Replace (MEMBER level)	[name] SELECT MEMBER=((name1,,R)) [name] COPY OUTDD=ddname1,INDD=ddname2
	[name] SELECT MEMBER=name1,(name2,,R) [name] COPY OUTDD=ddname1,INDD=ddname2
Selective Rename	[name] SELECT MEMBER=name1,(oldname,newname) [name] COPY INDD=ddname1,OUTDD=ddname2
	[name] SELECT MEMBER=((oldname,newname)) [name] COPY INDD=ddname1,OUTDD=ddname2
Selective Rename and Replace and	[name] SELECT MEMBER=name1,(oldname,newname,R) [name] COPY OUTDD=ddname1,INDD=ddname2
	[name] SELECT MEMBER=((oldname,newname,R)) [name] COPY INDD=ddname1,OUTDD=ddname2
Exclusive Copy	[name] EXCLUDE MEMBER=name1 [name] copy OUTDD=ddname1,INDD=ddname2
Compress In Place	[name] COPY OUTDD=sameddnm,INDD=sameddnm

IEBCOPY Figure 1 shows the copy operation/copy step concept. In the figure are two copy operations: the first begins with the statement containing the name COPOPER1 and ends with the EXCLUDE statement preceding the statement containing the name COPOPER2; the second operation begins with the statement containing the name COPOPER2 and ends with the /* (delimiter) statement.

Within the first copy operation are two copy steps: 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.

Within the second copy operation there are also two copy steps: the first begins with the COPY statement and continues through the SELECT statement; the second copy step begins with the INDD statement immediately following the SELECT statement and ends with the same /* (delimiter) statement that ended the copy operation.

Note: There is no limit to the number of steps allowed within a single copy operation; nor is there a limit to the number of copy operations allowed within a single job step.



IEBCOPY Figure 1. Copy Operation/Copy Step Concept

Program Applications

Creating a Back-Up Copy

A partitioned data set can be copied, totally, or in part, from one direct access volume to another. In addition, a data set can be copied onto its own volume, provided its data set name is changed. (If the data set name is not changed, it is compressed in place.) IEBCOPY Example 1, in the back of this section, shows a copy operation in which a back-up copy is made. Note that the copied members are not reordered; i.e., they are copied in the order in which they exist on the original data set. If the members, themselves, are to be collated, the IEHMOVE system utility program can be used for the copy operation. Refer to "Section 1: System Utility Program" for a discussion of the IEHMOVE program.

Copying From More Than One Input Partitioned Data Set

More than one input partitioned data set can be copied, totally or in part, from one or more direct access volumes to a single direct access volume. This can be done by specifying more than one parameter for the INDD operand. See "The COPY Statement" in this section for a discussion on specifying more than one input partitioned data set. The input partitioned data sets are copied in the order in which they are specified on the INDD statement.

IEBCOPY Example 2, in the back of this section, shows a copy operation in which members are copied from three input partitioned data sets onto an existing output partitioned data set. The sequence in which the control statements occur is important; this controls the manner and sequence in which partitioned data sets are processed.

Replacing Identically Named Data Set Members

In many copy operations, the output partitioned data set may contain some members that have names identical to the names of the input partitioned data set members to be copied. When this occurs, the user may specify the replace option. This causes the identically named members to be copied from the input partitioned data set, and the pointer in the output partitioned data set directory to be changed to point to the member; i.e., the replace option allows the user to cause an input member to override an existing member on the output partitioned data set with the same name. 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 two levels:

- The INDD (data set) level; or
- The MEMBER level.

The level is determined by using the "R" parameter on an INDD statement (for the INDD level), or with a MEMBER parameter on a SELECT statement (for the MEMBER level).

When replace is specified on the INDD level, the input data is copied as follows:

1. In a full copy operation, all members on an input partitioned data set(s) are copied onto an output partitioned data set; members whose names already exist on the output partitioned data set are replaced by the members copied from the input partitioned data set(s).

2. In a selective copy operation, all selected members on an input partitioned data set(s) are copied onto an output partitioned data set; all selected members "found" are copied and members whose names already exist on the output partitioned data set are replaced by the "found" members copied from the input partitioned data set(s).

3. In an exclusive copy operation, all non-excluded members on an input partitioned data set(s) are copied onto an output partitioned data set; non-excluded input members whose names already exist on the output partitioned data set replace those duplicate named members on the output partitioned data set.

When replace is specified on the MEMBER level, only selected members on the input partitioned data set(s) are copied, and identically named members on the output partitioned data set are replaced.

There are differences between full, selective, and exclusive copy operations that should be remembered when specifying the replace option with all of the multiple data sets containing member names common to some/all of the input partitioned data sets being copied:

1. When a full copy is performed, the output partitioned data set will contain the replacing members which were on the last input partitioned data set copied.

2. When a selective copy is performed, the output partitioned data set will contain 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, and if the same member exists on another input partitioned data set it is not searched for, and hence, not copied.

3. When an exclusive copy is performed, the output partitioned data set will contain the non-excluded replacing members which were on the last input partitioned data set copied.

IEBCOPY Examples 3 and 6, in the back of this section, show copy operations in which replace is specified on the INDD level.

Selecting Members to be Copied

Members can be selected from one or more input partitioned data sets. The SELECT option allows the user to copy only selected members from the input partitioned data set(s) specified on the INDD statement(s) preceding the SELECT utility control statement(s). Consecutive SELECT statements can be used in a copy step; when this occurs the second and subsequent statements are treated as a continuation of the first.

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

When selecting members from an input partitioned data set(s), it is important to remember that once a member is found it is not searched for on any subsequent input partitioned data set(s). Similarly, when all of the selected members are found, the copy step is terminated although all of the input partitioned data sets may not have been searched. For example, if members A and B are specified on the SELECT statement and A is found on the first of three input partitioned data sets, it is not searched for again; if B is found on the second input partitioned data set, the copy step is successfully terminated after the second input

partitioned data set has been processed, although both members may also exist on the third input partitioned data set.

However, if the first member name is not found on the first input partitioned data set, the second selected member is searched for; if it is not found, the third is searched for, and so on. This process continues until there are no more members to be searched for in this input partitioned data set. All the members that were found on the input partitioned data set are then processed for copying onto the output partitioned data set. This process is repeated for the second input partitioned data set (except that the members that were found on the first input partitioned data set are not searched for again). IEBCOPY Example 4, in the back of this section, shows a copy operation in which selected members are copied.

Replacing Only Selected Members

The user may specify the replace option on either the INDD level or the MEMBER level when members are being selected for copying. If the option is specified on the INDD level, all selected members found on the designated input partitioned data set(s) 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 option is specified on the MEMBER level, only the specified members on the input partitioned data set(s) replace identically named members on the output partitioned data set. Once a member is found it is not searched for again. IEBCOPY Example 5, in the back of this section, shows a copy operation in which a selective replace is specified on the MEMBER level. (Also refer to "Replacing Identically Named Data Set Members" in this section.)

Renaming Selected Members

Selected members on input partitioned data set(s) can be copied and renamed on the output partitioned data set. However, if the new name is identical to a member name on the output partitioned data set, the input member is not copied unless the replace option is also specified. See "The SELECT Statement" in this section for information on renaming selected members. IEBCOPY Example 7, in the back of this section, shows a copy operation in which a member is renamed and replaces an identically named member on the output partitioned data set.

(Note: Renaming is not physically done to the input partitioned data set directory entry. However, after the member is copied onto the output partitioned data set, the new name is entered into the output partitioned data set directory.)

Excluding Members From A Copy Operation

Members from one or more input partitioned data sets can be excluded from being copied. Unlike the SELECT option, when a member is specified on the EXCLUDE statement it is not copied from any of the input partitioned data sets; i.e., the excluded member is searched for on every input partitioned data set in the copy step, and always omitted from the copy. Like the select option, it excludes members only from the input partitioned data set(s) referenced by the INDD statement(s) that precede it in the copy step.

The replace option can be specified on the INDD level in an exclusive copy, in which case, non-excluded members on the input partitioned data set replace identically named members on the output partitioned data set. See "Replacing Identically Named Data Set Members" in this section for more information on the replace option.

Consecutive EXCLUDE statements can be used in a copy step; when this occurs the second and subsequent statements are treated as a continuation of the first. IEBCOPY Example 8, in the back of this section, shows a copy operation in which members are excluded from a copy; in addition, replace is specified for one input partitioned data set.

Compressing A Data Set

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

In order 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 simply by specifying the same data set for both the input (INDD) and the output (OUTDD) parameters in a copy operation/step. It is recommended that a back-up copy of the partitioned data set to be compressed in place be retained 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.

When a compression is invoked by specifying the same ddname for the INDD and OUTDD keywords, and the DD statement specifies a blocksize that differs from the blocksize specified in the DSCB, the DSCB blocksize is overridden; however, no physical re/de-blocking is done by the IEBCOPY program.

IEBCOPY Examples 9 and 10, in the back of this section, show data sets being compressed in place.

Merging Data Sets

A merged data set is one onto which an additional member(s) is copied. It is created by copying the additional member(s) onto an existing output partitioned data set; the merge operation -- i.e., the ordering of the output partitioned data set's directory -- is automatically performed by the IEBCOPY program.

Note: If there is a question about whether or not enough directory blocks are allocated to the output partitioned data set onto which an input partitioned data set is being merged, the output partitioned data set should be 'recreated' prior to the merge operation. (See "Recreating A Data Set" in this section.)

IEBCOPY Examples 2, 3, 4, 5, 6, 7, 8, 10, 11, and 12, in the back of this section, show copy operations in which members are merged onto existing output partitioned data sets.

Recreating A Data Set

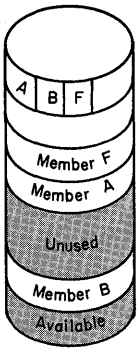
A data set can be recreated by copying it and allocating a larger amount of space than was allocated for the original data set. This application of the program is especially useful if insufficient directory space was allocated to a data set. Space cannot be allocated in this manner for an existing data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

IEBCOPY Examples

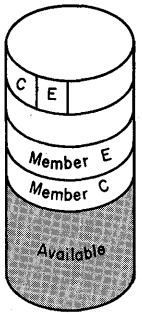
The following examples illustrate some typical uses of the IEBCOPY program. Examples 1-10 use the partitioned data sets pictured below; Examples 11 and 12 use partitioned data sets suitable to the particular situation described in each.

PARTITIONED DATA SETS
DEFINED BY
THE JOB CONTROL STATEMENTS

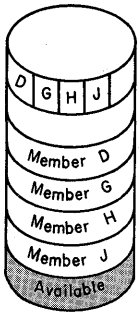
DATASET1
(defined by
INOUT1)



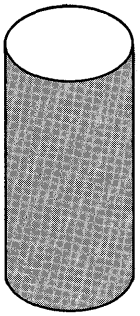
DATASET2
(defined by
INOUT2)



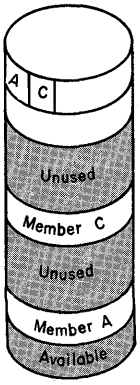
DATASET3
(defined by
INOUT3)



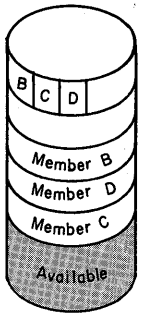
DATASET4
(defined by
INOUT4)



DATASET5
(defined by
INOUT5)



DATASET6
(defined by
INOUT6)



IEBCOPY Example 1

Operation	Data Set Organization	Input Device	Output Device	Comments
COPY	Input-PARTITIONED Output-PARTITIONED	DISK - 2311	DISK - 2311	1. Full copy

In this example, a partitioned data set (DATASET5) is to be copied from one disk storage volume to another.

- The INOUT4 DD Statement: 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 2311 Disk Storage Volume. Two blocks are allocated for directory entries.

- The INOUT5 DD Statement: defines a partitioned data set (DATASET5). It resides on a 2311 Disk Storage Volume and contains two members (A and C).

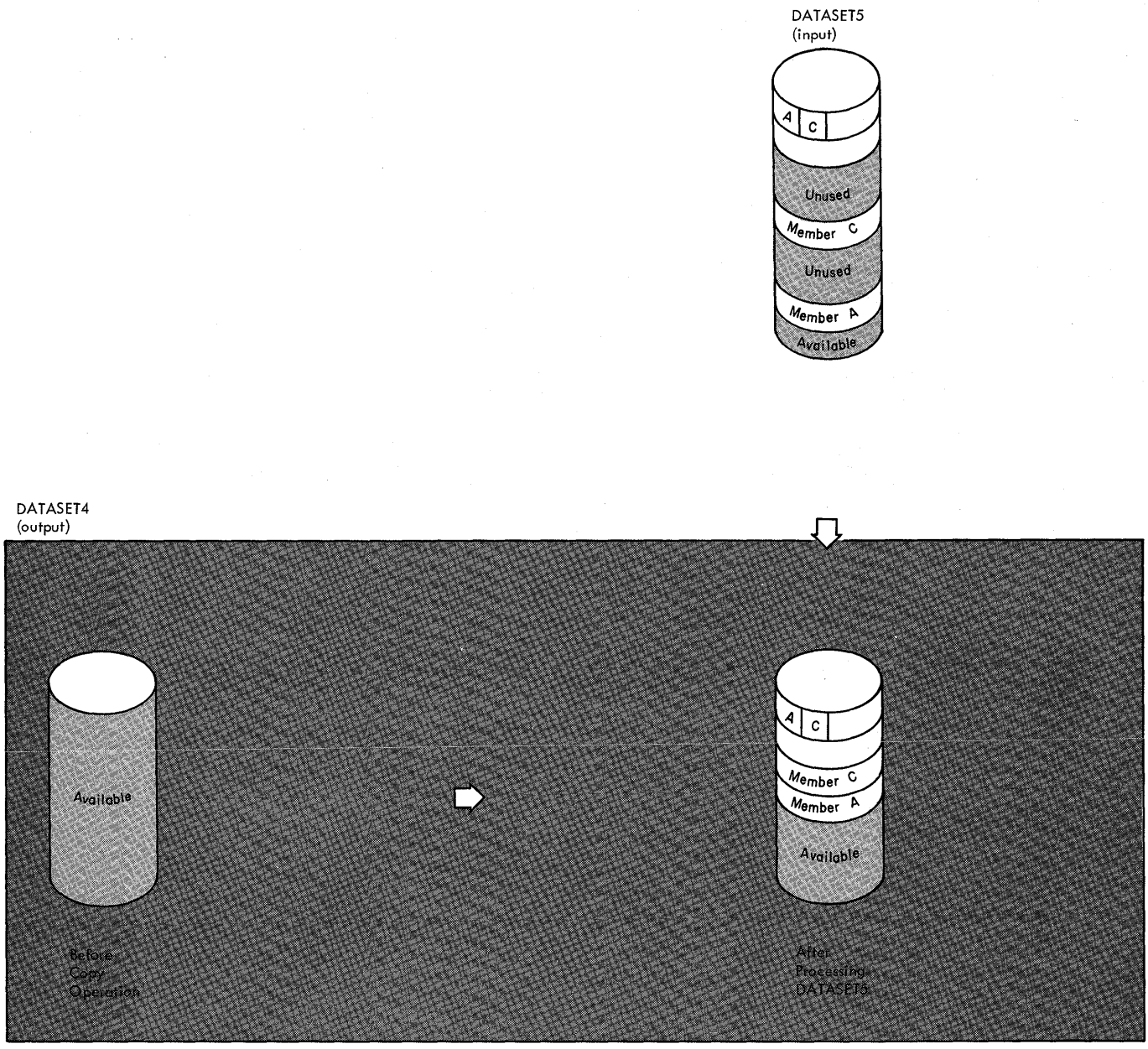
- The SYSUT3 DD Statement: defines the temporary spill data set to be used if there is not enough space in main storage for the input partitioned data set's directory entries. One track is allocated on a 2311 Disk Storage Volume. This data set may or may not be opened, depending on the amount of main storage available; therefore, it is suggested that the statement always appear in the job stream.

- The SYSUT4 DD Statement: 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. One track is allocated on a 2311 Disk Storage Volume. 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.

- The SYSIN DD Statement: defines the control data set, which follows in the input stream. The data set contains a COPY utility control statement.

- The COPY Statement: indicates the start of the copy operation. The absence of a SELECT or EXCLUDE statement causes a default to a full copy. The OUTDD operand specifies INOUT4 as the DD statement for the output data set (DATASET4); the INDD operand 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.

```
//COPY      JOB   06#990,MCEWAN
//JOBSTEP   EXEC  PGM=IEBCOPY
//SYSPRINT  DD    SYSOUT=A
//INOUT4    DD    DSNAME=DATASET4,UNIT=2311,VOL=SER=111112,                    X
//                DISP=(NEW,KEEP),SPACE=(TRK,(5,1,2))
//INOUT5    DD    DSNAME=DATASET5,UNIT=2311,VOL=SER=111113,                    X
//                DISP=OLD
//SYSUT3    DD    DSNAME=TEMP1,UNIT=2311,VOL=SER=111118,                       X
//                DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSUT4    DD    DSNAME=TEMP2,UNIT=2311,VOL=SER=111119,                       X
//                DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSIN     DD    *
COPYOPER    COPY  OUTDD=INOUT4,INDD=INOUT5
/*
```



IEBCOPY Example 1. Copying A Partitioned Data Set -- Full Copy

IEBCOPY Example 2

Operation	Data Set Organization	Input Device	Output Device	Comments
COPY	Input-PARTITIONED Output-PARTITIONED	DISK - 2311 DRUM - 2301 DRUM - 2301	DISK - 2302	1. Multiple input partitioned data sets. 2. Record formats: fixed blocked and fixed.

In this example, members are to be copied from three input partitioned data sets (DATASET1, DATASET5, and DATASET6) onto an existing output partitioned data set (DATASET2).

- The INOUT1 DD Statement: defines a partitioned data set (DATASET1). This data set contains three members (A, B, and F) in fixed format with a logical record length of 80 bytes and a blocksize of 80 bytes. This data set resides on a 2311 Disk Storage Volume.

- The INOUT5 DD Statement: defines a partitioned data set (DATASET5) which resides on a 2301 Drum Storage Unit. This data set contains two members (A and C) in fixed blocked format with a logical record length of 80 bytes and a blocksize of 160 bytes.

- The INOUT2 DD Statement: defines a partitioned data set (DATASET2) which resides on a 2302 Disk Storage Unit. 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 blocksize of 240 bytes.

- The INOUT6 DD Statement: defines a partitioned data set (DATASET6) which resides on a 2301 Drum Storage Unit. This data set contains three members (B, C, and D) in fixed blocked format with a logical record length of 80 bytes and a blocksize of 400 bytes. This data set is to be deleted when processing is completed.

- The SYSUT3 DD Statement: defines the temporary spill data set to be used if there is not enough space in main storage for the input partitioned data set's directory entries. One track is allocated on a 2311 Disk Storage Volume. 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.

- The SYSUT4 DD Statement: 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. One track is allocated on a 2311 Disk Storage Volume. 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.

- The SYSIN DD Statement: defines the control data set, which follows in the input stream. The data set contains a COPY utility control statement and three INDD statements.

- The COPY Statement: indicates the start of the copy operation. The absence of a SELECT or EXCLUDE statement causes a default to a full copy. The OUTDD operand 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. Processing occurs as follows:
  1. All members (A, B, and F) are copied onto the output data set (DATASET2).

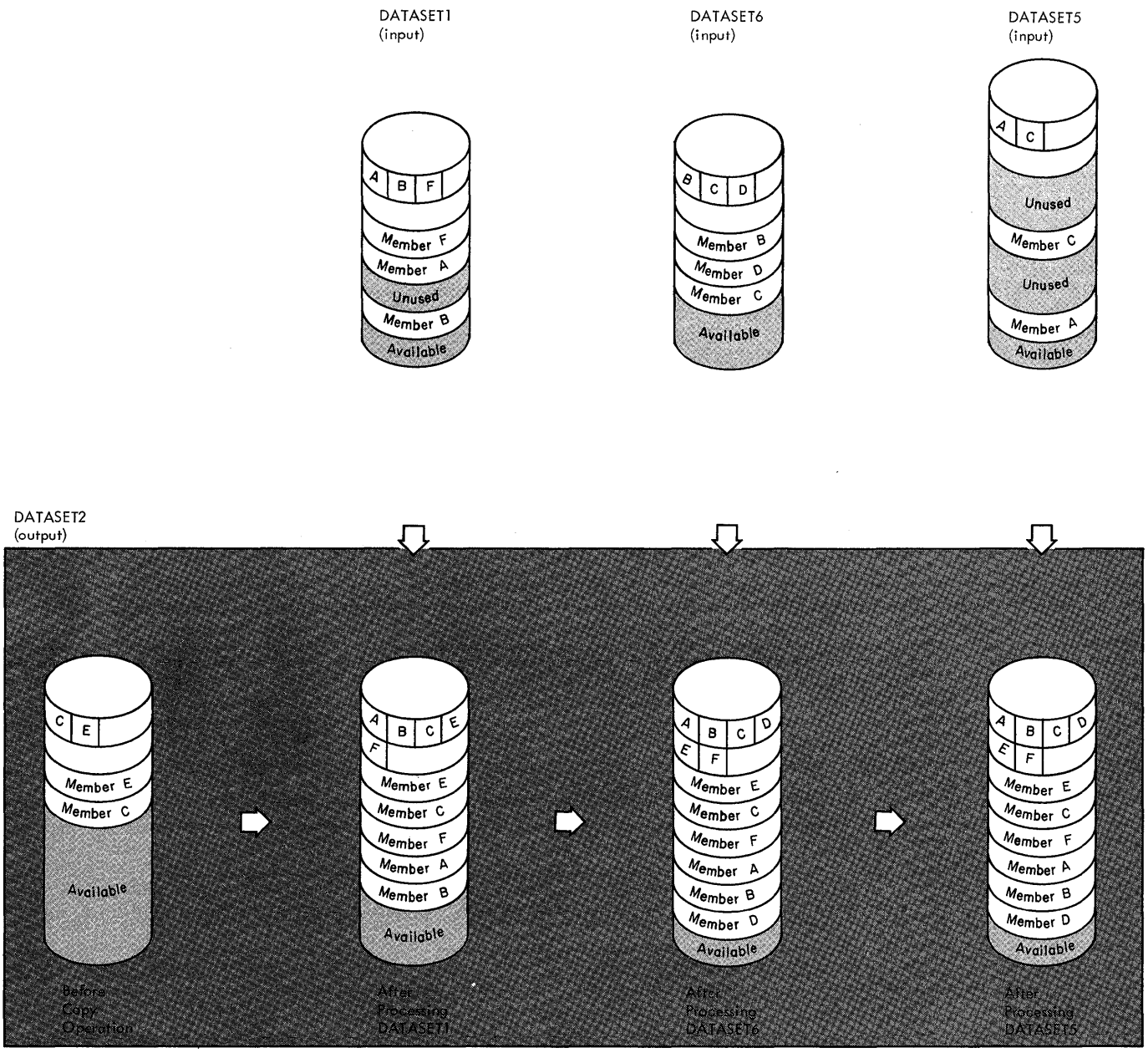
- The second INDD Statement: specifies INOUT6 as the DD statement for the second input data set (DATASET6) to be processed. Processing occurs as follows:

  1. Member B is not copied onto the output data set (DATASET2). It already exists on DATASET2.
  2. Member C is not copied onto the output data set (DATASET2). It already exists on DATASET2.
  3. Member D is copied onto the output data set (DATASET2).
  4. 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. Processing occurs as follows:

  1. No members are copied onto the output data set (DATASET2). All of them exist on DATASET2.

```
//COPY      JOB   06#990,MCEWAN
//JOBSTEP   EXEC  PGM=IEBCOPY
//SYSPRINT  DD    SYSOUT=A
//INOUT1    DD    DSNAME=DATASET1,UNIT=2311,VOL=SER=111112,                  X
//              DISP=(OLD,KEEP)
//INOUT5    DD    DSNAME=DATASET5,UNIT=2301,VOL=SER=111114,                  X
//              DISP=OLD
//INOUT2    DD    DSNAME=DATASET2,UNIT=2302,VOL=SER=111115,                  X
//              DISP=(OLD,KEEP)
//INOUT6    DD    DSNAME=DATASET6,UNIT=2301,VOL=SER=111117,                  X
//              DISP=(OLD,DELETE)
//SYSUT3    DD    DSNAME=TEMP1,UNIT=2311,VOL=SER=111118,                     X
//              DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSUT4    DD    DSNAME=TEMP2,UNIT=2311,VOL=SER=111119,                     X
//              DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSIN     DD    *
COPYOPER    COPY  OUTDD=INOUT2
                  INDD=INOUT1
                  INDD=INOUT6
                  INDD=INOUT5
/*
```



IEBCOPY Example 2. Copying From Three Input Partitioned Data Sets

IEBCOPY Example 3

Operation	Data Set Organization	Input Device	Output Device	Comments
COPY	Input-PARTITIONED Output-PARTITIONED	DISK - 2302	DISK - 2314	1. All members on the input partitioned data set are copied. Identically named members on the output partitioned data set are replaced.

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

- The INOUT2 DD Statement: defines a partitioned data set (DATASET2) which resides on a 2314 Direct Access Storage Volume. This data set contains two members (C and E).

- The INOUT6 DD Statement: defines a partitioned data set (DATASET6). This data set resides on a 2302 Disk Storage Unit and contains three members (B, C, and D).

- The SYSUT3 DD Statement: defines the temporary spill data set to be used if there is not enough space in main storage for the input partitioned data set's directory entries. One track is allocated on a 2311 Disk Storage Volume. 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.

- The SYSUT4 DD Statement: 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. One track is allocated on a 2311 Disk Storage Volume. 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.

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

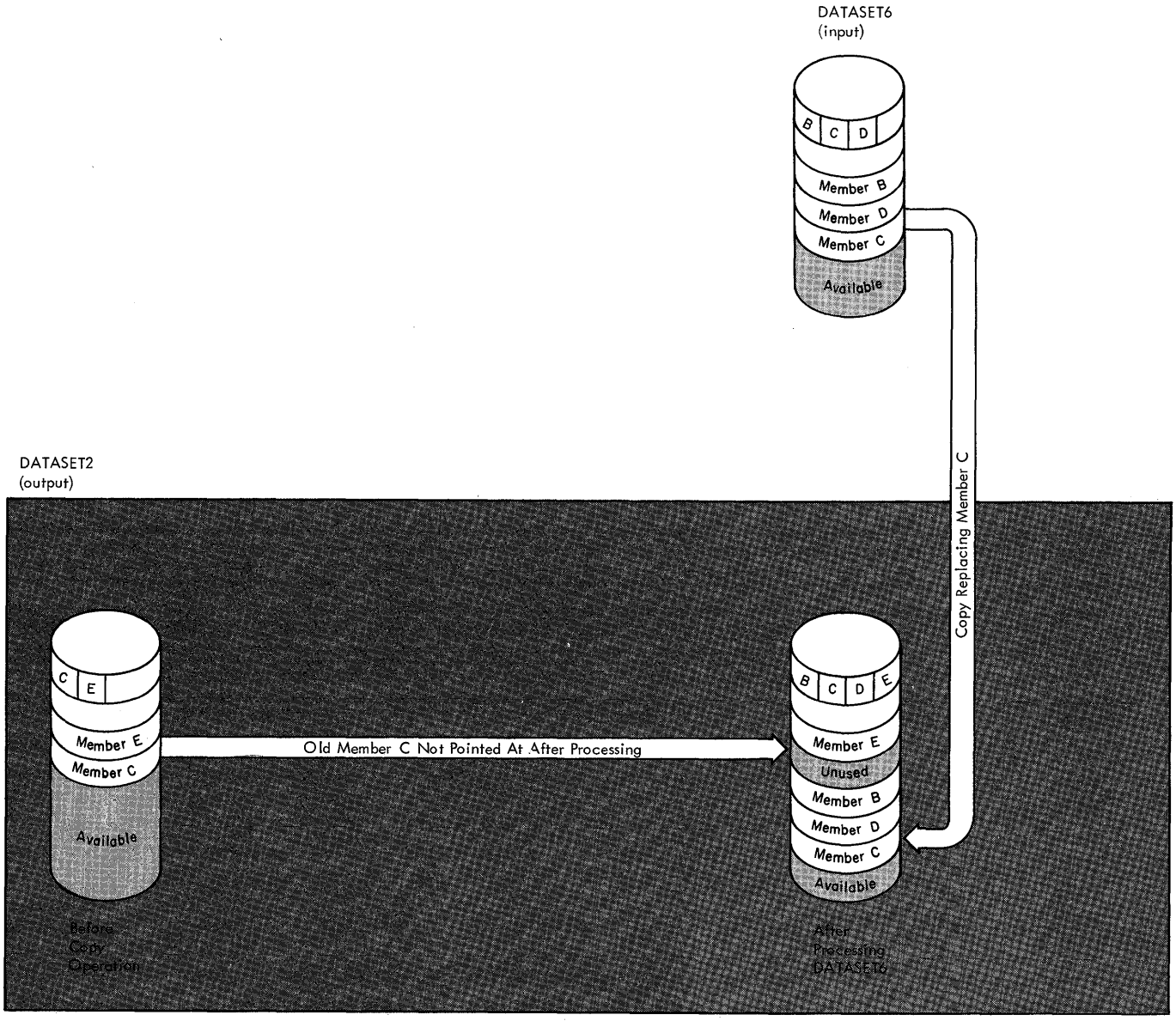
- The COPY Statement: indicates the start of the copy operation. The absence of a SELECT or EXCLUDE statement causes a default to a full copy. The OUTDD operand specifies INOUT2 as the DD statement for the output data set (DATASET2).

- The INDD Statement: specifies INOUT6 as the DD statement for the input data set (DATASET6). Processing occurs as follows:

  1. Member B is copied onto the output data set (DATASET2).
  2. Member C is copied onto the output data set (DATASET2). This member is copied despite the fact that the output data set already contains a member named 'C' because the replace option is specified for all identically named members on the input data set; i.e., the replace option is specified on the INDD level.
  3. Member D is copied onto the output data set (DATASET2).

  (Note: 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.)

```
//COPY      JOB   06#990,MCEWAN
//JOBSTEP   EXEC  PGM=IEBCOPY
//SYSPRINT  DD    SYSOUT=A
//INOUT2    DD    DSNAME=DATASET2,UNIT=2314,VOL=SER=111113,                X
//              DISP=OLD
//INOUT6    DD    DSNAME=DATASET6,UNIT=2302,VOL=SER=111117,                X
//              DISP=(OLD,KEEP)
//SYSUT3    DD    DSNAME=TEMP1,UNIT=2311,VOL=SER=111118,                   X
//              DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSUT4    DD    DSNAME=TEMP2,UNIT=2311,VOL=SER=111119,                   X
//              DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSIN     DD    *
COPYOPER    COPY  OUTDD=INOUT2
                  INDD=((INOUT6,R))
/*
```



IEBCOPY Example 3. Copy Operation With 'Replace' Specified on INDD Level

IEBCOPY Example 4

Operation	Data Set Organization	Input Device	Output Device	Comments
COPY	Input-PARTITIONED Output-PARTITIONED	DISK - 2302 DISK - 2314	DISK - 2311	1. Only selected members are copied. 2. Creating a variable blocked data set. 3. Record formats: variable blocked and variable.

In this example, five members (A, C, D, E, and G) are to be selected from two input partitioned data sets (DATASET6 and DATASET3) to be copied onto a new output partitioned data set (DATASET4).

- The INOUT2 DD Statement: defines a partitioned data set (DATASET2) which resides on a 2314 Direct Access Storage Volume. This data set contains two members (C and E) in variable blocked format with a logical record length of 96 bytes and a blocksize of 500 bytes. This data set is to be deleted when processing is completed.

- The INOUT6 DD Statement: defines a partitioned data set (DATASET6) which resides on a 2302 Disk Storage Unit. This data set contains three members (B, C, and D) in variable format with a logical record length of 96 bytes and a blocksize of 100 bytes.

- The INOUT4 DD Statement: 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 2311 Disk Storage Volume. Two blocks are allocated for directory entries. In addition, records are to be copied onto this data set in variable blocked format with a logical record length of 96 bytes and a blocksize of 300 bytes.

- The SYSUT3 DD Statement: defines the temporary spill data set to be used if there is not enough space in main storage for the input partitioned data set's directory entries. One track is allocated on a 2311 Disk Storage Volume. 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.

- The SYSUT4 DD Statement: 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. One track is allocated on a 2311 Disk Storage Volume. 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.

- The SYSIN DD Statement: defines the control data set, which follows in the input stream. The data set contains a COPY utility control statement, two INDD statements, and a SELECT utility control statement.

- The COPY Statement: indicates the start of the copy operation. The presence of a SELECT statement causes a selective copy. The OUTDD operand specifies INOUT4 as the DD statement for the output data set (DATASET4).

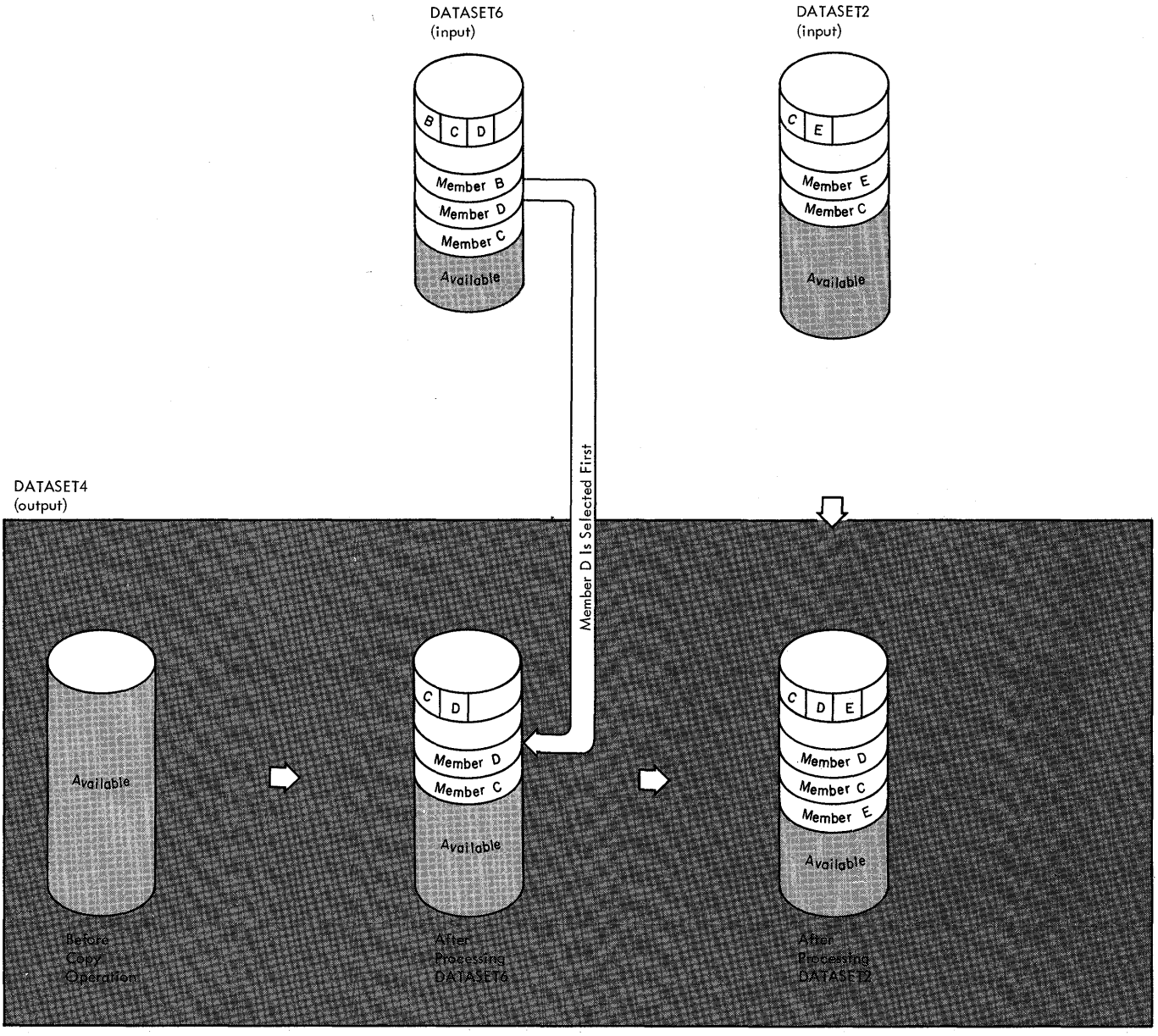
- The first INDD Statement: specifies INOUT6 as the DD statement for the first input data set (DATASET6) to be processed. Processing occurs as follows:
  1. Member A is searched for and not found.
  2. Member C is searched for and found. It is not searched for again.
  3. Member D is searched for and found. It is not searched for again.
  4. Member E is searched for and not found.
  5. Member G is searched for and not found.

6. Found members (C and D) are copied onto the output data set (DATASET4) in TTR order; i.e., member 'D' is copied first, then member 'C'

- The second INDD Statement: specifies INOUT2 as the DD statement for the second input data set (DATASET2) to be processed. Processing occurs as follows:
  1. Member A is searched for and not found.
  2. Member E is searched for and found. It is not searched for again.
  3. Member G is searched for and not found.
  4. Found members (E) are copied onto the output data set (DATASET4) in TTR order.
  5. All members on DATASET2 are lost when the data set is deleted.

- The SELECT Statement: specifies the members to be selected from the input data sets (DATASET6 and DATASET2) to be copied onto the output data set (DATASET4). Selected members not found on any of the input data sets are not copied in this copy operation.

```
//COPY     JOB      06#990,MCEWAN
//JOBSTEP  EXEC     PGM=IEBCOPY
//SYSPRINT DD       SYSOUT=A
//INOUT2   DD       DSNAME=DATASET2,UNIT=2314,VOL=SER=111114,                X
//                 DISP=(OLD,DELETE)
//INOUT6   DD       DSNAME=DATASET6,UNIT=2302,VOL=SER=111117,                X
//                 DISP=(OLD,KEEP)
//INOUT4   DD       DSNAME=DATASET4,UNIT=2311,VOL=SER=111116,                X
//                 DISP=(NEW,KEEP),SPACE=(TRK,(5,,2)),                      X
//                 DCB=(RECFM=VB,LRECL=96,BLKSIZE=300)
//SYSUT3   DD       DSNAME=TEMP1,UNIT=2311,VOL=SER=111118,                   X
//                 DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSUT4   DD       DSNAME=TEMP2,UNIT=2311,VOL=SER=111119,                   X
//                 DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSIN    DD       *
COPYOPER  COPY      OUTDD=INOUT4
                    INDD=INOUT6
                    INDD=INOUT2
          SELECT    MEMBER=C,D,E,A,G
/*
```



IEBCOPY Example 4. Copying Selected Members (With Reblocking and Deblocking)

IEBCOPY Example 5

Operation	Data Set Organization	Input Device	Output Device	Comments
COPY	Input-PARTITIONED Output-PARTITIONED	DISK - 2302 DISK - 2311	DISK - 2311	1. Selective copy. 2. One member is to replace an identically named member on the output data set.

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

- The INOUT1 DD Statement: defines a partitioned data set (DATASET1). This data set resides on a 2311 Disk Storage Volume and contains three members (A, B, and F).

- The INOUT6 DD Statement: defines a partitioned data set (DATASET6) which resides on a 2302 Disk Storage Unit. This data set contains three members (B, C, and D).

- The INOUT5 DD Statement: defines a partitioned data set (DATASET5). This data set resides on a 2311 Disk Storage Volume and contains two members (A and C).

- The SYSUT3 DD Statement: defines the temporary spill data set to be used if there is not enough space in main storage for the input partitioned data set's directory entries. One track is allocated on a 2311 Disk Storage Volume. 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.

- The SYSUT4 DD Statement: 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. One track is allocated on a 2311 Disk Storage Volume. 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.

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

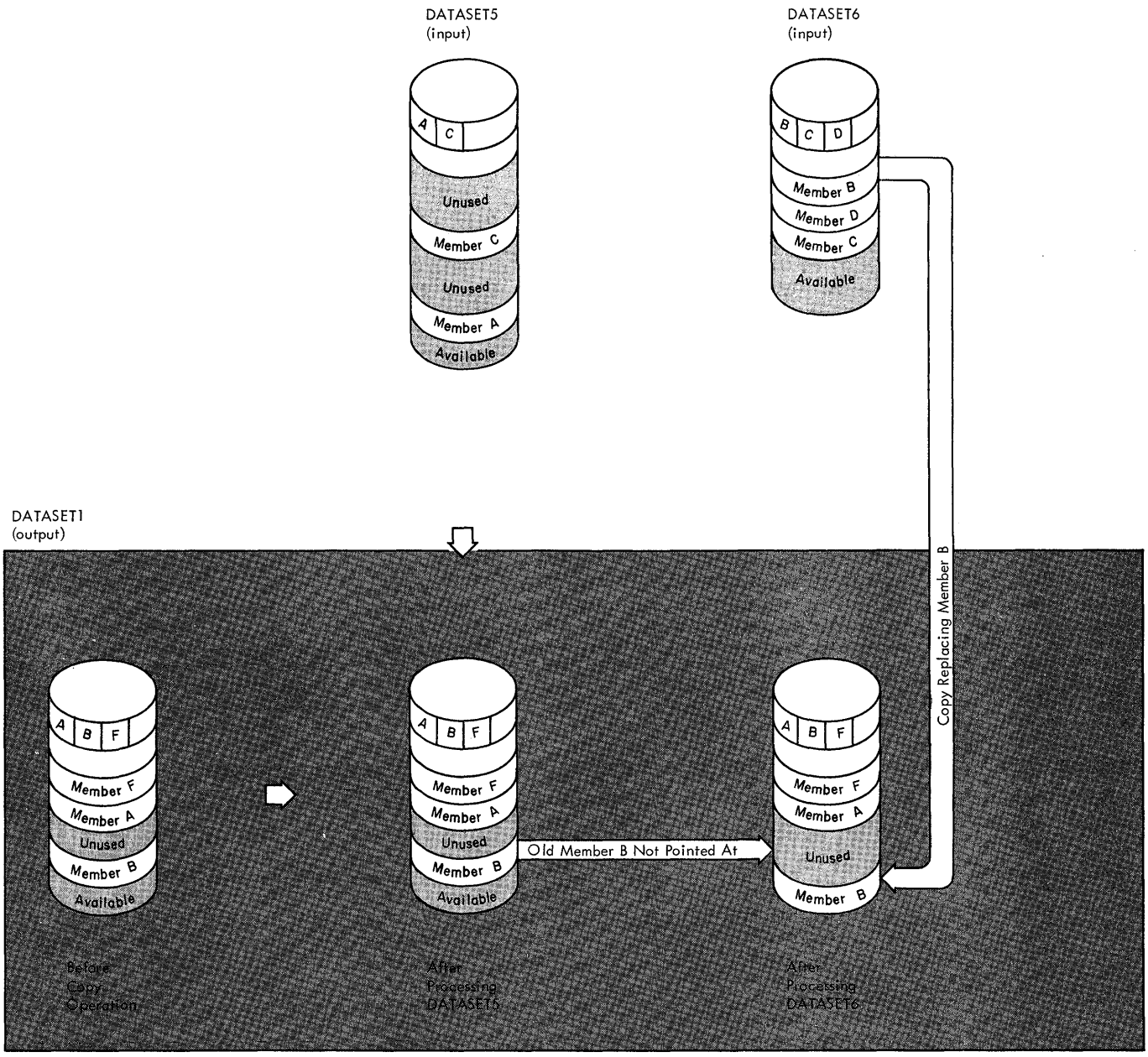
- The COPY Statement: indicates the start of the copy operation. The presence of a SELECT statement causes a selective copy. The OUTDD operand specifies INOUT1 as the DD statement for the output data set (DATASET1).

- The INDD Statement: 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. Member A is searched for on DATASET5 and found. It is not searched for again.
  2. Member B is searched for on DATASET5 and not found.
  3. Member A is not copied onto the output data set (DATASET1). It already exists on DATASET2 and the replace option is not specified on either the INDD level or on the MEMBER level.
  4. Member B is searched for on DATASET6 and found. It is not searched for again.
  5. Member B is copied onto 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.

(Note: 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.)

- The SELECT Statement: specifies the members to be selected from the input data sets (DATASET5 and DATASET6) to be copied onto the output data set (DATASET1). Both of the selected members are found, but only one is copied.

```
//COPY      JOB     06#990,MCEWAN
//JOBSTEP   EXEC    PGM=IEBCOPY
//SYSPRINT  DD      SYSOUT=A
//INOUT1    DD      DSNAME=DATASET1,UNIT=2311,VOL=SER=111112,                X
//                 DISP=(OLD,KEEP)
//INOUT6    DD      DSNAME=DATASET6,UNIT=2302,VOL=SER=111115,                X
//                 DISP=OLD
//INOUT5    DD      DSNAME=DATASET5,UNIT=2311,VOL=SER=111116,                X
//                 DISP=(OLD,KEEP)
//SYSUT3    DD      DSNAME=TEMP1,UNIT=2311,VOL=SER=111118,                   X
//                 DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSUT4    DD      DSNAME=TEMP2,UNIT=2311,VOL=SER=111119,                   X
//                 DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSIN     DD      *
COPYOPER    COPY    OUTDD=INOUT1
                    INDD=INOUT5,INOUT6
            SELECT  MEMBER=((B,,R),A)
/*
```



IEBCOPY Example 5. Selective Copy With 'Replace' Specified On MEMBER Level

IEBCOPY Example 6

Operation	Data Set Organization	Input Device	Output Device	Comments
COPY	Input-PARTITIONED Output-PARTITIONED	DRUM - 2301 DISK - 2314	DISK - 2311	1. Selective copy. 2. All members found on first input data set replace identically named members on output data set.

In this example, two members (A and B) are to be selected from two input partitioned data sets (DATASET5 and DATASET6) to be copied onto an existing output partitioned data set (DATASET1). All members found on DATASET5 are to replace identically named members on DATASET1.

- The INOUT1 DD Statement: defines a partitioned data set (DATASET1). This data set resides on a 2311 Disk Storage Volume and contains three members (A, B, and F).

- The INPUT5 DD Statement: defines a partitioned data set (DATASET5). This data set contains two members (A and C) and resides on a 2314 Direct Access Storage Volume. This data set is to be deleted when processing is completed.

- The INOUT6 DD Statement: defines a partitioned data set (DATASET6). This data set contains three members (B, C, and D) and resides on a 2301 Drum Storage Unit.

- The SYSUT3 DD Statement: defines the temporary spill data set to be used if there is not enough space in main storage for the input partitioned data set's directory entries. One track is allocated on a 2311 Disk Storage Volume. 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.

- The SYSUT4 DD Statement: 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. One track is allocated on a 2311 Disk Storage Volume. 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.

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

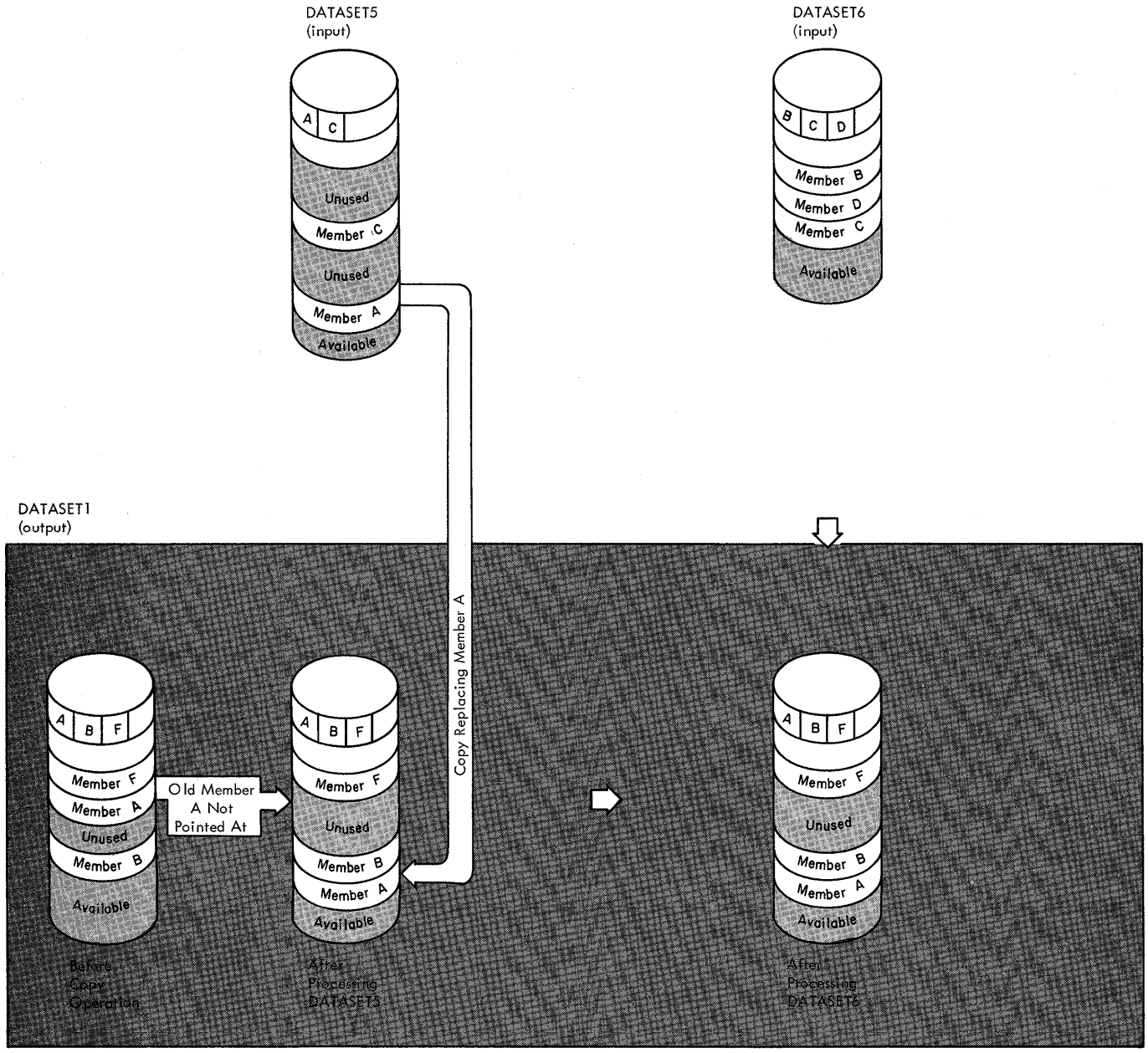
- The COPY Statement: indicates the start of the copy operation. The presence of a SELECT statement causes a selective copy. The OUTDD operand specifies INOUT1 as the DD statement for the output data set (DATASET1).

- The INDD Statement: 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. Member A is searched for on DATASET5 and found. It is not searched for again.
  2. Member B is searched for on DATASET5 and not found.
  3. Member A is copied onto the output data set (DATASET1) because the replace option is specified on the INDD level for all members found on DATASET5. (Note: 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.)

4. Member B is searched for on DATASET6 and found. It is not copied because DATASET1 already contains a member called member 'B' and the replace option is not specified on either the INDD level or on the MEMBER level for this member.

- The SELECT Statement: specifies the members to be selected from the input data sets (DATASET5 and DATASET6) to be copied onto the output data set (DATASET1). Both selected members are found, but only one is copied.

```
//COPY      JOB     06#990,MCEWAN
//JOBSTEP   EXEC    PGM=IEBCOPY
//SYSPRINT  DD      SYSOUT=A
//INOUT1    DD      DSNAME=DATASET1,UNIT=2311,VOL=SER=111112,                X
//                 DISP=(OLD,KEEP)
//INOUT5    DD      DSNAME=DATASET5,UNIT=2314,VOL=SER=111114,                X
//                 DISP=(OLD,DELETE)
//INOUT6    DD      DSNAME=DATASET6,UNIT=2301,VOL=SER=111115,                X
//                 DISP=(OLD,KEEP)
//SYSUT3    DD      DSNAME=TEMP1,UNIT=2311,VOL=SER=111118,                   X
//                 DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSUT4    DD      DSNAME=TEMP2,UNIT=2311,VOL=SER=111119,                   X
//                 DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSIN     DD      *
COPYOPER    COPY    OUTDD=INOUT1
                    INDD=((INOUT5,R),INOUT6)
            SELECT  MEMBER=(A,B)
/*
```



IEBCOPY Example 6. Selective Copy With 'Replace' Specified On INDD Level - Multiple Inputs

IEBCOPY Example 7

Operation	Data Set Organization	Input Device	Output Device	Comments
COPY	Input-PARTITIONED Output-PARTITIONED	DISK - 2311	DISK - 2311	1. Selective copy. 2. Two members renamed. 3. One renamed member replaces identically named member on output data set.

In this example, four members (A, B, C, and D) are to be selected from an input partitioned data set (DATASET6) to be copied onto 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.

- The INOUT3 DD Statement: defines a partitioned data set (DATASET3). This data set contain four members (D, G, H, and J) and resides on a 2311 Disk Storage Volume.

- The INOUT6 DD Statement: defines a partitioned data set (DATASET6). This data set contains three members (B, C, and D) and resides on a 2311 Disk Storage Volume. DATASET6 is to be deleted when processing is completed; thus, all members on this data set are lost.

- The SYSUT3 DD Statement: defines the temporary spill data set to be used if there is not enough space in main storage for the input partitioned data set's directory entries. One track is allocated on a 2311 Disk Storage Volume. 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.

- The SYSUT4 DD Statement: 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. One track is allocated on a 2311 Disk Storage Volume. 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.

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

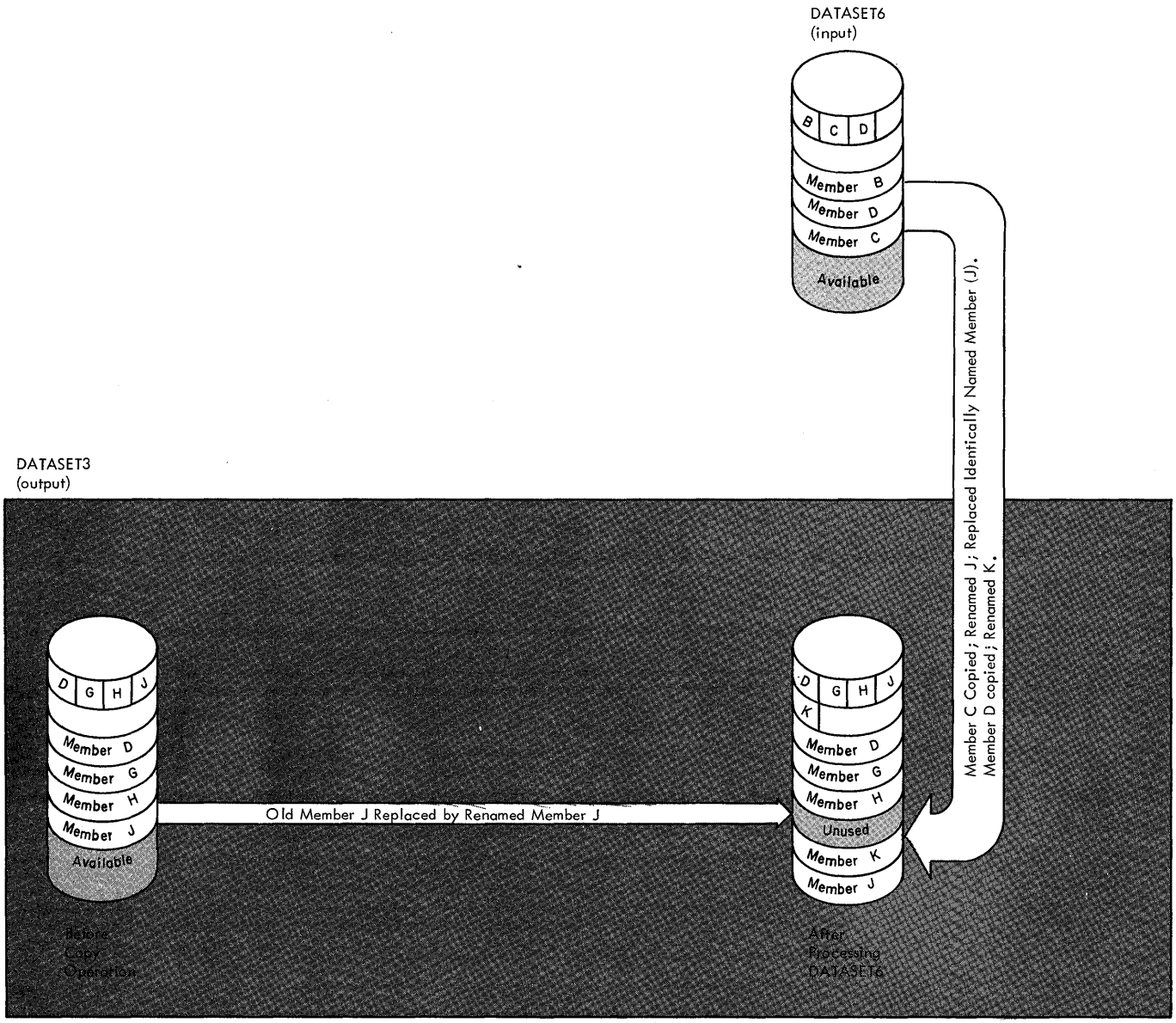
- The COPY Statement: indicates the start of the copy operation. The presence of a SELECT statement causes a selective copy. The OUTDD operand specifies INOUT3 as the DD statement for the output data set (DATASET3).

- The INDD Statement: specifies INOUT6 as the DD statement for the input data set (DATASET6). Processing occurs as follows:
  1. Member A is searched for and not found.
  2. Member B is searched for and found. It is not searched for again.
  3. Member C is searched for and found. It is not searched for again.
  4. Member D is searched for and found. It is not searched for again.
  5. Member B is not copied onto the output data set (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 on either the INDD or the MEMBER level.
  6. Member C is copied onto 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.

7. Member D is copied onto the output data set (DATASET3) because its new name (K) does not already exist there.

- The SELECT Statement: specifies the members to be selected from the input data set (DATASET6) to be copied onto the output data set (DATASET3). Member A is not found on the input data set; it is not copied or searched for again, in this copy operation.

```
//COPY      JOB      #990,MCEWAN
//JOBSTEP   EXEC     PGM=IEBCOPY
//SYSPRINT  DD       SYSOUT=A
//INOUT3    DD       DSNAME=DATASET3,UNIT=2311,VOL=SER==111114,                X
//                  DISP=(OLD,KEEP)
//INOUT6    DD       DSNAME=DATASET6,UNIT=2311,VOL=SER=111117,                 X
//                  DISP=(OLD,DELETE)
//SYSUT3    DD       DSNAME=TEMP1,UNIT=2311,VOL=SER=111118,                    X
//                  DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSUT4    DD       DSNAME=TEMP2,UNIT=2311,VOL=SER=111119,                    X
//                  DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSIN     DD       *
COPYOPER    COPY     OUTDD=INOUT3
                     INDD=INOUT6
            SELECT   MEMBER=((B,H),(C,J,R),A,(D,K))
/*
```



IEBCOPY Example 7. Renaming Selected Members; One Renamed Member Replaces Identically Named Member

IEBCOPY Example 8

Operation	Data Set Organization	Input Device	Output Device	Comments
COPY	Input-PARTITIONED Output-PARTITIONED	DISK - 2311 DISK - 2311 DISK - 2311	DISK - 2311	1. Exclusive copy. 2. Record formats: fixed blocked and fixed.

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).

- The INOUT1 DD Statement: defines a partitioned data set (DATASET1). This data set contains three members (A, B, and F) and resides on a 2311 Disk Storage Volume. The record format is fixed blocked with a logical record length of 100 bytes and a blocksize of 400 bytes.

- The INOUT3 DD Statement: defines a partitioned data set (DATASET3) which resides on a 2311 Disk Storage Volume. This data set contains four members (D, G, H, and J) in fixed blocked format with a logical record length 100 bytes and a blocksize of 600 bytes.

- The INOUT4 DD Statement: defines a new partitioned data set (DATASET4). Five tracks are allocated for the copied members on a 2311 Disk Storage Volume. Two blocks are allocated for directory entries. In addition records are to be copied onto this data set in fixed blocked format with a logical record length of 100 bytes and a blocksize of 400 bytes.

- The INOUT6 DD Statement: 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 blocksize of 100 bytes. This data set resides on a 2311 Disk Storage Volume.

- The SYSUT3 DD Statement: defines the temporary spill data set to be used if there is not enough space in main storage for the input partitioned data set's directory entries. One track is allocated on a 2311 Disk Storage Volume. 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.

- The SYSUT4 DD Statement: 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. One track is allocated on a 2311 Disk Storage Volume. 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.

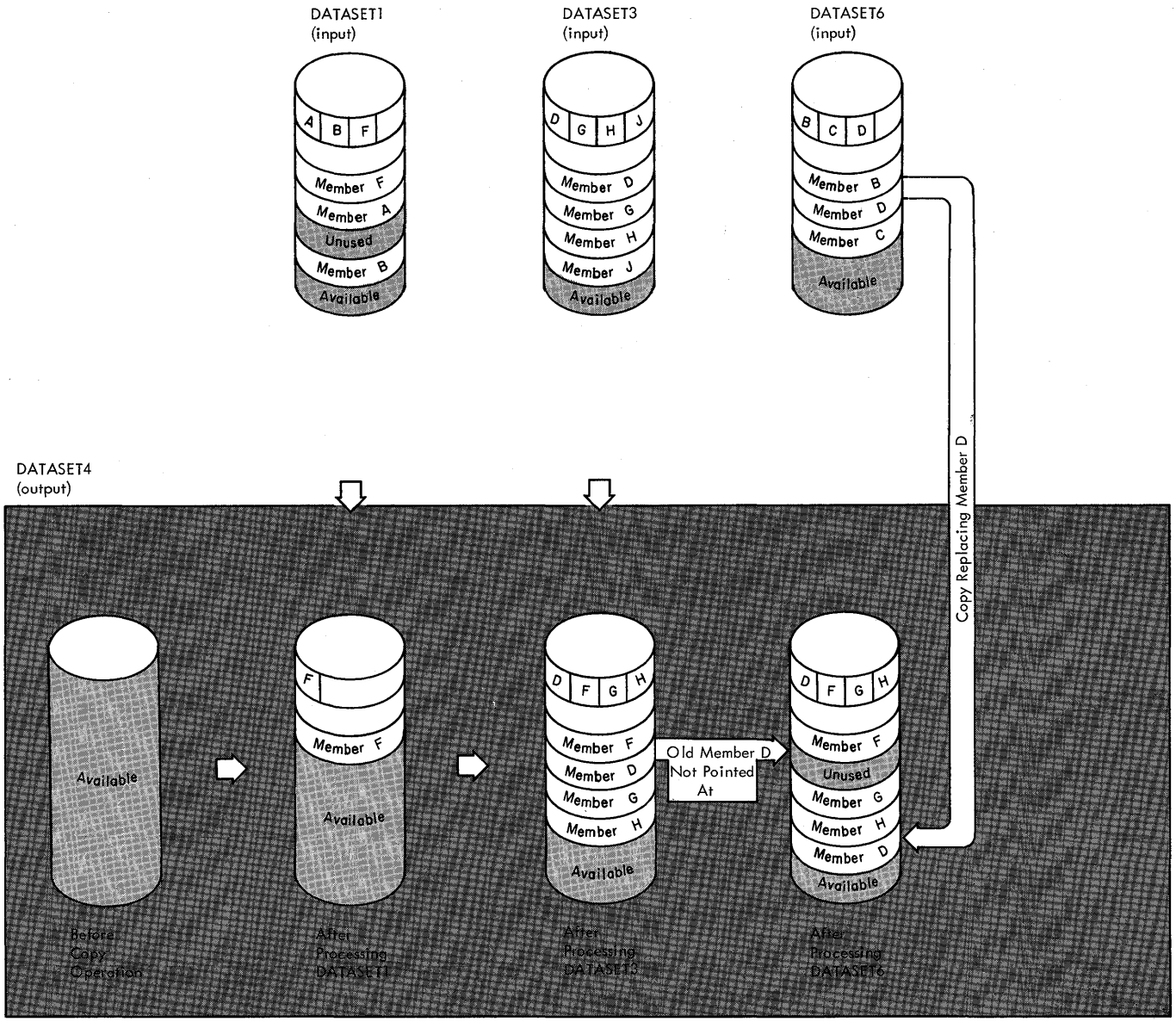
- The SYSIN DD Statement: defines the control data set, which follows in the input stream. The data set contains a COPY utility control statement and an EXCLUDE utility control statement.

- The COPY Statement: indicates the start of the copy operation. The presence of an EXCLUDE statement causes an exclusive copy. The OUTDD operand specifies INOUT4 as the DD statement for the output data set (DATASET4). The INDD operand 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. Only Member F is copied from DATASET1; all other member are specified on the EXCLUDE statement.
2. Members D, G, and H are copied from DATASET3; Member J is not copied because 'J' is one of the members named on the EXCLUDE statement.
3. Member D is copied from DATASET6 because the replace option is specified for all members found on this data set that are not specified on the EXCLUDE statement.
   (Note: 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.)

- The EXCLUDE Statement: 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.

```
//COPY      JOB       06#990,MCEWAN
//JOBSTEP   EXEC      PGM=IEBCOPY
//SYSPRINT  DD        SYSOUT=A
//INOUT1    DD        DSNAME=DATASET1,UNIT=2311,VOL=SER=111112,                    X
//                   DISP=(OLD,KEEP)
//INOUT3    DD        DSNAME=DATASET3,UNIT=2311,VOL=SER=111114,                    X
//                   DISP=OLD
//INOUT4    DD        DSNAME=DATASET4,UNIT=2311,VOL=SER=111115,                    X
//                   DISP=(NEW,KEEP),SPACE=(TRK,(5,1,2)),                         X
//                   DCB=(LRECL=100,RECFM=FB,BLKSIZE=400)
//INOUT6    DD        DSNAME=DATASET6,UNIT=2311,VOL=SER=111116,                    X
//                   DISP=OLD
//SYSUT3    DD        DSNAME=TEMP1,UNIT=2311,VOL=SER=111118,                       X
//                   DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSUT4    DD        DSNAME=TEMP2,UNIT=2311,VOL=SER=111119,                       X
//                   DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSIN     DD        *
COPYOPER    COPY      OUTDD=INOUT4,INDD=INOUT1,INOUT3,(INOUT6,R)
            EXCLUDE   MEMBER=A,J,B,L,C
/*
```



IEBCOPY Example 8. Exclusive Copy With Replace Specified For One Input Partitioned Data Set

IEBCOPY Example 9

Operation	Data Set Organization	Input Device	Output Device	Comments
COPY	Input-PARTITIONED Output-PARTITIONED	DISK - 2314	DISK - 2314	1. Compress in place.

In this example, a partitioned data set (DATSET5) is to be compressed in place.

- The INOUT5 DD Statement: defines a partitioned data set (DATASET5). This data set contains two members (A and C) and resides on a 2314 Direct Access Storage Volume.

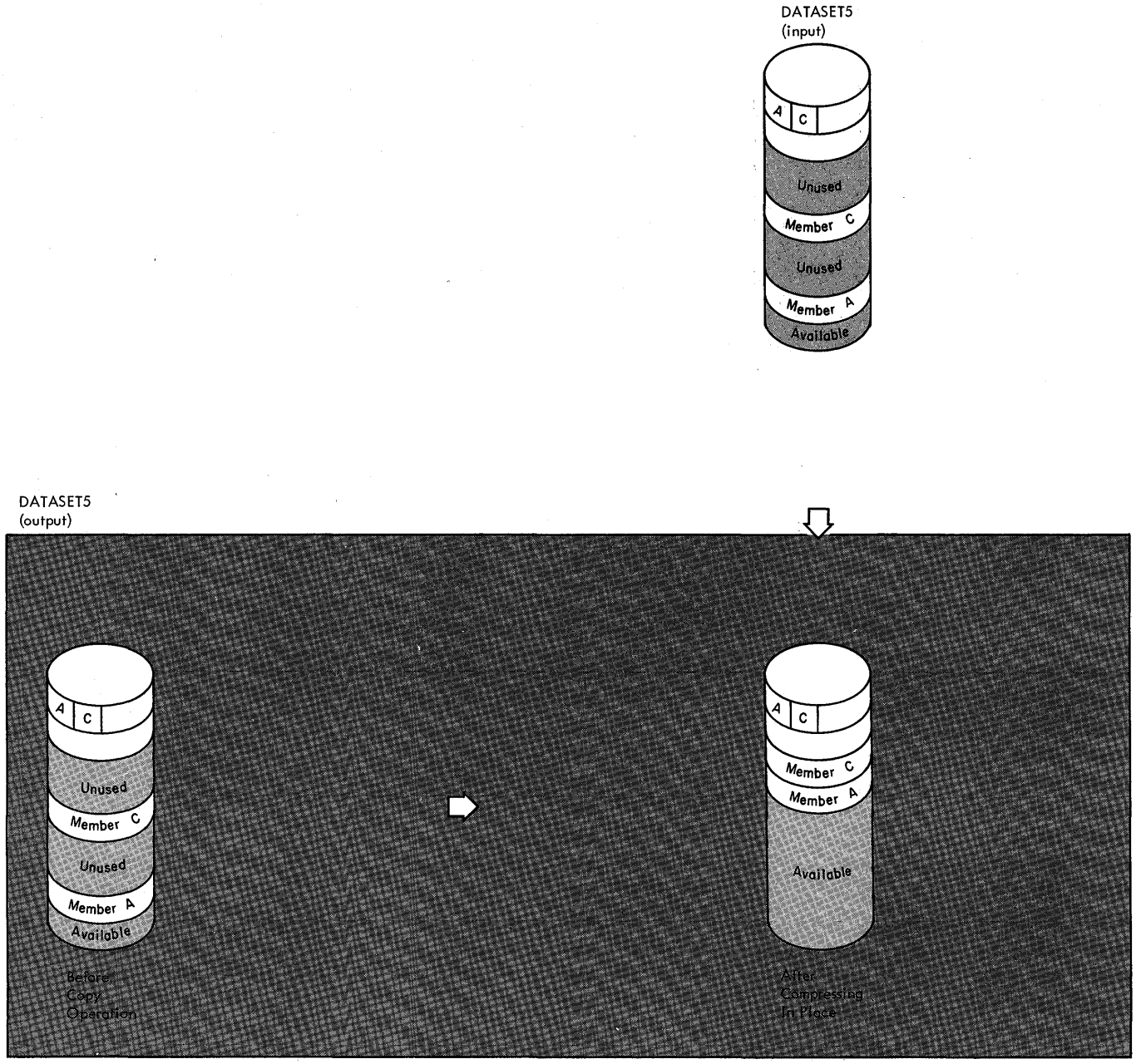
- The SYSUT3 DD Statement: defines the temporary spill data set to be used if there is not enough space in main storage for the input partitioned data set's directory entries. One track is allocated on a 2311 Disk Storage Volume. This data set may or may not be opened, depending on the amount of main storage available; therefore, it is suggested that the statement always appear in the job stream.

- The SYSUT4 DD Statement: 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. One track is allocated on a 2311 Disk Storage Volume. 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.

- The SYSIN DD Statement: defines the control data set, which follows in the input stream. The data set contains a COPY utility control statement.

- The COPY Statement: indicates the start of the copy operation. The absence of a SELECT or EXCLUDE statement causes a default to a full copy; however, the same DD statement is specified for both the INDD and OUTDD operands, thus causing a compress in place of the specified data set. The OUTDD operand specifies INOUT5 as the DD statement for the output data set (DATASET5). The INDD operand also specifies INOUT5 as the DD statement for the input data set (DATASET5).

```
//COPY     JOB   06#990,MCEWAN
//JOBSTEP  EXEC  PGM=IEBCOPY
//SYSPRINT DD    SYSOUT=A
//INOUT5   DD    DSNAME=DATASET5,UNIT=2314,VOL=SER=111113,                  X
//              DISP=(OLD,KEEP)
//SYSUT3   DD    DSNAME=TEMP1,UNIT=2311,VOL=SER=111118,                     X
//              DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSUT4   DD    DSNAME=TEMP2,UNIT=2311,VOL=SER=111119,                     X
//              DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSIN    DD    *
COPYOPER  COPY  OUTDD=INOUT5,INDD=INOUT5
/*
```



IEBCOPY Example 9. Compressing a Data Set in Place

IEBCOPY Example 10

Operation	Data Set Organization	Input Device	Output Device	Comments
COPY	Input-PARTITIONED Output-PARTITIONED	DISK - 2311 DISK - 2311	DISK - 2314	1. Full copy followed by a compress in place of the output data set. 2. Replace option specified for one input data set.

In this example, two input partitioned data sets (DATASET5 and DATASET6) are to be copied onto an existing output partitioned data set (DATASET1). In addition, all members on DATASET6 are to be copied, with members on the output data set which have the same names as the copied members being replaced. After DATASET6 is processed, the output data set (DATASET1) is to be compressed in place.

- The INOUT1 DD Statement: defines a partitioned data set (DATASET1). This data set contains three members (A, B, and F) and resides on a 2314 Direct Access Storage Volume.

- The INOUT5 DD Statement: defines a partitioned data set (DATASET5). This data set contains two members (A and C) and resides on a 2311 Disk Storage Volume.

- The INOUT6 DD Statement: defines a partitioned data set (DATASET6). This data set contains three members (B, C, and D) and resides on a 2311 Disk Storage Volume.

- The SYSUT3 DD Statement: defines the temporary spill data set to be used if there is not enough space in main storage for the input partitioned data set's directory entries. One track is allocated on a 2311 Disk Storage Volume. 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.

- The SYSUT4 DD Statement: 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. One track is allocated on a 2311 Disk Storage Volume. 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.

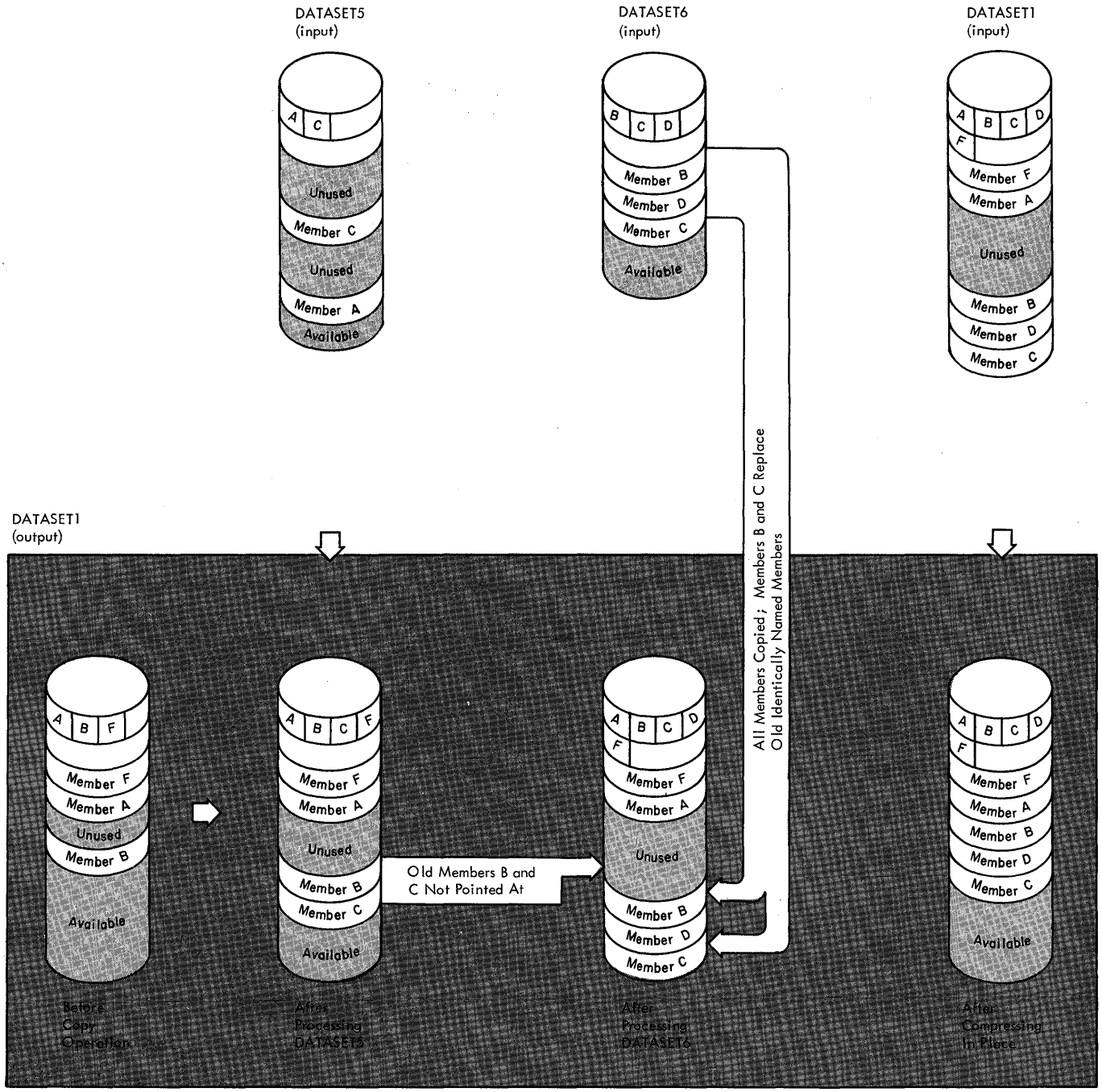
- The SYSIN DD Statement: defines the control data set, which follows in the input stream. The data set contains a COPY utility control statement and an INDD statement.

- The COPY Statement: 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.

- The INDD Statement: 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.
  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 onto the output data set (DATASET1), occupying the first available space.
3. All members are copied from DATASET6 onto the output data set (DATASET1), immediately following the last member. Members B and C are copied despite the fact that the output data set already contains members with the same names because the replace option is specified on the INDD level. (Note: 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.)
4. The members currently on DATASET1 are compressed in place, thereby eliminating embedded unused space.

```
//COPY      JOB   06#990,MCEWAN
//JOBSTEP   EXEC  PGM=IEBCOPY
//SYSPRINT  DD    SYSOUT=A
//INOUT1    DD    DSNAME=DATASET1,UNIT=2314,VOL=SER=111112,                  X
//                DISP=(OLD,KEEP)
//INOUT5    DD    DSNAME=DATASET5,UNIT=2311,VOL=SER=111114,                  X
//                DISP=OLD
//INOUT6    DD    DSNAME=DATASET6,UNIT=2311,VOL=SER=111115,                  X
//                DISP=(OLD,KEEP)
//SYSUT3    DD    DSNAME=TEMP1,UNIT=2311,VOL=SER=111118,                     X
//                DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSUT4    DD    DSNAME=TEMP2,UNIT=2311,VOL=SER=111119,                     X
//                DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSIN     DD    *
COPYOPER    COPY  OUTDD=INOUT1
                  INDD=INOUT5,(INOUT6,R),INOUT1
```



IEBCOPY Example 10. Compress In Place Following Full Copy With Replace

IEBCOPY Example 11

Operation	Data Set Organization	Input Device	Output Device	Comments
COPY	Input-PARTITIONED Output-PARTITIONED	DISK - 2311 DISK - 2311 DISK - 2311 DISK - 2311 DISK - 2311	DISK - 2311 DISK - 2311 DISK - 2311	1. Three copy operations; one with multiple steps.

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 three copy operations that are made up of one or three copy steps. It uses most of the applications illustrated in IEBCOPY Examples 1 - 10.

- The INOUTA DD Statement: defines a partitioned data (DATASETA). This data set contains eight members (MA, MB, MC, MD, ME, MF, MG, and MH) and resides on a 2311 Disk Storage Volume.

- The INOUTB DD Statement: defines a partitioned data set (DATASETB). This data set resides on a 2311 Disk Storage Volume and contains two members (MA and MJ).

- The INOUTC DD Statement: defines a partitioned data set (DATASETC) which resides on a 2311 Disk Storage Volume. The data set contains four members (MF, ML, MM, and MN).

- The INOUTD DD Statement: defines a partitioned data set (DATASETD). This data set resides on a 2311 Disk Storage Volume and contains two member (MM and MP).

- The INOUTE DD Statement: defines a partitioned data set (DATASETE). This data set contains four members (MD, ME, MF, and MT) and resides on a 2311 Disk Storage Volume.

- The INOUTX DD Statement: 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 2311 Disk Storage Volume. Two blocks are allocated for directory entries.

- The SYSUT3 DD Statement: defines the temporary spill data set to be used if there is not enough space in main storage for the input partitioned data set's directory entries. One track is allocated on a 2311 Disk Storage Volume. 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.

- The SYSUT4 DD Statement: 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. One track is allocated on a 2311 Disk Storage Volume. 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.

- The SYSIN DD Statement: defines the control data set, which follows in the input stream. The data set contains two COPY utility control statements, several INDD statements, SELECT and EXCLUDE utility control statements.

- 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 will subsequently be compressed in place.

- The second COPY Statement: indicates the start of the second copy operation. The absence of a SELECT or EXCLUDE statement causes a default to a full copy; however, the same DD statement is specified for both the INDD and OUTDD operands, thus causing a compress in place of the specified data set. The OUTDD operand specifies INOUTA as the DD statement for the output data set (DATASETA). The INDD operand also specifies INOUTA as the DD statement for the input data set (DATASETA).

- The INDD Statement: 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 presence of both an EXCLUDE statement and a SELECT statement in this copy operation causes an exclusive copy (first copy step) followed by a selective copy (second copy step). In addition there is another INDD statement following the SELECT statement that causes a default to a full copy (third copy step). The OUTDD operand specifes INOUTA as the DD statement for the output data set (DATASETA).

- 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 onto the output data set (DATASETA) because member 'MM' is specified on the EXCLUDE statement.

- The EXCLUDE Statement: specifies the member(s) to be excluded from this copy step.

- The second INDD Statement: specifies INOUTC as the DD statement for the second input data set (DATASETC) to be processed. Processing occurs as follows:

  1. Member ML is searched for and found.
  2. Member ML is copied onto the output data set (DATASETA), although 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.

- The SELECT Statement: specifies the member to be selected from the input data set (DATASETC) to be copied onto the output partitioned data set.

- The third INDD Statement: 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.

  Note: This example is designed to show the user that a copy operation can contain any number of copy steps. (The shaded areas show the separate COPY steps; the COPY statements mark the start of each copy operation.) IEBCOPY Example 11 utilizes many of the applications explained in IEBCOPY Examples 1-10. Of special interest in this example is the fact that the output data set is compressed in place first to save space since it is known that it contains embedded unused space.

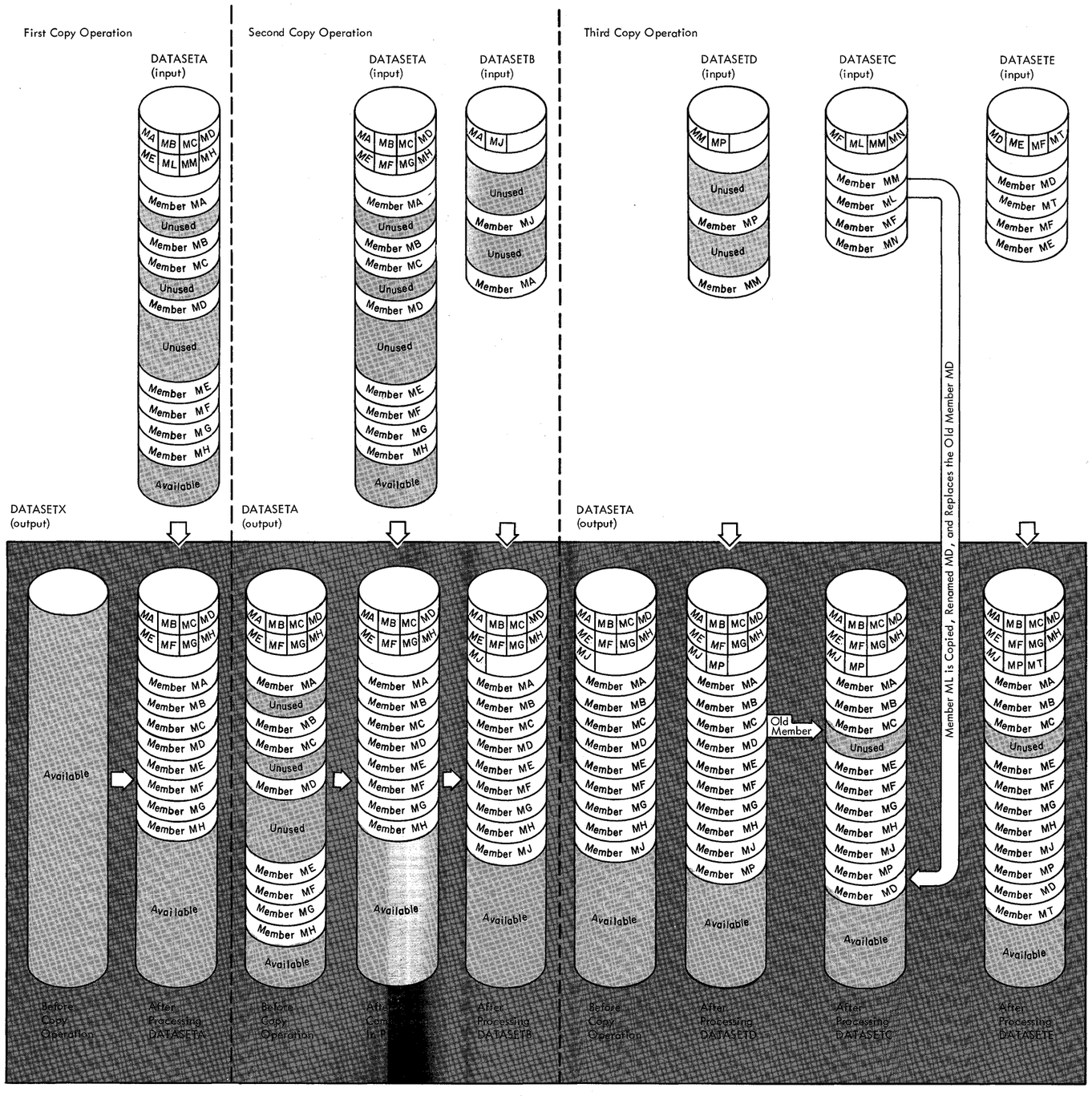
```
//COPY      JOB       06#990,MCEWAN
//JOBSTEP   EXEC      PGM=IEBCOPY
//SYSPRINT  DD        SYSOUT=A
//INOUTA    DD        DSNAME=DATASETA,UNIT=2311,VOL=SER=111113,                X
//                   DISP=OLD
//INOUTB    DD        DSNAME=DATASETB,UNIT=2311,VOL=SER=111115,                X
//                   DISP=(OLD,KEEP)
//INOUTC    DD        DSNAME=DATASETC,UNIT=2311,VOL=SER=111114,                X
//                   DISP=(OLD,KEEP)
//INOUTD    DD        DSNAME=DATASETD,UNIT=2311,VOL=SER=111116,                X
//                   DISP=OLD
//INOUTE    DD        DSNAME=DATASETE,UNIT=2311,VOL=SER=111117,                X
//                   DISP=OLD
//INOUTX    DD        DSNAME=DATASETX,UNIT=2311,VOL=SER=111112,                X
                     DISP=(NEW,KEEP),SPACE=(TRK,(5,1,2))
//SYSUT3    DD        DSNAME=TEMP1,UNIT=2311,VOL=SER=111118,                   X
//                   DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSUT4    DD        DSNAME=TEMP2,UNIT=2311,VOL=SER=111119,                   X
//                   DISP=(NEW,DELETE),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
/*
```



IEBCOPY Example 11. Multiple Copy Steps Within a Copy Operation

IEBCOPY Example 12

Operation	Data Set Organization	Input Device	Output Device	Comments
COPY	Input-PARTITIONED Output-PARTITIONED	DISK - 2311 DISK - 2311 DISK - 2311 DISK - 2311	DISK - 2311	1. Multiple copy operations with multiple steps.

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 with multiple copy steps. It uses many of the applications illustrated in IEBCOPY Examples 1-10.

- The INOUTA DD Statement: defines a partitioned data set (DATASETA). This data set contains three members (MA, MB, and MD) and resides on a 2311 Disk Storage Volume.

- The INOUTB DD Statement: defines a partitioned data set (DATASETB). This data set resides on a 2311 Disk Storage Volume and contains two members (MA and MJ).

- The INOUTC DD Statement: defines a partitioned data set (DATASETC) which resides on a 2311 Disk Storage Volume. This data set contains four members (MF, ML, MM, and MN).

- The INOUTD DD Statement: defines a partitioned data set (DATASETD). This data set resides on a 2311 Disk Storage Volume and contains two members (MM and MP).

- The INOUTE DD Statement: defines a partitioned data set (DATASETE) which resides on a 2311 Disk Storage Volume. This data set contains three members (MA, MJ and MK).

- The SYSUT3 DD Statement: defines the temporary spill data set to be used if there is not enough space in main storage for the input partitioned data set's directory entries. One track is allocated on a 2311 Disk Storage Volume. 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.

- The SYSUT4 DD Statement: 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. One track is allocated on a 2311 Disk Storage Volume. 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.

- The SYSIN DD Statement: defines the control data set, which follows in the input stream. The data set contains three COPY utility control statements, several INDD statements, SELECT and EXCLUDE statements.

- The first COPY Statement: indicates the start of the first copy operation. The presence of a SELECT statement and an EXCLUDE statement in this copy operation causes a selective copy (first copy step) followed by an exclusive copy (second copy step). 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. It is not searched for again.
2. Member MJ is searched for and found. It is not searched for again.
3. Member MA is not copied onto the output data set because the replace option is not specified on either the INDD or MEMBER level.
4. Member MJ is copied onto the output data set.

- The SELECT Statement: specifies the members (MA and MJ) to be selected from the input data set (DATASETE) to be copied onto the output data set.

- The second INDD Statement: specifies INOUTC as the DD statement for the second input data set (DATASETC) to be processed. Only members MF and ML may be copied because members MM and MN are specified on the EXCLUDE statement. Both MF and ML are copied because neither exists on the output data set.

- The EXCLUDE Statement: specifies the members (MM and MN) to be excluded from the copy step.

- The second COPY Statement: indicates the start of the second copy operation. The absence of a SELECT or EXCLUDE statement causes a default to a full copy. The OUTDD (O) operand specifies INOUTB as the output data set (DATASETB). The INDD operand specifies INOUTD as the first input data set (DATASETD) to be processed. Both members (MP and MM) are copied onto the output data set.

- The INDD (I) Statement: 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.

  Processing occurs as follows:

  1. Member MF is copied from DATASETC.
  2. Member ML is copied from DATASETC.
  3. Member MM is copied from DATASETC, although it already exists on the output partitioned data sets, because the replace option is specified. (Note: The pointer in the output 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.)
  4. Member MN is copied from DATASETC.

  Since DATASETB is also the data set specified in the OUTDD operand, a compress in place takes place.

- The third COPY Statement: indicates the start of the third copy operation. The presence of a SELECT statement causes a selective copy. The OUTDD (O) operand specifies INOUTD as the DD statement for the output data set (DATASETD). The INDD (I) operand specifies INOUTB as the DD statement for the input data set (DATASETB).

  The selected member (MM) is found on the input data set (DATASETB) and copied onto the output data set (DATASETD) because the replace option is specified on the INDD level.

  (Note: The pointer in the data set directory is changed to point at the new (copied) member MM; thus the space occupied by the old member MM is embedded unused space.)

- The SELECT Statement: specifies the member (MM) to be selected from the input partitioned data set (DATASETB) to be copied onto the output partitioned data set.

  Note: This example is designed to show the user that there can be more than one copy operation in a job step, as well as multiple copy steps within a copy operation. (The shaded areas show the separate copy steps; the COPY statements mark the start of each copy operation.) IEBCOPY Example 12 uses many of the applications expained in IEBCOPY Examples 1-10. Of special interest in this example are the following:

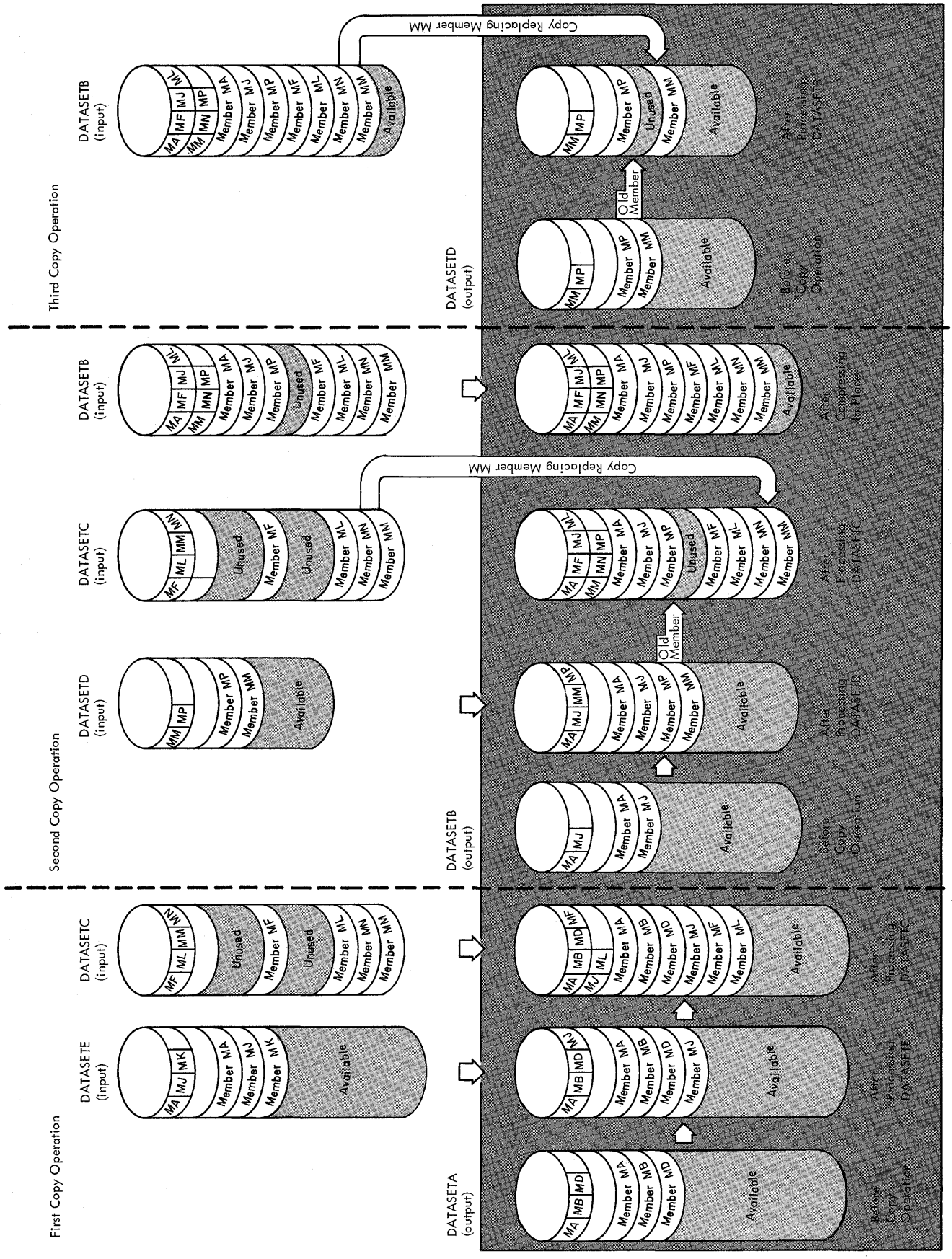
- Not all of the data sets defined by the DD statements are used in every copy operation.
- Data sets used as input data sets in one copy operation can be used as output data sets in another copy operation, and vice versa.

```
//COPY     JOB      06#990,MCEWAN
//JOBSTEP  EXEC     PGM=IEBCOPY
//SYSPRINT DD       SYSOUT=A
//INOUTA   DD       DSNAME=DATASETA,UNIT=2311,VOL=SER=111113,                X
//                 DISP=OLD
//INOUTB   DD       DSNAME=DATASETB,UNIT=2311,VOL=SER=111115,                X
//                 DISP=(OLD,KEEP)
//INOUTC   DD       DSNAME=DATASETC,UNIT=2311,VOL=SER=111114,                X
//                 DISP=(OLD,KEEP)
//INOUTD   DD       DSNAME=DATASETD,UNIT=2311,VOL=SER=111116,                X
//                 DISP=OLD
//INOUTE   DD       DSNAME=DATASETE,UNIT=2311,VOL=SER=111117,                X
//                 DISP=OLD
//SYSUT3   DD       DSNAME=TEMP1,UNIT=2311,VOL=SER=111118,                   X
//                 DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSUT4   DD       DSNAME=TEMP2,UNIT=2311,VOL=SER=111119,                   X
//                 DISP=(NEW,DELETE),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),INOUTE)

           COPY     O=INOUTD,I=((INOUTB,R))
           SELECT   MEMBER=MM
/*
```



IEBCOPY Example 12. Multiple Copy Operations Within A Job Step

Program Applications

The IEBGENER program can copy a sequential data set or a partitioned member, or it can create a partitioned data set from a sequential data set or from a partitioned member used as input. The program can expand an existing partitioned data set by creating partitioned members and merging them into the data set that is to be expanded.

The IEBGENER program provides optional editing facilities with all applications of the program. In addition, user exits are provided at appropriate places for user routines that process labels, manipulate input data, create keys, and handle uncorrectable input/output errors. Refer to the section "Exit Routine Linkage" for a discussion of linkage conventions that are applicable when user routines are provided.

The program can be used to:

- Create a back-up copy of a sequential data set or a partitioned member.
- Produce a partitioned data set from sequential input.
- Expand an existing partitioned data set.
- Produce an edited sequential or partitioned data set.
- Reblock or change the logical record length of a data set.
- Create user labels on sequential output data sets.

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

Creating a Back-Up Copy

A back-up copy of a sequential data set (or partitioned member) can be produced by copying the data set (or member) onto any IBM supported output device. For example:

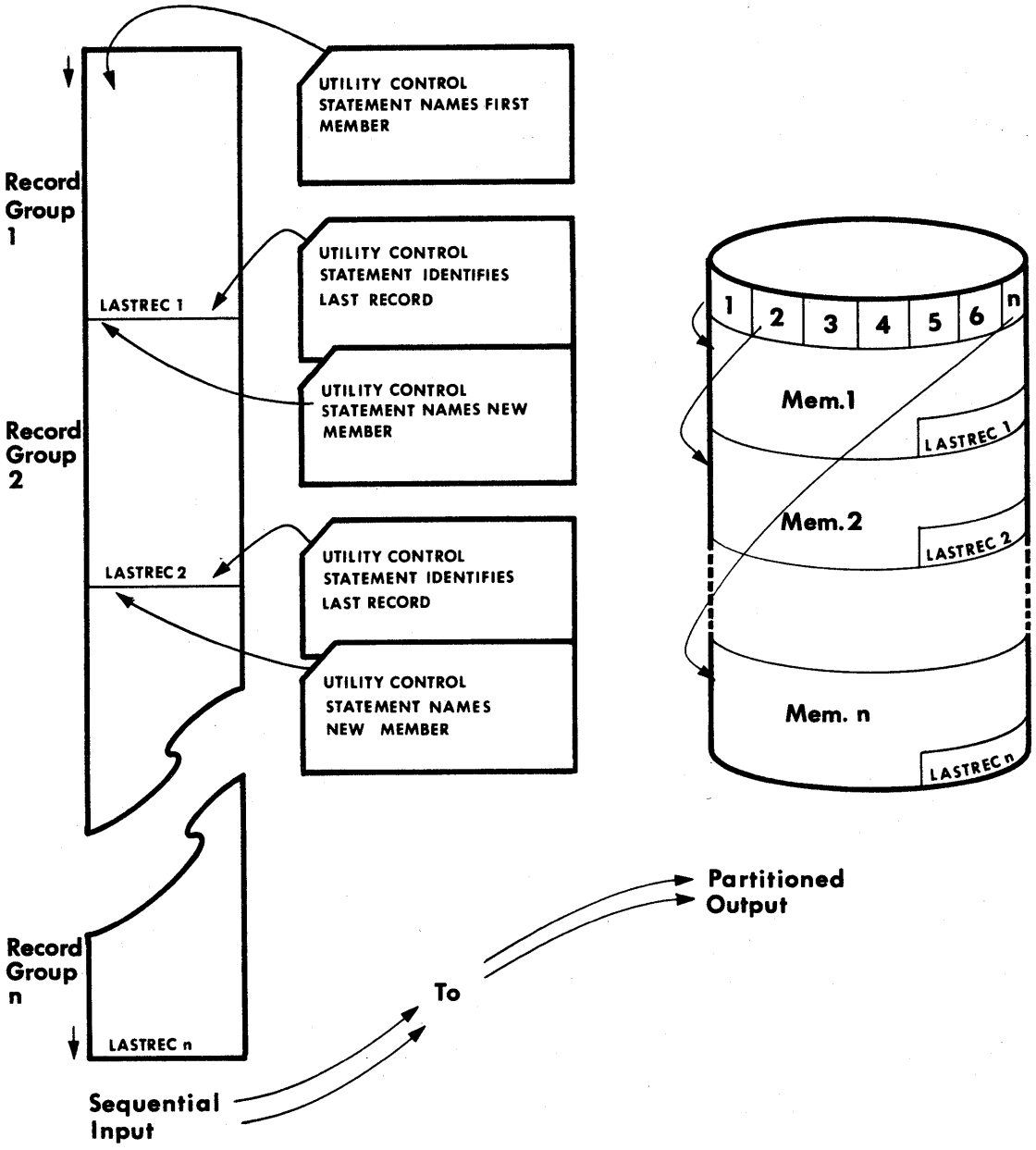
- Card to tape (7 or 9-track).
- Card to direct access.
- Card to printer or punch.
- Tape (7 or 9-track) to tape (7 or 9-track).
- Tape (7 or 9-track) to direct access.
- Tape (7 or 9-Track) to printer or punch.
- Direct-access to direct access.
- Direct-access to tape (7 or 9-track, sequential data sets only).
- Direct-access to printer or punch (sequential data sets).

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

Producing a Partitioned Data Set From Sequential Input

The IEBGENER program can produce a partitioned data set from sequential input. Through the use of utility control statements, the user can logically divide the sequential data set into "record groups" and assign member names to the appropriate record groups. The IEBGENER program places the newly created members into a partitioned (output) data set.

IEBGENER Figure 1 shows how a partitioned data set is produced from a sequential data set used as input. In this figure, each record group within the sequential data set becomes a member of the resulting partitioned data set.

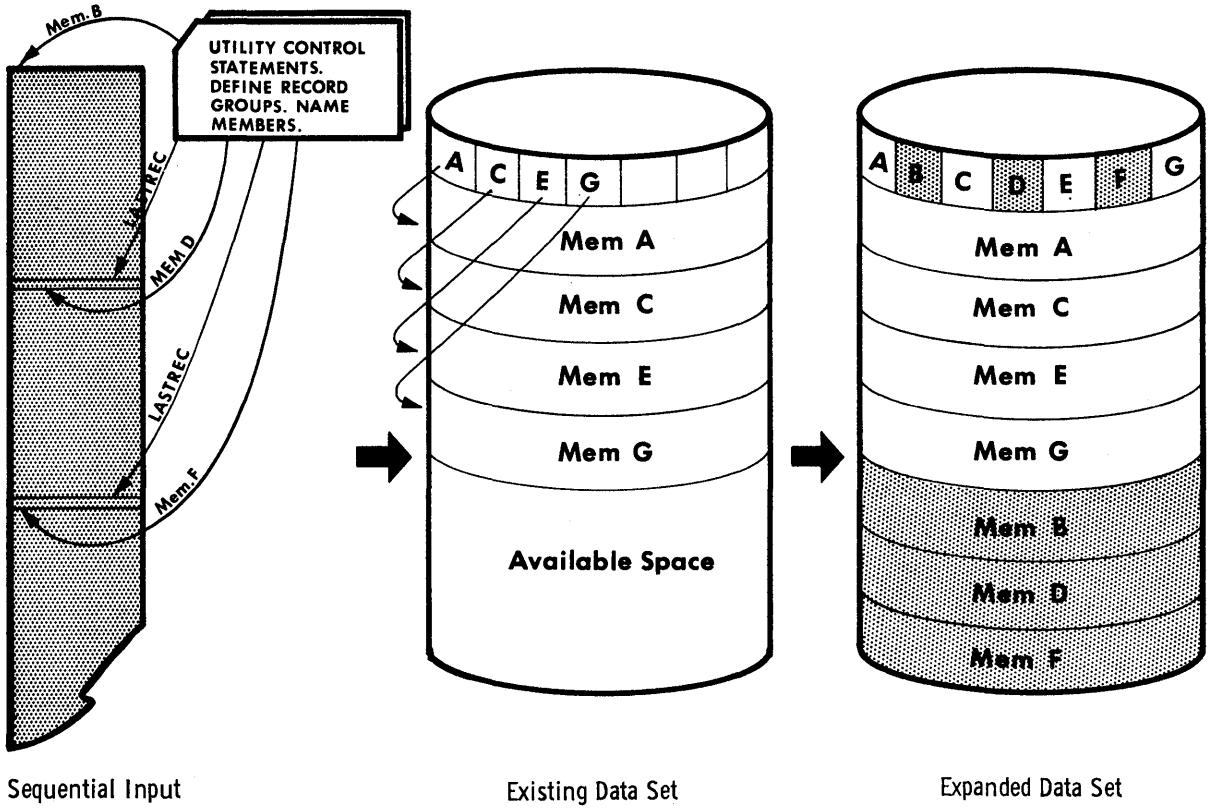


IEBGENER Figure 1. Creating a Partitioned Data Set from Sequential Input

Expanding a Partitioned Data Set

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

IEBGENER Figure 2 shows how sequential input to the program is converted into members that are merged into the existing partitioned data set.



IEBGENER Figure 2. Expanding a Partitioned Data Set

Producing an Edited Data Set

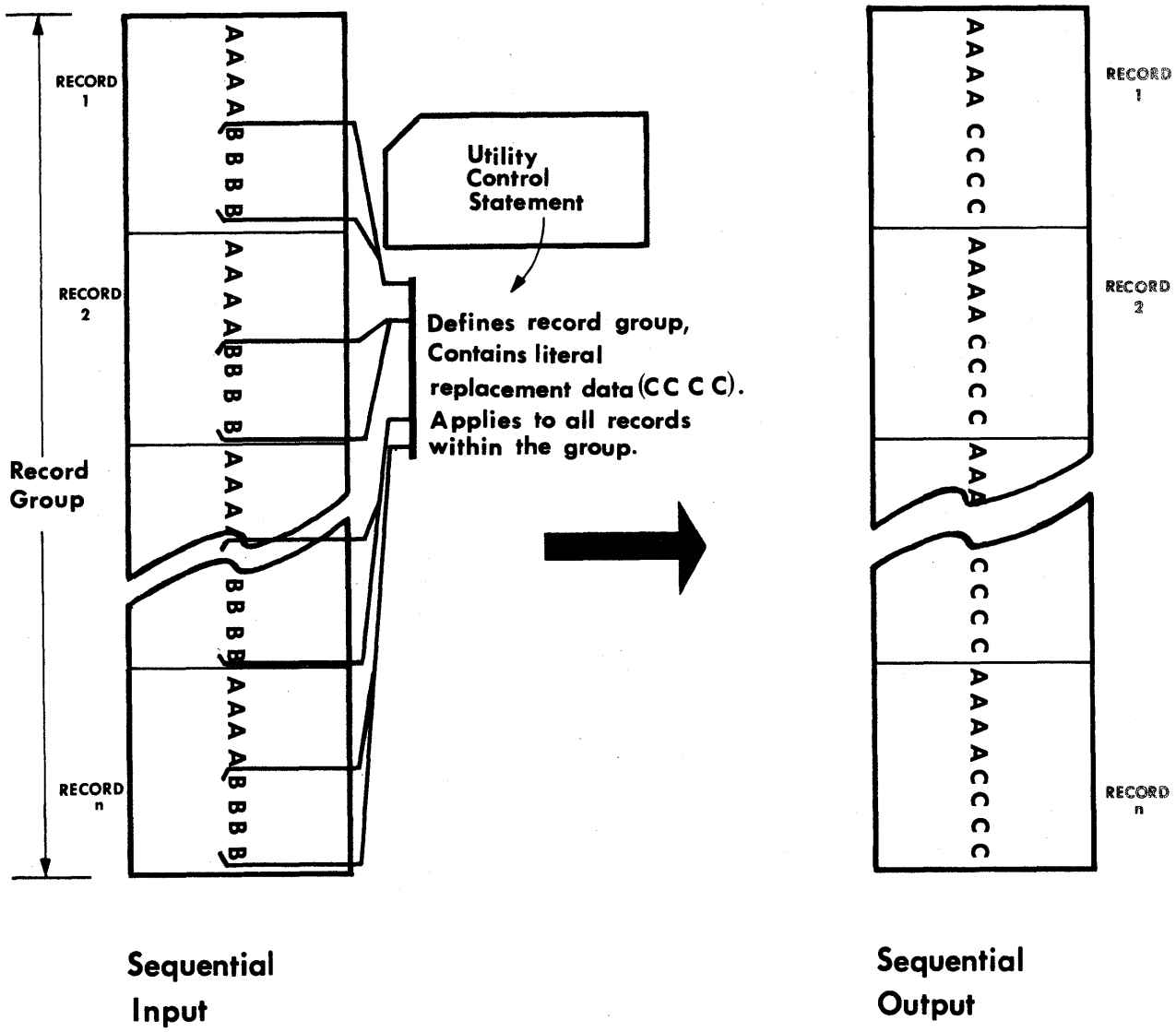
The IEBGENER program can produce an edited sequential or partitioned data set. Through the use of utility control statements, the user can specify editing information that applies to:

- A record.
- A group of records.
- Selected groups of records.
- An entire data set.

An edited data set can be produced by:

- Rearranging or omitting defined data fields within a record.
- Supplying literal information as replacement data.
- Converting data from packed to unpacked decimal mode, unpacked to packed decimal mode, or H-set BCD to EBCDIC mode.

IEBGENER Figure 3 shows an edited sequential data set. In this figure, literal replacement information is supplied for information within a defined field. (Data is rearranged, omitted, or converted in the same manner.)



IEBGENER Figure 3. Editing a Record Group

Note: The IEBGENER program will perform no editing if the input and output data sets consist of VS or VBS type records and have equal blocksizes and logical record lengths. In this case the utility ignores any utility control statements that specify editing. The utility performs a "straight copy"; 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 and/or Changing the Logical Record Length of a Data Set

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

Inputs and Outputs

IEBGENER Table 1 lists the major inputs to and outputs from the IEBGENER program.

IEBGENER Table 1. Data Sets Used (Input) and Produced (Output) by the IEBGENER Program

Input	Input Data Set: This data set 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. Control Data Set: This data set contains utility control statements. The data set is required if editing is to be performed or if the output data set is to be a partitioned data set.
Output	Output Data Set: This data set is the result of the IEBGENER operation. The data set can be either sequential or partitioned. It can be either a new data set (i.e., created during the present job step) or an existing partitioned data set that is to be expanded. Message Data Set: This data set contains informational messages (e.g., the contents of applicable utility control statements) and error messages, if applicable.

ADDITIONAL OUTPUTS

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

- 00 -- successful completion.
- 04 -- probable successful completion. A warning message to the user is written.
- 08 -- processing was terminated after the user requested processing of user header labels only.
- 12 -- an unrecoverable error has occurred. The job step is terminated.
- 16 -- a user routine has passed a return code of 16 to the IEBGENER program. The job step is terminated.

Control

The IEBGENER program is controlled by job control statements and utility control statements. The job control statements are required to execute or invoke the IEBGENER 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 the IEBGENER program.

JOB CONTROL STATEMENTS

IEBGENER Table 2 shows the job control statements necessary for executing or invoking the IEBGENER program.

IEBGENER Table 2. Job Control Statements for the IEBGENER Program (Part 1 of 2)

Statement	Usage
JOB statement	This statement initiates the job.
EXEC statement	This statement specifies the program name (PGM=IEBGENER) or, if the job control statements for the IEBGENER program reside in a procedure library, the procedure name.
SYSPRINT DD statement	This statement 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. (This DD statement must be present for each execution or invocation of the IEBGENER program.)
SYSUT1 DD statement	This statement defines the input data set. It can define a sequential data set on a card reader, a magnetic tape volume, or a direct access device, or, it can define a partitioned member on a direct access device.
SYSUT2 DD statement	This statement defines the output data set. It can define a sequential data set on a card punch, a printer, a magnetic tape volume, or a direct access device, or, it can define a partitioned data set on a direct access device. Space must be allocated for an output data set that is to reside on a direct access device unless the data set is an expanded data set, in which case space must not be allocated. If the output data set is on a card punch or a printer, the user must specify DCB information on the SYSUT2 DD statement. Note: If both the SYSUT1 and the SYSUT2 DD statements specify standard user labels (SUL), the IEBGENER program will copy user labels from SYSUT1 to SYSUT2. See "Appendix F: Utility Program Handling of User Labels" for a discussion of the available options for user label processing.
SYSIN DD statement	This statement 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 also be defined as being a member within a library of partitioned members. The SYSIN DD statement is required, regardless of the operation.

Notes: 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 blocksize for the SYSPRINT (message) data set must be a multiple of 121. The blocksize for the SYSIN (control) data set must be a multiple of 80. Any blocking factor can be specified for these blocksizes.

(Part 1 of 2)

IEBGENER Table 2. Job Control Statements for the IEBGENER Program (Part 2 of 2)

Both the input data set and the output data set can contain fixed-length, variable-length, undefined-length, or variable spanned records. Fixed-length, variable-length, and variable spanned 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-length or variable-length records, keys can be retained only by using the appropriate user exit. • For variable spanned records, keys can never be retained. Refer to the publication IBM System/360 Operating System: Supervisor and Data Management Services, GC28-6646, for information on estimating space allocations. Refer to the IEBGENER examples for typical uses of the job control statements.

UTILITY CONTROL STATEMENTS

The IEBGENER program is controlled by five utility control statements:

- The GENERATE statement.
- The EXITS statement.
- The LABELS statement
- The MEMBER statement.
- The RECORD statement.

The control statements are included in the control data set as required. IEBGENER Table 3 shows the order of inclusion and the uses of the control statements.

IEBGENER Table 3. Utility Control Statements for the IEBGENER Program

This Statement is Included	If
GENERATE	(1) The output is partitioned, or (2) editing is to be performed, or (3) user routines are provided and/or label processing is specified. This statement appears first in the control data set.
EXITS	User routines are provided. This statement, if included, should appear immediately after the GENERATE statement.
LABELS	The user wants either (1) no user labels to be copied onto the output data set, (2) user labels to be copied onto the output data set from records in the data portion of the SYSIN data set, or (3) user labels to be copied onto the output data set after they are modified by the user's label processing routines. See "Appendix F: Utility Program Handling of User Labels" for a more detailed discussion of this statement.
MEMBER	The output is partitioned. One MEMBER statement must be included for each member created by the IEBGENER program.
RECORD	(1) The output consists of partitioned members, or (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. (In all of these cases, this statement defines a record group.)
If no utility control statements are included in the control data set, the entire input data set is copied sequentially.	

The GENERATE Statement

The GENERATE statement must appear before other statements. If it contains errors, or is inconsistent with other statements, the program is terminated.

Name	Operation	Operand
[name]	GENERATE	[MAXNAME=n] [MAXFLDS=n] [MAXGPS=n] [MAXLITS=n]

MAXNAME=n
specifies a number that is no less than the total number of members names and aliases appearing in subsequent MEMBER statements. MAXNAME is required if there are one or more MEMBER statements.

MAXFLDS=n
specifies a number that is no less than the total number of FIELD parameters appearing in subsequent RECORD statements. MAXFLDS is required if there are any FIELD parameters in subsequent RECORD statements.

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=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.)

The EXITS Statement

The EXITS statement is used to identify exit routines supplied by the user. The exits OUTHDR and OUTTLR are ignored if the output data set is partitioned. Linkages to and from exit routines are discussed in Appendix A.

Name	Operation	Operand
[name]	EXITS	[INHDR=routinename]
		[OUTHDR=routinename]
		[INTLR=routinename]
		[OUTTLR=routinename]
		[KEY=routinename]
		[DATA=routinename]
		[IOERROR=routinename]
		[TOTAL=(routinename,size)]

INHDR=routinename
specifies the symbolic name of a routine that processes user input header labels.

OUTHDR=routinename
specifies the symbolic name of a routine that creates user output header labels.

INTLR=routinename
specifies the symbolic name of a routine that processes user input trailer labels.

OUTTLR=routinename
specifies the symbolic name of a routine that processes user output trailer labels.

For a detailed discussion of the processing of user labels as data set descriptors, refer to "Appendix F: Utility Program Handling of User Labels."

KEY=routinename
specifies the symbolic name of a routine that creates the output record key. (This routine can never receive control when the utility is processing a data set consisting of VS or VBS type records, since no processing of keys is permitted for this type of data.)

DATA=routinename
specifies the symbolic name of a routine that modifies the physical record (logical record for VS or VBS type records) before it is processed by the IEBGENER program.

IOERROR=routinename
specifies the symbolic name of a routine that handles permanent input/output error conditions.

TOTAL=(routinename,size)
specifies that exits to a user's routine are to be provided prior to writing each record. When the option is specified, the user must supply two parameters. The first is the name of the user's totaling routine. The other is the size (number of bytes) needed to contain his desired totals, counters, pointers, etc. In addition, the keyword OPTCD=T must be specified for the SYSUT2 (output) DD statement.

Refer to the section "Exit Routine Linkage" for a discussion of linkage conventions for user routines.

The above line parameters are valid only when the utility is processing sequential data sets. For a detailed discussion of the processing of user labels as data set descriptors, and for discussion of user label totaling), refer to "Appendix F: Utility Program Handling of User Labels."

The LABELS Statement

The LABELS statement specifies whether or not user labels are to be treated by the IEBGENER program as data. For a detailed discussion of this option, refer to the section entitled "Processing User Labels as Data," in "Appendix F: Utility Program Handling of User Labels."

Name	Operation	Operand
[name]	LABELS	DATA={YES / NO / ALL / ONLY / INPUT}

The MEMBER Statement

The MEMBER statement provides the name and alias names 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.

Name	Operation	Operand
[name]	MEMBER	NAME=(name[,alias]...)

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.

The RECORD Statement

The RECORD statement is used to define a record group and to supply editing information. A record group consists of one or more records that are to be processed identically. The RECORD statement defines a record group by identifying the last record of the group with a literal name.

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

Name	Operation	Operand
[name]	RECORD	[IDENT=(ident parameters)][FIELD = (field parameters)] LABELS=n

IDENT=(length,'name',input-location)
identifies the last record of the input group to which the FIELD parameters or MEMBER statement applies. If the IDENT statement is not followed by additional RECORD or MEMBER statements, it also defines the last record to be processed.

length: specifies the length (in bytes) of the identifying name. The length cannot exceed 8 characters.

'name': specifies the exact literal that identifies the last input record of a record group.

input-location: specifies the starting location of the field that contains the identifying name in the input records.

If IDENT is omitted, or if no match for 'name' is found, the remainder of the input data is considered to be in one record group; i.e., subsequent RECORD and MEMBER statements are ignored.

LABELS=n
where n=1 to 8, indicates that the following n records in the SYSIN data set will be treated as user labels. The number n must specify the exact number of label records that follow the RECORD statement.

If a RECORD LABELS=n statement is used, a LABELS DATA=INPUT statement must appear before it in the input stream.

If both output header labels and output trailer labels are to be contained in the SYSIN data set, the user must include one RECORD LABELS=n statement for each label group. (If the user wants to include only output trailer labels in the SYSIN data set, he must include a RECORD LABELS=0 statement to indicate that there are no output header labels in the SYSIN data set. This statement must precede the RECORD LABELS=n statement which signals the start of label input records.)

For a detailed discussion of this option, refer to the section entitled "Processing User Labels As Data", in "Appendix F: Utility Program Handling of User Labels."

FIELD=([length],[input-location-or-'literal'],[conversion],[output-location])...

specifies the field processing and editing information. FIELD parameters can determine the logical record length of output records.

length: specifies the length (in bytes) of the input field or literal to be processed. If no length is specified, a length of 80 bytes is assumed by the IEBGENER program. (A length of 40 or less must be specified if a literal is to be processed.)

input-location-or-'literal': specifies the starting byte of the field to be processed or, if enclosed in apostrophes, the literal (maximum length of 40 bytes) to be placed in the specified output location. If no input location is specified, the starting byte is assumed to be byte 1. If a literal contains apostrophes, each apostrophe must be written as two consecutive apostrophes.

conversion: specifies a 2-byte code that indicates the type of conversion to be performed on this field, as follows:

Code	Conversion	Output length (input length=L)
PZ	Packed to unpacked decimal mode	2L-1
ZP	Unpacked to packed decimal mode	(L/2)+C *
HE	H-set BCD to EBCDIC mode	L

* If L is odd, C is 1/2; if L is even, C is 1.

Note: 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 byte maximum applies.

If no conversion is specified, the field is moved to the output area without change.

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.

output-location: specifies the starting location of this field in the output records. If no output location is specified, byte 1 is assumed.

Contents of unspecified fields in the output records are not erased or modified by the IEBGENER program. These contents remain the same as they were before the program was executed.

Note: IDENT and FIELD parameters are ignored in "straight copy" processing of data sets that contain VS or VBS type records.

Using the Utility Control Statements

IEBGENER Table 4 shows the use of the utility control statements.

IEBGENER Table 4. Use of the GENERATE, EXITS, MEMBER, and RECORD Statements

TO	WITH	USE
Produce sequential output from a sequential data set or a partitioned member used as input	No editing, no user exits	No control statements
	Editing	GENERATE --1 per job step. RECORD -- as many per job step as required.
	Editing, user exits	GENERATE --1 per job step. EXITS --1 per job step. RECORD -- as many per job step as required.
Produce partitioned output from sequential input or from a partitioned member used as input	No editing, no user exits	GENERATE --1 per job step. MEMBER --1 per output member. RECORD --1 after each MEMBER statement (none required after last MEMBER statement).
	Editing	GENERATE --1 per job step. MEMBER --1 per output member. RECORD --1 or more (as required) after each MEMBER statement.
	Editing, user exits	GENERATE --1 per job step. EXITS --1 per job step MEMBER --1 per output member. RECORD --1 or more (as required) after each MEMBER statement.

Coding Utility Control Statements

Utility control statements are coded in columns 1 through 71. A statement that exceeds 71 characters may be continued on one or more additional cards.

Names: Names are not required for utility control statements. If they are used, they must begin in column 1. Names cannot be over eight characters long and must be followed by one or more blanks.

Operation: The operation field can begin in any column from 2 through 71. An operation field may be interrupted at column 71 and continued on the next card. When an operation field is interrupted, a nonblank character must be placed in column 72. The continued portion must start in any column from 4 through 16 of the next card. (Restriction: When an operation field ends in column 70 or 71, it cannot be followed by an operand or comment.)

Operands (Keywords and Parameters): Operands must be preceded by an operation field. They must begin on the same card that the operation field ends on. One or more blanks must separate the operand from the

operation field. When an operand ends in column 71, then column 72 must be left blank. If an operand is interrupted at column 71 and is to be continued on the next card, a nonblank character must be placed in column 72. An operand can also be interrupted at a comma (unless the comma is part of a literal). Whenever an operand is interrupted at a comma, column 72 may be either blank or nonblank. In all cases, the continuation must start in any column from 4 through 16 of the next card.

Comments: Comments must be preceded by an operand or operation field. They must begin on the same card that the last operand or the operation field ends on. (Exception: When an operand is interrupted at a comma, the remaining columns through 71 may contain a comment. A blank must separate the comma from the comment, however.) A comment cannot separate an operation field from an operand.

Comments may be interrupted and continued on the next card by placing a nonblank character in column 72. Comments do not have to be interrupted at column 71; blanks are acceptable in column 71 as part of the comment. The continued portion must start in any column from 4 through 16 of the next card.

IEBGENER Examples

The following examples illustrate some of the uses of the IEBGENER program.

IEBGENER Example 1

Operation	Data Set Organization	Input Device	Output Device	Comments
COPY	Input-SEQUENTIAL Output-SEQUENTIAL	CARD READER	TAPE - 9-track, standard label, 800 bits-per-inch	1. Blocked output.

In this example, a sequential data set (card input) is to be copied onto 9-track magnetic tape.

- The SYSIN DD Statement: defines a dummy data set. (No editing is to be performed; therefore, no utility control statements are needed.)
- The SYSUT2 DD Statement: defines the copied sequential data set (output). The data set is written onto a 9-track magnetic tape drive at a density of 800 bits-per-inch. The data set is to reside as the first (or only) data set on the magnetic tape volume.
- The SYSUT1 DD Statement: defines the input card data set. The data set can contain no // cards.

```
//CDTOTAPE JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DUMMY
//SYSUT2   DD   DSNAME=OUTSET,UNIT=2400,LABEL=(,SL),DISP=(,KEEP),
//              VOLUME=SER=001234,DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//SYSUT1   DD   *
            .
            .
            input card data set
            .
            .
/*
```

IEBGENER Example 1. Copying a Card Data Set onto 9-Track Magnetic Tape

IEBGENER Example 2

Operation	Data Set Organization	Input Device	Output Device	Comments
COPY-with editing	Input-SEQUENTIAL Output-SEQUENTIAL	CARD READER	TAPE- 7-track, standard label 556 bits-per-inch density, data conversion	1. Blocked output. 2. Utility control statements exist as a member of a partitioned data set.

In this example, a sequential data set (card input) is to be copied onto a 7-track magnetic tape volume.

- The SYSIN DD Statement: defines the data set containing the utility control statements. The statements have been previously placed into a partitioned data set. The set of control statements was assigned the member name STMNTS when it was placed in the partitioned data set.
- The SYSUT2 DD Statement: defines the copied sequential data set (output). The data set is written as the first or only data set on the volume. It is written at 556 bits-per-inch density on a 7-track magnetic tape volume.
- The SYSUT1 DD Statement: defines the input card data set. The data set can contain no // cards.

```
//CDTOTAPE JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DSNAME=CNTRLIBY(STMNTS),UNIT=2311,DISP=(OLD,KEEP),
//             VOLUME=SER=111112,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSUT2   DD   DSNAME=OUTSET,UNIT=2400-2,LABEL=(,SL),
//             DCB=(DEN=1,RECFM=FB,LRECL=80,BLKSIZE=2000,TRTCH=C),
//             DISP=(,KEEP),VOLUME=SER=001234
//SYSUT1   DD   *
                .
                .
                input card data set
                .
                .
/*
```

IEBGENER Example 2. Copying a Card Data Set Onto 7-Track Magnetic Tape

IEBGENER Example 3

Operation	Data Set Organization	Input Device	Output Device	Comments
COPY- with editing	Input-SEQUENTIAL Output-SEQUENTIAL	CARD READER	TAPE- 9-track, standard label, 800 bits-per-inch density	1. Blocked output. 2. Utility control statements (not shown) exist as a member of a partitioned data set. 3. Input data includes // cards.

In this example a card input sequential data set is to be copied onto a 9-track magnetic tape volume. The input contains cards that have slashes (//) in columns 1 and 2.

- The SYSIN DD Statement: defines the data set containing the utility control statements. The statements have been previously placed into a partitioned data set. The set of control statements was assigned the member name STMNTS when it was placed in the partitioned data set.
- The SYSUT2 DD Statement: defines the copied sequential data set (output). The data set is written as the second data set on the specified magnetic tape volume.
- The SYSUT1 DD Statement: defines the input card data set. The data set is to be edited as specified in the utility control statements (not shown). The input card data set contains // cards.

```
//CDTOTAPE JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DSNAME=CNTRLIBY(STMNTS),UNIT=2311,DISP=(OLD,KEEP),
//             VOLUME=SER=111112,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSUT2   DD   DSNAME=OUTSET,UNIT=2400,LABEL=(2,SL),VOLUME=SER=001234,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000),DISP=(,KEEP)
//SYSUT1   DD   DATA
             .
             .
             input card data set (including // cards)
             .
             .
/*
```

IEBGENER Example 3. Copying a Card Data Set Onto 9-Track Tape -- Input Includes // Cards

IEBGENER Example 4

Operation	Data Set Organization	Input Device	Output Device	Comments
COPY-with editing	Input-SEQUENTIAL Output-SEQUENTIAL	CARD READER	DISK - 2311	1. Blocked output. 2. Utility control statements (not shown) exist as a member of a partitioned data set. 3. Input data includes // cards.

In this example, a card input sequential data set is to be copied onto an IBM 2311 Disk Storage Drive. The input contains cards that have slashes (//) in columns 1 and 2.

- The SYSIN DD Statement: defines the data set containing the utility control statements. The control statements have been previously placed into a partitioned data set. The set of control statements was assigned the member name STMNTS when it was placed in the partitioned data set.

- The SYSUT2 DD Statement: defines the output sequential data set. Twenty tracks of primary storage space and ten tracks of secondary space are allocated for the data set on the 2311 Disk Storage Drive.

- The SYSUT1 DD Statement: defines the input card data set. The data set is to be edited as specified in the utility control statements (not shown). The input card data set contains // cards.

```
//CDTOTAPE JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DSNAME=CNTRLIBY(STMNTS),UNIT=2311,DISP=(OLD,KEEP),
//              VOLUME=SER=111112,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSUT2   DD   DSNAME=OUTSET,UNIT=2311,VOLUME=SER=111113,DISP=(,KEEP),
//              SPACE=(TRK,(20,10)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//SYSUT1   DD   DATA
           .
           .
           input card data set (including // cards)
           .
           .
/*
```

IEBGENER Example 4. Copying a Card Data Set Onto a 2311 Disk Storage Volume -- Input Includes // Cards

IEBGENER Example 5

Operation	Data Set Organization	Input Device	Output Device	Comments
PRINT	Input-SEQUENTIAL	CARD READER	PRINTER	1. Input data includes // cards. 2. This example assumes that the system output device is a printer.

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

- The SYSIN DD Statement: defines a dummy data set. (No editing is to be performed; therefore, no utility control statements are required.)
- The SYSUT2 DD Statement: 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 (below) to RECFM=FA.
- The SYSUT1 DD Statement: defines the input card data set. The input data contains // cards.

```
//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)
           .
           .
/*
```

IEBGENER Example 5. Copying a Card Data Set Onto a Printer (Input Includes // Cards)

IEBGENER Example 6

Operation	Data Set Organization	Input Device	Output Device	Comments
CONVERT to partitioned organization	Input-SEQUENTIAL Output-PARTITIONED	TAPE- 9-track, standard label, 800 bits-per-inch	DISK - 2311	1. Blocked output. 2. Three members. 3. Utility control statements in input stream.

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

- The SYSUT1 DD Statement: defines the input sequential data set (INSET). The data set was originally written on a 9-track magnetic tape drive at 800 bits-per-inch density.
- The SYSUT2 DD Statement: defines the output partitioned data set (NEWSET). The data set is to be placed on an IBM 2311 Disk Storage Drive. Twenty tracks of primary space, ten 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 amount of disk storage space required by the data set.
- The SYSIN DD Statement: 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 perform any editing functions.
- The GENERATE Statement: (1) indicates that three member names are included in subsequent MEMBER statements. (2) Indicates that 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.

```
//TAPEDISK JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=INSET,UNIT=2400,LABEL=(,SL),DISP=(OLD,KEEP),
//              VOLUME=SER=001234,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSUT2   DD   DSNAME=NEWSET,UNIT=2311,DISP=(,KEEP),
//              VOLUME=SER=111112,SPACE=(TRK,(20,10,5)),
//              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
/*
```

IEBGENER Example 6. Creating a Partitioned Data Set From Sequential Input (No Editing)

IEBGENER Example 7

Operation	Data Set Organization	Input Device	Output Device	Comments
CONVERT to partitioned organization and EXPAND an existing partitioned data set	Input-SEQUENTIAL Output-PARTITIONED	DRUM - 2301	DRUM - 2301	1. Blocked output. 2. Two members are merged into existing data set. 3. Utility control statements in input stream.

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 will be passed to the user label Exit routines.

- The SYSUT1 DD Statement: defines the input sequential data set (INSET). The input data set resides on an IBM 2301 Drum Storage Device, and is specified to have standard and user labels.

- The SYSUT2 DD Statement: defines the output partitioned data set (EXISTSET). The members created during this job step are merged into the partitioned data set. The output records are blocked to reduce the amount of drum storage space required by the new members.

- The SYSIN DD Statement: 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 perform any editing functions.

- The GENERATE Statement: (1) indicates that two member names and one alias name are included in subsequent MEMBER statements. (2) indicates that an IDENT parameter appears in a subsequent RECORD statement.

- the EXITS Statement: defines the user routines that will get control to process user labels.

- The First MEMBER Statement: assigns a member name (MEMX) and an alias name (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 sequential data set is included in this member.

```
//DRUMDRUM JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=INSET,UNIT=2301,DISP=(OLD,KEEP),LABEL=(,SUL)
//              VOLUME=SER=111112,DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//SYSUT2   DD   DSNAME=EXISTSET,UNIT=2301,DISP=(MOD,KEEP),
//              VOLUME=SER=111113,DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//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
/*
```

IEBGENER Example 7. Expanding a Partitioned Data Set With Members Created From Sequential Input

IEBGENER Example 8

Operation	Data Set Organization	Input Device	Output Device	Comments
COPY - with editing	Input-SEQUENTIAL Output-SEQUENTIAL	TAPE- 7-track, 800 bits-per-inch, standard label, data conversion	TAPE- 7-track, 800 bits-per-inch, standard label, data conversion	1. Blocked output 2. Utility control statements in input stream. 3. Entire data set edited as one record group.

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

- The SYSUT1 DD Statement: defines the input sequential data set (OLDSET). The data set was originally written as the third data set (800 bits-per-inch) on a 7-track magnetic tape volume.

- The SYSUT2 DD Statement: defines the output sequential data set (NEWSET). The data set is written as the first or only data set on a 7-track magnetic tape volume. A density of 800 bits-per-inch and data conversion are specified for the write operation. The output records are blocked to reduce the amount of 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.

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

- The GENERATE Statement: (1) indicates that a maximum of three FIELD parameters is included in subsequent RECORD statements. (2) indicates that a maximum of 11 literal characters is included in subsequent FIELD parameters.

- The EXITS Statement: indicates that the specified user routines require control when SYSUT1 is opened and when SYSUT2 is closed.

- The LABELS Statement: indicates that labels will be included in the input stream.

- The RECORD Statement: controls the editing, as follows:

    1. Asterisks are placed in positions 1 to 10.
    2. Bytes 1 to 5 of the input record are converted from H-set BCD to EBCDIC mode and moved to positions 11 to 15.
    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 out as user trailer labels on SYSUT2.

Note: This example shows the relationship between the RECORD LABELS statement and the EXITS statement. The IEBGENER program will attempt to write out two records:

```
first label trailer
second label trailer
```

These will be written as user labels at close time of SYSUT2, but, before returning control to the system, the user routine ROUT2 can review these records and change them, if necessary.

```
//TAPETAPE JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=OLDSET,UNIT=2400-2,DISP=(OLD,KEEP),
//              VOLUME=SER=001234,LABEL=(3,SL),
//              DCB=(RECFM=F,LRECL=80,BLKSIZE=80,TRTCH=C)
//SYSUT2   DD   DSNAME=NEWSET,UNIT=2400-2,DISP=(NEW,PASS),
//              VOLUME=SER=001235,LABEL=(,SL),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000,TRTCH=C)
//SYSIN    DD   *
     GENERATE   MAXFLDS=3,MAXLITS=11
       RECORD   FIELD=(10,'**********',,1),FIELD=(5,1,HE,11),          C
              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
/*
```

IEBGENER Example 8. Copying and Editing an Input Sequential Data Set

IEBGENER Example 9

Operation	Data Set Organization	Input Device	Output Device	Comments
COPY - with editing	Input-SEQUENTIAL Output-SEQUENTIAL	DISK - 2311	DISK - 2311	1. Blocked output. 2. New record length specified for output data set. 3. Utility control statements in input stream. 4. Two record groups defined for editing.

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

- The SYSUT1 DD Statement: defines the input sequential data set (OLDSET). The logical record length of the input records is 100 bytes.

- The SYSUT2 DD Statement: defines the output sequential data set (OUTSET). Twenty tracks of primary storage space and ten tracks of secondary storage space are allocated for the data set on an IBM 2311 Disk Storage Drive. The logical record length of the output records is 80 bytes, and the output is blocked.

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

- The GENERATE Statement: (1) indicates that a maximum of 4 FIELD parameters is included in subsequent RECORD statements. (2) indicates that a maximum of 1 IDENT parameter appears in a subsequent RECORD statement.

- The EXITS Statement: 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 is defined as being the last record in the first group of records. The name FIRSTGRP appears in bytes 1 through 8 of the input record.

    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 (2nd 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 (2nd FIELD parameter).

Note: 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.

```
//DISKDISK JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=OLDSET,UNIT=2311,DISP=(OLD,KEEP),
//              VOLUME=SER=111112,DCB=(RECFM=F,LRECL=100,BLKSIZE=100)
//SYSUT2   DD   DSNAME=NEWSET,UNIT=2311,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),FIELD=(21,80,,60),              C
              FIELD=(59,1,,1)
GROUP2 RECORD   FIELD=(11,90,,70),FIELD=(69,1,,1)
/*
```

IEBGENER Example 9. Changing the Logical Record Length of a Data Set

IEBGENER Example 10

Operation	Data Set Organization	Input Device	Output Device	Comments
Copy - with editing	Input-SEQUENTIAL Output-SEQUENTIAL	TAPE-9-track, 800 bits-per-inch, standard and user labels	TAPE-9-track, 800 bits-per-inch, standard and user labels	1. Blocked output 2. Utility control statements in input stream. 3. Entire data set edited as one record group.

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

- The SYSUT1 DD Statement: defines the input sequential data set (OLDSET). The data set was originally written as the third data set (800 bits-per-inch) on a 9-track magnetic tape volume.

- The SYSUT2 DD Statement: defines the output sequential data set (NEWSET). The data set is written as the first or only data set on a 9-track magnetic tape volume. A density of 800 bits-per-inch is specified for the write operation. The output records are blocked to reduce the amount of 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.

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

- The GENERATE statement: (1) indicates that a maximum of three FIELD parameters is included in subsequent RECORD statements. (2) indicates that a maximum of 11 literal characters is included in subsequent FIELD parameters.

- The LABELS Statements: indicates that label records are included in the input stream.

- The RECORD Statement: controls the editing, as follows:

  1. Asterisks are placed in positions 1 to 10.

  2. Bytes 1 to 5 of the input record are converted from H-set BCD to EBCDIC mode and moved to positions 11 to 15.

  3. An equal sign is placed in byte 16.

  4. Unrelated (meaningless) data.

The second and third RECORD statements indicate that immediately following these statements there are three 80-bytes records (cards) that have to be written as user labels on the output data set. The first RECORD LABELS statement indicates that the following cards are to be treated as header labels. The second RECORD LABELS statement indicates that the following cards are to be treated as trailer labels.

```
//TAPETAPE JOB  09#660,SMITH
//         EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=OLDSET,UNIT=2400,DISP=(OLD,KEEP),
//              VOLUME=SER=001234,LABEL=(3,SUL),
//              DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSUT2   DD   DSNAME=NEWSET,UNIT=2400,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),FIELD=(5,1,HE,11),          C
                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
/*
```

IEBGENER Example 10. Copying and Editing an Input Sequential Data Set and Generating Labels From the Input Stream

Program Applications

The IEBCOMPR utility program compares two identically organized data sets at the logical record level. Data sets to be compared can be either sequential or partitioned.

The program can be used to:

- Verify a back-up copy of a sequential or partitioned data set.
- Verify portions of records within a sequential or partitioned data set.

User exits are provided at appropriate places for optional user routines that process user labels, handle error conditions, and modify source records. (Refer to the section "Exit Routine Linkage" for a discussion of linkage conventions applicable when user routines are provided.)

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

Comparing Sequential Data Sets: Two sequential data sets are considered "equal" if:

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

If these conditions are not met, an unequal comparison will result. If two corresponding 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 will terminate the job step unless a user routine is provided to handle error conditions.

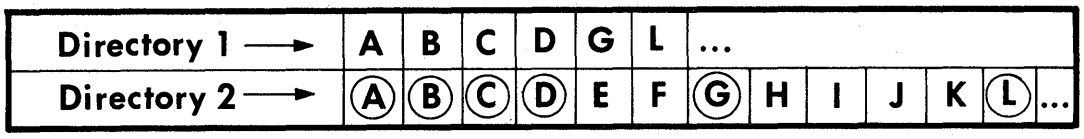
Comparing Partitioned Data Sets: Two partitioned data sets are considered equal if:

- 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 will result. If two corresponding 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 will cause processing to continue with the next member unless a user routine is provided to handle error conditions.

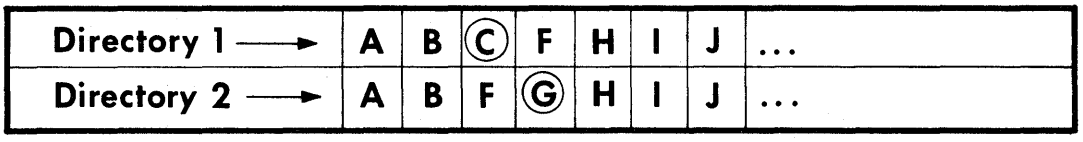
Note: Two partitioned data sets can be compared only if all the names in one or both of the directories have counterpart entries in the other directory. The comparison is made on members identified by these entries, and corresponding user data.

IEBCOMPR Figure 1 shows the directories of two partitioned data sets. The directory of partitioned data set 2 contains corresponding entries for all of the names in the directory of partitioned data set 1; therefore, the data sets can be compared.



IEBCOMPR Figure 1. Partitioned Directories -- Data Sets Can be Compared

IEBCOMPR Figure 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.



IEBCOMPR Figure 2. Partitioned Directories -- Data Sets Cannot be Compared

Verifying Back-Up Copies

The IEBCOMPR program can be used to check the results of a job step that has created a back-up copy of a sequential or partitioned data set. The IEBCOMPR program can compare fixed-length, variable-length, or undefined-length records from blocked or unblocked data sets or members.

Verifying Portions of Records

The IEBCOMPR program can be used to verify portions of records from two sequential or partitioned data sets. Since the program compares entire records, the user must provide routines to make equal those portions that he does not wish to verify; i.e., the undesired portions of two corresponding records must be made identical, or an unequal comparison will result.

Inputs and Outputs

IEBCOMPR Table 1 lists the major inputs to and outputs from the IEBCOMPR program.

IEBCOMPR Table 1. Data Sets Used (Input) and Produced (Output) by the IEBCOMPR Program

Inputs	Input Data Sets: Two input data sets are required by the IEBCOMPR program. The data sets, which contain the information to be compared, can be both sequential or both partitioned. Control Data Set: This data set contains utility control statements. The data set is required if the input data sets are partitioned, or if user routines are provided.
Output	Message Data Set: This data set contains informational messages (e.g., the contents of applicable utility control statements), the results of comparisons, and error messages, if applicable.

ADDITIONAL OUTPUTS

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

00 -- successful completion.

08 -- an "unequal" condition exists. Processing continues.

12 -- an unrecoverable error has occurred. The job step is terminated.

16 -- a user routine has passed a return code of 16 to the IEBCOMPR program. The job step is terminated.

Control

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

JOB CONTROL STATEMENTS

IEBCOMPR Table 2 shows the job control statements necessary for executing or invoking the IEBCOMPR program.

IEBCOMPR Table 2. Job Control Statements for the IEBCOMPR Program

Statement	Usage
JOB statement	This statement initiates the job.
EXEC statement	This statement specifies the program name (PGM=IEBCOMPR) or, if the job control statements for the IEBCOMPR program reside in a procedure library, the procedure name.
SYSPRINT DD statement	This statement defines a sequential message data set. The data set can be written onto a system output device (e.g., a printer), a magnetic tape volume, or a direct access volume. (This DD statement must be present for each execution or invocation of the IEBCOMPR program.)
SYSUT1 DD statement	This DD statement defines one of the input data sets to be compared. It can define a sequential data set on a card reader, a magnetic tape volume, or a direct access device. Or, it can define a partitioned data set on a direct access device.
SYSUT2 DD statement	This DD statement defines one of the input data sets to be compared. It can define a sequential data set on a card reader, a magnetic tape volume, or a direct access device. Or, it can define a partitioned data set on a direct access device.
SYSIN DD statement	This statement 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 also be defined as being a member within a library of partitioned members. The SYSIN DD statement is required, regardless of the operation.
Notes: The logical record lengths of the input data sets must be identical; otherwise unequal comparisons will result. The block sizes of the input data sets can differ; however, both of the block sizes must be multiples of the logical record length. The blocksize for the SYSPRINT (message) data set must be a multiple of 121. The blocksize for the SYSIN (control) data set must be a multiple of 80. Any blocking factor can be specified for these blocksizes. 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 sequential data sets written at different densities can be compared.	

UTILITY CONTROL STATEMENTS

The IEBCOMPR program is controlled by three utility control statements:

- The COMPARE statement.
- The EXITS statement.
- The LABELS statement.

The COMPARE statement is required if the EXITS or LABELS statement is used or if the input data sets are partitioned. The COMPARE statement, if present, must be the first utility statement. The EXITS statement is used if user routines are provided. The LABELS statement indicates the treatment of user labels by the IEBCOMPR program.

The COMPARE Statement

The COMPARE statement is used to indicate the type of data set organization.

Name	Operation	Operand
[name]	COMPARE	{TYPORG=PS} {TYPORG=PO}

TYPORG=PS
indicates that the input data sets are organized sequentially.

TYPORG=PO
indicates that the input data sets are partitioned.

If TYPORG is omitted, the input data sets are assumed to be sequential.

The EXITS Statement

The EXITS statement is used to identify exit routines supplied by the user. Exits to label processing routines will result in termination of the utility if partitioned data sets are being compared. Linkages to and from exit routines are discussed in Appendix A.

Name	Operation	Operand
[name]	EXITS	[INHDR=routinename] [INTLR=routinename] [ERROR=routinename] [PRECOMP=routinename]

INHDR=routinename
specifies the symbolic name of a routine that processes user input header labels.

INTLR=routinename
specifies the symbolic name of a routine that processes user input trailer labels.

For a more detailed discussion of the processing of user labels as data set descriptors, refer to "Appendix F: Utility Program Handling of User Labels."

ERROR=routinename
specifies the symbolic name of an error routine to be given control after each unequal comparison.

If the ERROR parameter is omitted and ten consecutive unequal comparisons occur while the IEBCOMPR program is comparing sequential data sets, processing is terminated. If the input data sets are partitioned, processing continues with the next member.

PRECOMP=routinename
specifies the symbolic name of a user routine that processes logical records (physical blocks in the case of VS or VBS type records longer than 32K bytes) from either or both of the input data sets before they are compared.

Note: If you code more than one valid EXITS statement (that is, free from syntax errors), the IEBCOMPR program will not issue a message but will simply treat the last EXITS statement as though it were the only EXITS statement in the SYSIN data set.

The LABELS Statement

The LABELS statement specifies whether or not user labels are to be treated by the IEBCOMPR program as data. For a detailed discussion of this option, refer to the section entitled "Processing User Labels as Data," in "Appendix F: Utility Program Handling of User Labels."

Name	Operation	Operand
[name]	LABELS	DATA={YES / NO / ALL / ONLY}

Note: If you code more than one valid LABELS statement (that is, free from syntax errors), the IEBCOMPR program will not issue a message but will simply treat the last LABELS statement as though it were the only LABELS statement in the SYSIN data set.

Using the Utility Control Statements

IEBCOMPR Table 3 shows the use of the utility control statements.

IEBCOMPR Table 3. Use of the COMPARE, EXITS, and LABELS Statements

TO	WITH	USE
Compare two sequential data sets	No user routines and processing of user labels as data	COMPARE TYPORG=PS LABELS DATA= YES ALL or: no control statements
	No user routines and no processing of user labels as data	COMPARE TYPORG=PS LABELS DATA=NO
	User routines and processing of user labels as data	COMPARE TYPORG=PS EXITS applicable routine name(s) and type(s)[1] LABELS DATA= YES ALL
	User routines and no processing of user labels as data	COMPARE TYPORG=PS EXITS applicable routine name(s) and type(s)[1] LABELS DATA=NO
Compare two partitioned data sets (User labels are invalid for partitioned data sets)	No user routines	COMPARE TYPORG=PO
	User routines	COMPARE TYPORG=PO EXITS [PRECOMP=routinename] [ERROR=routinename]

[1]One or more of the four EXITS statement operands can be coded. For example:

```
EXITS   ERROR=routinename, INHDR=routinename
```

Coding Utility Control Statements

Utility control statements are coded in columns 1 through 71. A statement that exceeds 71 characters may be continued on one or more additional cards.

Names: Names are not required for utility control statements. If they are used, they must begin in column 1. Names cannot be over eight characters long and must be followed by one or more blanks.

Operation: The operation field can begin in any column from 2 through 71. An operation field may be interrupted at column 71 and continued on the next card. When an operation field is interrupted, a nonblank character must be placed in column 72. The continued portion must start in any column from 4 through 16 of the next card. (Restriction: When an operation field ends in column 70 or 71, it cannot be followed by an operand or comment.)

Operands (Keywords and Parameters): Operands must be preceded by an operation field. They must begin on the same card that the operation field ends on. One or more blanks must separate the operand from the

operation field. When an operand ends in column 71, then column 72 must be left blank. If an operand is interrupted at column 71 and is to be continued on the next card, a nonblank character must be placed in column 72. An operand can also be interrupted at a comma (unless the comma is part of a literal). Whenever an operand is interrupted at a comma, column 72 may be either blank or nonblank. In all cases, the continuation must start in any column from 4 through 16 of the next card.

Comments: Comments must be preceded by an operand or an operation field. They must begin on the same card that the last operand or the operation field ends on. (Exception: When an operand is interrupted at a comma, the remaining columns through 71 may contain a comment. A blank must separate the comma from the comment, however.) A comment cannot separate an operation field from an operand.

Comments may be interrupted and continued on the next card by placing a nonblank character in column 72. Comments do not have to be interrupted at column 71; blanks are acceptable in column 71 as part of the comment. The continued portion must start in any column from 4 through 16 of the next card.

IEBCOMPR Examples

The following examples illustrate some of the uses of the IEBCOMPR program.

IEBCOMPR Example 1

Operation	Data set Organization	Input Device 1	Input Device 2	Comments
COMPARE	Inputs-SEQUENTIAL	TAPE - 9-track, unlabeled, 800 bits-per-inch density	TAPE - 9-track, unlabeled, 800 bits-per-inch density	1. No user routines. 2. Blocked inputs.

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

- The SYSUT1 DD Statement: defines an input data set. The data set resides on an unlabeled, 9-track magnetic tape volume. The blocked data set was originally written at 800 bits-per-inch density.
- The SYSUT2 DD Statement: defines an input data set. The data set resides on an unlabeled, 9-track magnetic tape volume. The blocked data set was originally written at 800 bits-per-inch density.
- The SYSIN DD Statement: defines a dummy data set. (Since no user routines are provided and the input is sequential, no utility control statements are required.)

```
//TAPETAPE JOB  09#660,SMITH
//         EXEC PGM=IEBCOMPR
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=2400,LABEL=(,NL),DISP=(OLD,KEEP),
//             VOLUME=SER=001234,DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//SYSUT2   DD   UNIT=2400,LABEL=(,NL),DISP=(OLD,KEEP),
//             VOLUME=SER=001235,DCB=(RECFM=FB,LRECL=80,BLKSIZE=1040)
//SYSIN    DD   DUMMY
/*
```

IEBCOMPR Example 1. Comparing Sequential Data Sets -- Inputs on 9-Track Tape

IEBCOMPR Example 2

Operation	Data set Organization	Input Device 1	Input Device 2	Comments
COMPARE	Inputs-SEQUENTIAL	TAPE - 7-track, standard and user labels, 800 bits-per-inch density, data conversion	TAPE - 7-track, standard and user labels, 800 bits-per-inch density, data conversion	1. No user routines. 2. Blocked inputs.

In this example two sequential data sets are to be compared.

the second data set on a labeled, 7-track magnetic tape volume. The blocked data set was originally written at 800 bits-per-inch density with the data converter on.

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

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

IEBCOMPR Example 2. Comparing User Header Labels on Sequential Data Sets -- Inputs on 7-Track Tape

IEBCOMPR Example 3

Operation	Data set Organization	Input Device 1	Input Device 2	Comments
COMPARE	Inputs-SEQUENTIAL	TAPE - 7-track, standard and user labels, 556 bits-per-inch density, data conversion	TAPE - 9-track, standard and user labels, 800 bits-per-inch density	1. User routines. 2. Blocked inputs. 3. Different density tapes.

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

- The SYSUT1 DD Statement: defines an input data set. The data set resides as the first or only data set on a labeled, 7-track magnetic tape volume. The blocked data set was originally written at 556 bits-per-inch density with the data converter on.
- The SYSUT2 DD Statement: defines an input data set. The data set resides as the first or only data set on a labeled, 9-track magnetic tape volume. The blocked data set was originally written at 800 bits-per-inch density.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream.
- The COMPARE Statement: specifies that the input data sets are sequentially organized.
- The EXITS Statement: identifies the names of user input header label and user input trailer label processing routines.
- The LABELS Statement: specifies that the comparing of user input head or trailer labels as data is not required.

```
//TAPETAPE JOB  09#660,SMITH
//         EXEC PGM=IEBCOMPR
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=SET1,UNIT=2400-2,LABEL=(,SUL),DISP=(OLD,KEEP),
//              VOLUME=SER=001234,DCB=(DEN=1,RECFM=FB,LRECL=80,
//              BLKSIZE=320,TRTCH=C)
//SYSUT2   DD   DSNAME=SET2,UNIT=2400,LABEL=(,SUL),DISP=(OLD,KEEP),
//              VOLUME=SER=001235,DCB=(RECFM=FB,LRECL=80,BLKSIZE=640)
//SYSIN    DD   *
      COMPARE   TYPORG=PS
        EXITS   INHDR=HDRS,INTLR=TLRS
       LABELS   DATA=NO
/*
```

IEBCOMPR Example 3. Comparing Sequential Data Sets -- Inputs on 7- and 9-Track Tapes

IEBCOMPR Example 4

Operation	Data set Organization	Input Device 1	Input Device 2	Comments
COMPARE	Inputs-SEQUENTIAL	CARD READER	TAPE- 9-track, unlabeled, 800 bits-per-inch density	1. No user routines. 2. Blocked (tape) input.

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

- The SYSIN DD Statement: defines a dummy control data set. (Since no user routines are provided and the input data sets are sequential, no utility control statements are required.)
- The SYSUT2 DD Statement: defines an input data set. The data set resides on an unlabeled, 9-track magnetic tape volume. The blocked data set was originally written at 800 bits-per-inch density.
- The SYSUT1 DD Statement: defines an input data set (card input).

```
//CARDTAPE JOB  09#660,SMITH
//         EXEC PGM=IEBCOMPR
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DUMMY
//SYSUT2   DD   UNIT=2400,VOLUME=SER=001234,LABEL=(,NL),
//              DISP=(OLD,KEEP),DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//SYSUT1   DD   DATA
           .
           .
           input card data set
           .
           .
/*
```

IEBCOMPR Example 4. Comparing Sequential Data Sets -- Inputs on Tape and Card Reader

IEBCOMPR Example 5

Operation	Data set Organization	Input Device 1	Input Device 2	Comments
COMPARE	Inputs-PARTITIONED	DISK - 2311	DISK - 2311	1. No user routines. 2. Blocked inputs.

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

- The SYSUT1 DD Statement: defines an input partitioned data set. The blocked data set resides on an IBM 2311 Disk Storage Drive.
- The SYSUT2 DD Statement: defines an input partitioned data set. The blocked data set resides on an IBM 2311 Disk Storage Drive.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream. The data set consists of one utility control statement.

```
//DISKDISK JOB  09#660,SMITH
//         EXEC PGM=IEBCOMPR
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=PDSSET,UNIT=2311,DISP=(OLD,KEEP),
//              VOLUME=SER=111112,DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//SYSUT2   DD   DSNAME=PDSSET,UNIT=2311,DISP=(OLD,KEEP),
//              VOLUME=SER=111113,DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//SYSIN    DD   *
      COMPARE   TYPORG=PO
/*
```

IEBCOMPR Example 5. Comparing Partitioned Data Sets -- Inputs on Disk

IEBCOMPR Example 6

Operation	Data set Organization	Input Device 1	Input Device 2	Comments
COPY - (Program IEBGENER) & COMPARE	Inputs-SEQUENTIAL	TAPE - 9-track, standard label, 800 bits-per-inch density	TAPE - 9-TRACK, standard label, 800 bits-per-inch density	1. No user routines. 2. Blocked input. 3. 2 job steps -- data sets passed to 2nd job step.

This example is a 2-step example in which a sequential data set is to be copied and compared; the first step is to pass the original and the copied data sets to the second job step. The second job step compares the two data sets. The second job step (i.e., the IEBCOMPR job step) is explained below:

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

```
//TAPETAPE JOB  09#660,SMITH
//STEPA    EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=COPYSET,UNIT=2400,DISP=(OLD,PASS),LABEL=(,SL),
//              VOLUME=SER=001234,DCB=(RECFM=FB,LRECL=80,BLKSIZE=640)
//SYSUT2   DD   DSNAME=COPYSET,UNIT=2400,DISP=(,PASS),LABEL=(,SL),
//              VOLUME=SER=001235,DCB=(RECFM=FB,LRECL=80,BLKSIZE=640)
//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
/*
```

IEBCOMPR Example 6. Comparing Passed (Sequential) Data Sets -- Identical Input Data Set Names

IEBCOMPR Example 7

Operation	Data Set Organization	Input Device 1	Input Device 2	Comments
COPY- (Program IEBCOPY) & COMPARE	Inputs-PARTITIONED	DISK - 2311	DISK - 2311	1. User routine provided. 2. Blocked input. 3. 2-step job. -- Data sets passed to 2nd job step.

This example is a 2-step example. The first job step (IEBCOPY) is to copy a partitioned data set and pass the original and copied data sets to the second job step. The second job step (IEBCOMPR) compares the two data sets. The IEBCOMPR job step is explained below.

- The SYSUT1 DD Statement: defines a blocked, input data set that is passed from the preceding job step. The data set resides on an IBM 2311 Disk Storage Drive.
- The SYSUT2 DD Statement: defines a blocked, input data set that is passed from the preceding job step. The data set resides on an IBM 2311 Disk Storage Drive.
- The SYSIN DD Statement: defines the control data set, which contains a COMPARE statement and an EXITS statement. (The COMPARE statement specifies partitioned organization and the EXITS statement identifies the user routine.)

Note: Since the input data set names are not identical, the data sets can be retrieved by their data set names (see the SYSUT1 and SYSUT2 DD statements).

```
//DISKDISK JOB  09#660,SMITH
//STEPA    EXEC PGM=IEBCOPY
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=OLDSET,UNIT=2311,DISP=(OLD,PASS),
//              VOLUME=SER=111112,DCB=(RECFM=FB,LRECL=80,
//              BLKSIZE=640)
//SYSUT2   DD   DSNAME=NEWMEMS,UNIT=2311,DISP=(,PASS),
//              VOLUME=SER=111113,SPACE=(TRK,(10,5,5)),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=640)
//SYSIN    DD   *
         COPY   TYPCOPY=I,MAXNAME=5
       MEMBER   NAME=(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
/*
```

IEBCOMPR Example 7. Comparing Passed (Partitioned) Data Sets -- User Routine Provided

Program Applications

The IEBPTPCH program prints or punches 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 as follows:

- Each logical record begins on a new printed line or punched card.
- Each printed line consists of 12 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.

Other formats (i.e., user formats) can be specified, provided that no output record exceeds the output device capacity.

The IEBPTPCH program provides optional editing facilities with each application of the program. In addition, user exits are provided at appropriate places for user routines that process labels and/or manipulate input or output records.

The program 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 in its Entirety

The IEBPTPCH program can 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 bit configurations (alphameric). For a print operation, packed decimal data should be converted to unpacked decimal or hexadecimal mode to ensure that all characters are printable.

Standard Print Operation: For a standard print operation, each logical record is printed in groups of 8 characters, as shown below:

Sequential or Partitioned Input	Printed Output
ABCDEFGHIJKLMNOPQRSTUVWX...	ABCDEFGH IJKLMNOP QRSTUVWX ... 2 blanks between each group of characters

In this manner, up to 96 characters can be included on a printed line. (An edited output can be produced to omit the blank delimiters and print up to 120 characters per line. Refer to the IEBPTPCH examples for details.)

Standard Punch Operation: 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 73-80 of the punched cards.

Printing or Punching Selected Members

The IEBPTPCH program can be used to print or punch selected members of a partitioned data set. Utility control statements are used to specify those members that are to be printed or punched.

Printing or Punching Selected Records

The IEBPTPCH program 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.

Printing or Punching a Partitioned Directory

The IEBPTPCH program can be used to print or punch the contents of a partitioned directory.

Standard Print Operation: Each directory block is printed in groups of 8 characters. If the directory is printed in hexadecimal representation, the first four printed characters (two bytes) of each directory block indicate the total number of used bytes in that block. For details of the format of the directory, refer to the publication IBM System/360 Operating System: System Control Blocks, GC28-6628.

Standard Punch Operation: Data from a directory block is punched in contiguous columns in the punched cards representing that block.

Printing or Punching an Edited Data Set

The IEBPTPCH program can 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.
- 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 to unpacked decimal or from alphameric to hexadecimal representation.

Inputs and Outputs

IEBPTPCH Table 1 lists the major inputs to and outputs from the IEBPTPCH program.

IEBPTPCH Table 1. Data Sets Used (Input) and Produced (Output) by the IEBPTPCH Program

Inputs	Input Data Set: This data set contains the data that is to be printed or punched. The input data set can be either sequential or partitioned. Control Data Set: This data set contains utility control statements. The data set is required for each application of the program.
Outputs	Output Data Set: This data set is the printed or punched data set. Message Data Set: This data set contains informational messages (e.g., the contents of control statements) and error messages, if applicable.

ADDITIONAL OUTPUTS

The IEBPTPCH program provides a return code to indicate the results of program execution. The return codes and their interpretations are as follows:

00 -- successful completion
08 -- a member specified for printing does not exist in the input data set. Processing continues with the next member.
12 -- an unrecoverable error has occurred or a user routine has passed a return code of 12 to the IEBPTPCH program. The job step is terminated.
16 -- a user routine has passed a return code of 16 to the IEBPTPCH program. The job step is terminated.

Control

The IEBPTPCH program 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 the IEBPTPCH program.

JOB CONTROL STATEMENTS

IEBPTPCH Table 2 shows the job control statements necessary for executing or invoking the IEBPTPCH program.

IEBPTPCH Table 2. Job Control Statements for the IEBPTPCH Program

Statement	Usage
JOB statement	This statement initiates the job step.
EXEC statement	This statement specifies the program name (PGM=IEBPTPCH) or, if the job control statements for the IEBPTPCH program reside in a procedure library, the procedure name.
SYSPRINT DD statement	This statement 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. This DD statement must be present for each application of the IEBPTPCH program.
SYSUT1 DD statement	This statement defines the input data set. It can define a sequential data set on a card reader, a magnetic tape volume, or a direct access device; or it can define a partitioned data set on a direct access device.
SYSUT2 DD statement	This statement defines the output (printed or punched) data set. The SYSUT2 data set cannot be blocked. The BLKSIZE parameter in the DCB information specifies the number of characters to be written per printed line or per punched card (this count includes a control character). The default values for this parameter are 120 characters per printed line and 80 characters per punched card.
SYSIN DD statement	This statement defines the control data set. The control data set normally resides in the input stream; however, it can also be defined as being a member of a library of partitioned members. The SYSIN DD statement is required for each application of the program.

Notes: The input data set can contain fixed-length, variable-length, undefined-length, or variable spanned records.

The blocksize for the SYSPRINT (message) data set must be a multiple of 121. The blocksize for the SYSIN (control) data set must be a multiple of 80. Any blocking factor can be specified for these blocksizes.

A partitioned directory to be printed must be defined as a sequential data set. Refer to the discussion of the PRINT utility control statement for details.

Both the output data set and the message data set can reside on the system output device (this assumes that the system output device is a printer).

If the logical record length of the input records is such that the output would exceed the output record length, the utility will break the record up into multiple lines or cards.

Refer to the IEBPTPCH examples for typical uses of the job control statements.

UTILITY CONTROL STATEMENTS

The IEBPTPCH program is controlled by combinations of the following utility control statements:

- The PRINT or PUNCH statement.
- The TITLE statement.
- The EXITS statement.
- The MEMBER statement.
- The RECORD statement.
- The LABELS statement.

The control statements are included in the control data set as required. IEBPTPCH Table 3 shows the order of inclusion and the uses of the control statements.

IEBPTPCH Table 3. Utility Control Statements for the IEBPTPCH Program

This Statement is included	If
PRINT	Data is to be printed. The PRINT statement or the PUNCH statement (for a punch operation) appears first in the control data set.
PUNCH	Data is to be punched. The PUNCH statement or the PRINT statement (for a print operation) appears first in the control data set.
TITLE (optional)	A title is to precede the printed or punched data. Two TITLE statements can be included per execution of the program. The TITLE statements(s) must follow immediately the PRINT or PUNCH statement in the control data set.
EXITS (optional)	User routines are provided. This statement must appear immediately after the TITLE statement(s) or after the PRINT or PUNCH statement.
MEMBER (optional)	The input is partitioned and a selected member (identified by the MEMBER statement) is to be printed or punched.
RECORD (optional)	Editing is to be performed; i.e., records are to be printed or punched to non-standard specifications.
LABELS (optional)	Specifies whether or not user labels are to be treated as data.

Note: Any number of MEMBER and/or RECORD statements can be included in a job step.

Coding Utility Control Statements

Utility control statements are coded in columns 1 through 71. A statement that exceeds 71 characters may be continued on one or more additional cards.

Names: Names are not required for utility control statements. If they are used, they must begin in column 1. Names cannot be over eight characters long and must be followed by one or more blanks.

Operation: The operation field can begin in any column from 2 through 71. An operation field may be interrupted at column 71 and continued on the next card. When an operation field is interrupted, a nonblank character must be placed in column 72. The continued portion must start in any column from 4 through 16 of the next card. (Restriction: When an operation field ends in column 70 or 71, it cannot be followed by an operand or comment.)

Operands (Keywords and Parameters): Operands must be preceded by an operation field. They must begin on the same card that the operation field ends on. One or more blanks must separate the operand from the operation field. When an operand ends in column 71, then column 72 must be left blank. If an operand is interrupted at column 71 and is to be continued on the next card, a nonblank character must be placed in column 72. An operand can also be interrupted at a comma (unless the comma is part of a literal). Whenever an operand is interrupted at a comma, column 72 may be either blank or nonblank. In all cases, the continuation must start in any column from 4 through 16 of the next card.

Comments: Comments must be preceded by an operand or operation field. They must begin on the same card that the last operand or the operation field ends on. (Exception: When an operand is interrupted at a comma, the remaining columns through 71 may contain a comment. A blank must separate the comma from the comment, however.) A comment cannot separate an operation field from an operand.

Comments may be interrupted and continued on the next card by placing a nonblank character in column 72. Comments do not have to be interrupted at column 71; blanks are acceptable in column 71 as part of the comment. The continued portion must start in any column from 4 through 16 of the next card.

The Print or Punch Statement

The PRINT or PUNCH statement is used to initiate the utility operation.

Name	Operation	Operand	
[name]	PRINT PUNCH	[PREFORM=A] [PREFORM=M] {TYPORG=PS} {TYPORG=PO} [TOTCONV=XE] [TOTCONV=PZ] [CNTRL=n] [STOPAFT=n] [SKIP=n] [MAXNAME=n] [MAXFLDS=n] [MAXGPS=n] [MAXLITS=n]	Applicable to a PRINT or PUNCH operation.
		[INITPG=n] [MAXLINE=n]	Applicable only to a PRINT operation.
		[CDSEQ=n] [CDINCR=n]	Applicable only to a PUNCH operation.

PREFORM=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 will (1) for punched output, duplicate the ASA character on each output card of the record, or (2) for printed output, use the ASA character for printing the first line, with a single space character on all subsequent lines of the record.

During a printing operation, the control characters are used to control the printer carriage; i.e., to control spacing, number of lines per page, and page ejection. During a punching operation, the control characters are used to select a stacker. If an error is discovered, the printing or punching operation is terminated.

PREFORM=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 will (1) for punched output, duplicate the machine control character on each output card of the record, or (2) for printed output, print 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.

During a printing operation, the control characters are used to control the printer carriage; i.e., to control spacing, number of lines per page, and page ejection. During a punching operation, the control characters are used to select a stacker. If an error is uncovered, the operation is terminated.

Notes: The PREFORM parameter must not be used for printing or punching data sets with VS or VBS type records longer than 32K bytes.

If the PREFORM operand is used, then, except for syntax checking purposes, any additional PRINT or PUNCH operands and all other control statements except LABELS statements are ignored.

TYPORG=PS
specifies that the input data set is organized sequentially.

TYPORG=PO
specifies that the input data set is partitioned.

If TYPORG is omitted, PS is assumed.

TOTCONV=XE
specifies that data is to be printed or punched in 2-character-per-byte hexadecimal representation, e.g., C3 40 F4 F6.

If TOTCONV=XE is not specified, data will be printed or punched in 1-character-per-byte alphameric representation. The above example would appear as C 46. The TOTCONV operand is overridden by any user specifications (RECORD statements) that pertain to the same data.

TOTCONV=PZ
specifies that data in packed decimal mode is to be converted to unpacked decimal mode.

If TOTCONV is omitted, data is not converted. The TOTCONV operand is overridden by any user specifications (RECORD statements) that pertain to the same data.

CNTRL=n
specifies a control character for the output device. For a printer the number indicates line spacing, as follows:

1 -- single spacing
2 -- double spacing
3 -- triple spacing

For a card punch the number selects the stacker:

1 -- first stacker
2 -- second stacker

If CNTRL is omitted, 1 is assumed.

STOPAFT=n
Sequential data sets: specifies the number of logical records (physical blocks in the case of VS or VBS type records longer than 32K bytes) to be printed or punched (n must not exceed 32,767).

Partitioned data sets: specifies the number of logical records (physical blocks in the case of VS or VBS type records longer than 32K bytes) to be printed or punched in each member to be processed (n 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.

SKIP=n
specifies that every "nth" record (physical block in the case of VS or VBS type records longer than 32K bytes) is to be printed or punched.

If SKIP is omitted, successive logical records are printed or punched.

MAXNAME=n
specifies a number no less than the total number of subsequent MEMBER statements.

If MAXNAME is omitted and there are one or more MEMBER statements, the PRINT or PUNCH request is terminated.

MAXFLDS=n
specifies a number no less than the total number of FIELD parameters appearing in subsequent RECORD statements.

If MAXFLDS is omitted and there are one or more FIELD parameters, the PRINT or PUNCH request is terminated.

MAXGPS=n
specifies a number no less than the total number of IDENT parameters appearing in subsequent RECORD statements.

If MAXGPS is omitted and there are one or more IDENT parameters, the PRINT or PUNCH request is terminated.

MAXLITS=n
specifies a number no less than the total number of characters contained in the IDENT literals of subsequent RECORD statements.

If MAXLITS is omitted and there are one or more literals, the PRINT or PUNCH request is terminated.

INITPG=n (PRINT operation only)
specifies the initial page number. Printed pages are numbered sequentially thereafter.

If INITPG is omitted, 1 is assumed.

MAXLINE=n (PRINT operation only)
specifies the maximum number of lines to a printed page. Spaces, titles, and subtitles are included in this number.

If MAXLINE is omitted, 60 is assumed.

CDSEQ=n (PUNCH operation only)
specifies the initial sequence number of a deck of punched cards. This value can be up to 8 digits and occupies columns 73 to 80. Sequence numbering is initialized for each member of a partitioned data set.

If CDSEQ is omitted, the cards are not numbered. If n is omitted, 00000000 is assumed as a starting sequence number.

CDINCR=n (PUNCH operation only)
specifies the increment to be used in generating sequence numbers.

If CDINCR is omitted and CDSEQ is coded, 10 is assumed as an increment value for sequence numbering.

The TITLE Statement

The TITLE statement is used to request title and subtitle records. A first TITLE statement defines the title and a second defines the subtitle.

Name	Operation	Operand
[name]	TITLE	ITEM=('title'[,output-location])

ITEM=('title'[,output-location])
specifies title or subtitle information.

'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 is to be placed in the output record. If no output location is specified, 1 is assumed.

The 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.

Name	Operation	Operand
[name]	EXITS	[INHDR=routinename] [INTLR=routinename] [INREC=routinename] [OUTREC=routinename]

INHDR=routinename
specifies the symbolic name of a routine that processes user input header labels.

INTLR=routinename
specifies the symbolic name of a routine that processes user input trailer labels.

INREC=routinename
specifies the symbolic name of a routine that manipulates each logical record (physical block in the case of VS or VBS type records longer than 32K bytes) before it is processed.

OUTREC=routinename
specifies the symbolic name of a routine that manipulates each logical record (physical block in the case of VS or VBS type records longer than 32K bytes) before it is printed or punched. When standard specifications are used, this exit is not available.

The 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.

If no MEMBER statement appears, and a partitioned data set is being processed, all members of the data set are printed or punched.

Name	Operation	Operand
[name]	MEMBER	{NAME=membername {NAME=aliasname }

NAME=membername or NAME=aliasname
identifies a member by its member name or its alias name.

The RECORD Statement

The RECORD statement is used to define a group of records (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. A RECORD statement can contain one IDENT parameter and any number of FIELD parameters.

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 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.

Name	Operation	Operand
[name]	RECORD	[IDENT=(length,'name',input-location)] [FIELD=(length,[input-location],[conversion], [[output-location])...]

IDENT=(length,'name',input-location)
identifies the last record of the record group to which the FIELD parameters apply.

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.

length: specifies the length (in bytes) of the field that contains the identifying name in the input records. The length cannot exceed 8 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.

FIELD=(length,[input-location],[conversion],[output-location])...
specifies the field processing and editing information.

length: specifies the length (in bytes) of the input field to be processed. This length cannot exceed 120 bytes.

input-location: specifies the starting byte of the input field to be processed. If no starting byte is specified, 1 is assumed.

Conversion: specifies a 2-byte code that indicates the type of conversion to be performed on this field before it is printed or punched, as follows:

Code	Conversion	Output Length (Where L is the Input Length)
PZ	Packed to unpacked decimal mode	2L-1
XE	Alphameric to hexadecimal representation	2L

If no conversion is specified, the field is moved to the output area without change.

output-location: specifies the starting location of this field in the output records. If no starting location is specified, 1 is assumed. Unspecified fields in the output records appear as blanks in the printed or punched output.

The LABELS Statement

The LABELS statement specifies whether or not the IEBPTPCH program should treat user labels as data. For a detailed discussion of this option, refer to the section entitled "Processing User Labels as Data," in "Appendix F: Utility Program Handling of User Labels."

Name	Operation	Operand
[name]	LABELS	DATA={YES / NO / ALL / ONLY}

IEBPTPCH Examples

The following examples illustrate some of the uses of the IEBPTPCH program.

IEBPTPCH Example 1

Operation	Data Set Organization	Input Device	Output Device	Comments
PRINT	Input-SEQUENTIAL	TAPE - 9-track, unlabeled, 800 bits-per-inch density	System Output device (PRINTER assumed)	1. Standard specifications. 2. Conversion to hexadecimal representation.

In this example, a sequential data set is to be printed according to standard specifications. The input data set resides on a 9-track magnetic tape volume, originally written at a density of 800 bits per inch. The printed output is to be converted to hexadecimal.

- The SYSUT1 DD Statement: defines the input data set. The data set contains U-type records, no record being larger than 2000 bytes.
- The SYSUT2 DD Statement: defines the output (printed) data set. The data set resides on the system output device (printer assumed). Each printed line contains up to 12 groups (8 characters each) of hexadecimal information. Each record begins a new line of printed output; i.e., no two records appear on the same line of printed output.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream. The control data set contains the PRINT and TITLE utility control statements.
- The PRINT Statement: initiates the print operation and specifies conversion from alphameric to hexadecimal representation.
- The TITLE Statement: specifies a title to be placed beginning in column 10 of the printed output. The title is not converted to hexadecimal.

```
//PRINT    JOB  09#660,SMITH
//         EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=2400,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)
/*
```

IEBPTPCH Example 1. Printing a Sequential Data Set With Standard Specifications

IEBPTPCH Example 2

Operation	Data Set Organization	Input Device	Output Device	Comments
PUNCH	Input-SEQUENTIAL	TAPE - 7-track, unlabeled, 556 bits-per-inch density, data converter on	CARD PUNCH - 2540-2	1. Standard specifications. 2. Conversion to hexadecimal representation.

In this example, a sequential data set is to be punched according to standard specifications. The input data set resides on a 7-track magnetic tape volume, originally written at a density of 556 bits-per-inch. The punched output is converted to hexadecimal.

- The SYSUT1 DD Statement: defines the input data set. The data set contains 80-byte fixed-length, blocked records.
- The SYSUT2 DD Statement: defines the output (punched) data set. The data set is to be punched by an IBM 2540-2 Card Read Punch (punch feed). Each record from the input data set is represented by two punched cards.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream. The control data set contains the PUNCH and TITLE utility control statements.
- The PUNCH Statement: initiates the punch operation and specifies conversion from alphameric to hexadecimal representation.
- The TITLE Statement: specifies a title to be placed beginning in column 10. The title is not converted to hexadecimal.

```
//PUNCHSET JOB  09#660,SMITH
//         EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=INSET,UNIT=2400-2,LABEL=(,NL),
//              VOLUME=SER=001234,DISP=(OLD,KEEP),
//              DCB=(DEN=1,RECFM=FB,LRECL=80,BLKSIZE=2000,TRTCH=C)
//SYSUT2   DD   UNIT=2540-2
//SYSIN    DD   *
         PUNCH  TOTCONV=XE
         TITLE  ITEM=('PUNCH SEQ DATA SET WITH CONV TO HEX',10)
/*
```

IEBPTPCH Example 2. Punching a Sequential Data Set With Standard Specifications

IEBPTPCH Example 3

Operation	Data Set Organization	Input Device	Output Device	Comments
PRINT	Input-PARTITIONED	DRUM - 2301	System Output Device (PRINTER assumed)	1. Standard specifications. 2. Conversion to hexadecimal representation. 3. 10 records from each member are printed.

In this example, a partitioned data set (10 records from each member) is to be printed according to standard specifications. The input data set resides on an IBM 2301 Drum Storage Device. The printed output is converted to hexadecimal.

- The SYSUT1 DD Statement: defines the input data set. The data set contains U-type records, no record being larger than 3625 bytes.
- The SYSUT2 DD Statement: defines the output data set on the system output device (printer assumed). Each printed line contains up to 12 groups (8 characters each) of hexadecimal information. Each record begins a new line of printed output; i.e., no two records appear on the same line of printed output. The size of the record determines how many lines of printed output are required per record.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream. The control data set contains the PRINT and TITLE utility control statements.
- The PRINT Statement: initiates the print operation, specifies conversion from alphameric to hexadecimal representation, indicates that the input data set is partitioned, and specifies that 10 records from each member be printed.
- The TITLE Statement: specifies a title to be placed beginning in column 20 of the printed output. The title is not converted to hexadecimal.

```
//PRINTPDS JOB  09#660,SMITH
//         EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=PDS,UNIT=2301,DISP=(OLD,KEEP),
//              VOLUME=SER=111112,DCB=(RECFM=U,BLKSIZE=3625)
//SYSUT2   DD   SYSOUT=A
//SYSIN    DD   *
        PRINT   TOTCONV=XE,TYPORG=PO,STOPAFT=10
        TITLE   ITEM=('PRINT PDS - 10 RECS EACH MEM',20)
/*
```

IEBPTPCH Example 3. Printing a Partitioned Data Set With Standard Specifications

IEBPTPCH Example 4

Operation	Data Set Organization	Input Device	Output Device	Comments
PRINT	Input-PARTITIONED	DISK - 2311	System Output Device (PRINTER assumed)	1. Standard specifications. 2. Conversion to hexadecimal representation. 3. Two members are printed.

In this example, two partitioned members are to be printed according to standard specifications. The input data set resides on an IBM 2311 Disk Storage Drive. The printed output is to be converted to hexadecimal.

- The SYSUT1 DD Statement: defines the input data set. The data set contains 80-byte, fixed-length records.
- The SYSUT2 DD Statement: defines the output data set on the system output device (printer assumed). Each printed line contains up to 12 groups (8 characters each) of hexadecimal information. Each record begins a new line of printed output; i.e., no two records appear on the same line of printed output.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream. The control data set contains PRINT, TITLE, and MEMBER utility control statements.
- The PRINT Statement: 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.
- The TITLE Statement: specifies a title to be placed beginning in column 10 of the printed output. The title is not converted to hexadecimal.
- The MEMBER Statements: specify the member names of the members to be printed.

```
//PRNTMEMS JOB  09#660,SMITH
//         EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=PDS,UNIT=2311,DISP=(OLD,KEEP),
//              VOLUME=SER=111112,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//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
/*
```

IEBPTPCH Example 4. Printing Two Partitioned Members With Standard Specifications

IEBPTPCH Example 5

Operation	Data Set Organization	Input Device	Output Device	Comments
PRINT	Input-SEQUENTIAL	TAPE 9-track, standard label, 800 bits-per-inch density	System Output Device (PRINTER assumed)	1. User (nonstandard specifications. 2. Input data set is 2nd data set on tape volume.

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 9-track magnetic tape volume. The data set was originally written at a density of 800 bits-per-inch.

- The SYSUT1 DD Statement: defines the input data set. The data set contains 80-byte, fixed-length blocked records.
- The SYSUT2 DD Statement: defines the output data set on the system output device (printer assumed). Each printed line contains 80 contiguous characters (1 record) of information.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream. The control data set contains the PRINT, RECORD, EXITS and LABELS utility control statements.
- The PRINT Statement: initiates the print operation and indicates that one FIELD parameter is included in a subsequent RECORD statement.
- The RECORD Statement: indicates that each input record is to be processed in its entirety (80 bytes). Each input record is printed in columns 1-80 on the printer.
- The LABELS Statement: specifies that user header and trailer labels are to be printed according to the return code issued by the user exits.
- The EXITS Statement: indicates that exits will be taken to user header and trailer label processing routines when these labels are encountered on the SYSUT1 data set.

```
//PTNONSTD JOB  09#660,SMITH
//         EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=SEQSET,UNIT=2400,LABEL=(2,SUL),
//              DISP=(OLD,KEEP),VOLUME=SER=001234,
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//SYSUT2   DD   SYSOUT=A
//SYSIN    DD   *
        PRINT   MAXFLDS=1
       RECORD   FIELD=(80)
       LABELS   DATA=YES
        EXITS   INHDR=HDRIN,INTLR=TRLIN
/*
```

IEBPTPCH Example 5. Printing a Sequential Data Set With User Specifications

IEBPTPCH Example 6

Operation	Data Set Organization	Input Device	Output Device	Comments
PUNCH	Input-SEQUENTIAL	DISK - 2311	CARD READ PUNCH 2540-2	1. User (nonstandard) specifications. 2. Sequence numbers assigned and punched.

In this example, a sequential data set is to be punched according to user specifications. The input data set resides on an IBM 2311 Disk Storage Drive.

- The SYSUT1 DD Statement: defines the input data set. The data set contains 80-byte, fixed-length blocked records.
- The SYSUT2 DD Statement: defines the output (punched) data set. The data set is to be punched by an IBM 2540-2 Card Read Punch (Punch feed). Each record from the input data set is represented by one punched card.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream. The control data set contains the PUNCH, RECORD, and LABELS utility control statements.
- The PUNCH Statements: 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 73-80 of the output records.
- The RECORD Statement: indicates that bytes 1-72 of the input records are to be punched. Bytes 73-80 of the input records are replaced by the new sequence numbers in the output card deck.
- The LABELS Statement: specifies that only user header labels are to be punched.

```
//PHSEQNO  JOB  09#660,SMITH
//         EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=SEQSET,UNIT=2311,
//              VOLUME=SER=111112,DISP=(OLD,KEEP),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//SYSUT2   DD   DSNAME=PUNCHSET,UNIT=2540-2
//SYSIN    DD   *
        PUNCH   MAXFLDS=1,CDSEQ=00000000,CDINCR=20
       RECORD   FIELD=(72)
       LABELS   DATA=ONLY
/*
```

IEBPTPCH Example 6. Punching a Sequential Data Set -- Assigning Sequence Numbers

IEBPTPCH Example 7

Operation	Data Set Organization	Input Device	Output Device	Comments
PRINT a partitioned directory	Input-SEQUENTIAL	DISK - 2311	System Output Device (PRINTER assumed)	1. Standard specifications. 2. Conversion to hexadecimal representation.

In this example, the directory of a partitioned data set is to be printed. The input data set resides on an IBM 2311 Disk Storage Drive. The printed output is to be converted to hexadecimal.

- The SYSUT1 DD Statement: defines the input data set (the partitioned directory).
- The SYSUT2 DD Statement: defines the output data set on the system output device (printer assumed). Each printed line contains up to 12 groups (8 characters each) of hexadecimal information. Six lines of print are required for each record; i.e., for each block of 256 bytes. Each record begins a new line of printed output; i.e., portions of two records cannot appear on the same line of print.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream. The control data set contains the PRINT, TITLE, and LABELS utility control statements.
- The PRINT Statement: initiates the print operation, indicates that the partitioned directory is organized sequentially, and specifies conversion from alphameric to hexadecimal representation.
- The TITLE Statements: specify a title and a subtitle. Neither the title nor the subtitle is converted to hexadecimal.
- The LABELS Statement: 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; i.e., the unused bytes are sometimes used by the Operating System/360 Control Program as temporary work areas. The first 4 characters of printed output indicate how many bytes of the 256-byte block pertain to the partitioned data set. Unused bytes, if any, occur in the latter portion of the directory block: i.e., they are not interspersed with the used bytes.

```
//PRINTDIR JOB  09#660,SMITH
//         EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=PDS,UNIT=2311,VOLUME=SER=111112,LABEL=(,SUL),
//              DISP=(OLD,KEEP),DCB=(RECFM=U,BLKSIZE=3625)
//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
/*
```

IEBPTPCH Example 7. Printing a Partitioned Directory

IEBPTPCH Example 8

Operation	Data Set Organization	Input Device	Output Device	Comments
PUNCH	Input-SEQUENTIAL (card deck)	CARD READER	CARD READ PUNCH 2540-2	1. Standard specifications. 2. The PUNCH utility control statement has been previously placed (as a member) in a partitioned data set. 3. The control data set is cataloged.

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.

- The SYSIN DD Statement: 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 SYSUT2 DD Statement: defines the output (punched) data set. The data set is to be punched on an IBM 2540-2 Card Read Punch (punch feed).
- The SYSUT1 DD Statement: defines the input card data set, which follows 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   UNIT=2540-2
//SYSUT1   DD   DATA
           .
           .
           input card data set
           .
           .
/*
```

IEBPTPCH Example 8. Duplicating a Card Deck

IEBPTPCH Example 9

Operation	Data Set Organization	Input Device	Output Device	Comments
PRINT	Input-SEQUENTIAL	DISK - 2311	System Output Device (PRINTER assumed)	1. User (nonstandard) specifications. 2. User routines are provided. 3. Processing ends after first record group is printed.

In this example a record group is to be printed. A user routine is provided to manipulate output records before they are printed.

- The SYSUT1 DD Statement: defines the input sequential data set. The data set resides on an IBM 2311 Disk Storage Drive.

- The SYSUT2 DD Statement: defines the output data set on the system output device (printer assumed).

- The SYSIN DD Statement: defines the control data set, which follows in the input stream. The control data set contains a PRINT, TITLE, EXITS, and two RECORD utility control statements.

- The PRINT Statement:

  initializes the print operation.

  indicates that two FIELD parameters are included in subsequent RECORD statements.

  indicates that one IDENT parameter is included in a subsequent RECORD statement.

  indicates that six literal characters are included in the subsequent ident parameter.

  indicates that 40 lines are printed on each printed page.

  indicates that processing is terminated after 32767 records are processed or after the first record group is processed, whichever comes first.

- The TITLE Statement: specifies a title.

- The EXITS Statement: specifies the name of a user routine (NEWTIME) that manipulates output records before they are printed.

- The RECORD Statement: defines the record group to be processed and indicates where information from the input records is placed in the output records. (Bytes 1-8 of the input records appear in columns 10-17 of the punched output, and bytes 9-38 are printed in hexadecimal representation and placed in columns 20-79.)

- The LABELS Statement: specifies that all user header or trailer labels are to be printed regardless of any return code issued by the user's exit routine (except 16). It also indicates that the labels are to be converted from alphameric to hexadecimal representation.

```
//PRINT    JOB  09#660,SMITH
//         EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=SEQDS,UNIT=2311,DISP=(OLD,KEEP),LABEL=(,SUL),
//              VOLUME=SER=111112,DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//SYSUT2   DD   SYSOUT=A
//SYSIN    DD   *
        PRINT   MAXFLDS=2,MAXGPS=1,MAXLITS=6,MAXLINE=40,STOPAFT=32767
        TITLE   ITEM=('TIMECONV-DEPT D06   JAN 10-17')
        EXITS   OUTREC=NEWTIME,INHDR=HDRS,INTLR=TLRS
       RECORD   IDENT=(6,'498414',1),FIELD=(8,1,,10),                  C
              FIELD=(30,9,XE,20)
       LABELS   DATA=ALL,CONV=XE
/*
```

IEBPTPCH Example 9. Printing an Edited Version of a Sequential Data Set

Program Applications

The IEBTCRIN utility program reads input from the IBM 2495 Tape Cartridge Reader (TCR), edits the data as specified by the user, and produces a sequentially organized output data set. The input is in the form of cartridges written by either the IBM Magnetic Tape SELECTRIC Typewriter (MTST) or the IBM 50 Magnetic Data Inscriber (hereafter referred to as MTDI). An input data set (one or more cartridges) must consist of either all MTST cartridges or all MTDI cartridges.

The program 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 output produced by the program is a sequential data set (a second sequential data set may be produced for error records) which can be written on any QSAM-supported output device (e.g., a system output device, a magentic tape volume or a direct access volume).

Inputs and Outputs

IEBTCRIN Table 1 lists the major inputs to and outputs from the IEBTCRIN program.

IEBTCRIN Table 1. Data Sets Used (Input) and Produced (Output) by the IEBTCRIN Program

Inputs	Input Data Set: contains the data on tape cartridges to be read from the Tape Cartridge Reader. The input data was created on either an MTST or an MTDI. Control Data Set: contains utility control statements that are used to control the functions of the utility. Utility control statements are required if the user wishes to specify options other than the default.
Outputs	Normal Output Data Set: is the sequential output produced by the utility as a result of processing the cartridge input according to the utility control statements. Error Output Data Set: contains those error records that do not conform to the specifications (utility and/or user) for a valid record. Message Data Set: contains any utility messages, diagnostic messages pertaining to the interpretation of the utility control statements and/or DD statement, and messages pertaining to the abnormal termination of the utility.

Control

The IEBTCRIN program is controlled by job control statements and utility control statements. The job control statements are required to execute or invoke the IEBTCRIN program 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

IEBTCRIN Table 2 shows the job control statements necessary for executingor invoking the IEBTCRIN program.

IEBTCRIN Table 2. Job Control Statements for the IEBTCRIN Program (Part 1 of 2)

Statement	Usage
JOB statement	This statement initiates the job.
EXEC statement	This statement specifies the program name (PGM=IEBTCRIN) or, if the job control statements for the IEBTCRIN program reside in a procedure library, the procedure name.
SYSPRINT DD statement	This statement defines a sequential message data set. The data set can be written on any QSAM supported output device. If the SYSPRINT DD statement is missing, a message is written on the operator console and the program continues processing. The user can supply DCB parameters or accept default values. If the user includes some parameters (e.g., RECFM) but omits others, the program attempts to set defaults for the missing parameters that are consistent with those that have been supplied. For example, if the user supplied RECFM=VBA, the program sets the BLKSIZE default to 129 and the LRECL default to 125. If LRECL,BLKSIZE, and RECFM are not specified the defaults will be LRECL=121, BLKSIZE=121, and RECFM=FBA. Since the program will always construct the SYSPRINT records with USASI (type A) control characters, type A control characters should always be indicated when RECFM is specified. If the user provides a parameter that is not consistent with other parameters, a message is issued and the program is terminated.
SYSUT1 DD statement	This statement defines the input data set. Only the UNIT keyword is required. The value placed in the UNIT=xxxx keyword can be '2495', the device address, or any other name that has been generated in the system as the unit device name. The VOLUME=SER=keyword may be specified to identify the tape cartridges to be mounted. The volume serial number specified 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 the VOLUME keyword 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 the two input buffers to be used. If BUFL is not specified, a value of 2000 is assumed. No other keywords/parameters are necessary. If the SYSUT1 DD statement is missing, a message is issued on SYSPRINT and the program is terminated.

(Part 1 of 2)

IEBTCRIN Table 2. Job Control Statements for the IEBTCRIN Program (Part 2 of 2)

Statement	Usage
SYSUT2 DD statement and SYSUT3 DD statement	These statements define the output data set for valid records (SYSUT2) and the output data set for error records (SYSUT3). They must be sequential data sets and can have fixed-length, variable-length, variable-length spanned, or undefined records. These data sets can be written on any QSAM supported output device. Fixed-length and variable-length records can be blocked through thespecification of the BLKSIZE and RECFM DCB parameters. The user can supply DCB parameters or accept default values. 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 4-byte error description word (see "Error Records" in this section) appended to the front of the record by this program. For variable-length records on either SYSUT2 or SYSUT3, the LRECL and BLKSIZE DCB parameters must be large enough to include the 4-byte record descriptor word. If the user includes some parameters but omits others, the program attempts to set defaults for the missing parameters that are consistent with those that have been supplied. For example, if the user specifies only the SYSUT2 parameter RECFM=FB, the program defaults BLKSIZE to MAXLN (see "The TCRGEN Statement" in this section). If LRECL, BLKSIZE, and RECFM are not specified, the defaults for SYSUT2 and SYSUT3 are LRECL=MAXLN+4, BLKSIZE=MAXLN+8, and RECFM=VB. (If editing of MTDI input is specified, the defaults for SYSUT3 are LRECL=MAXLN+8, BLKSIZE=MAXLN+12, and RECFM=VB to include the EDW). If the user provides a parameter that is not consistent with other parameters, a message is issued and the program is terminated. SYSUT2 and/or SYSUT3 DD statements may be omitted or specified as DUMMY. A message is issued on SYSPRINT indicating this situation, and processing continues.
SYSIN DD statement	This statement defines the control data set (TCRGEN and EXITS statements). The control data set normally resides in the input stream; however, it can also be defined as being a sequential data set or a member within a library (partitioned data set). If this statement is missing, all utility control statement defaults are assumed and a message is issued on SYSPRINT. If DUMMY is specified, all utility control statement defaults are assumed.

The DCB parameters defining the SYSIN, SYSPRINT, SYSUT2, and SYSUT3 data sets can be supplied from any valid source (e.g., DD statements or a data set label). Since 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, utility defaults will be 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 I/O error occurs on SYSPRINT, both the failing message and a SYNADAF message indicating the error, will be written on the programmer's console and the utility will be terminated.

UTILITY CONTROL STATEMENTS

The IEBTCRIN program is controlled by two utility control statements:

- The TCRGEN statement.
- The EXITS statement.

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. If EXITS is not specified, no exits are taken.

The TCRGEN Statement

This 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 (SYSUT1).

Name	Operation	Operand	Comments
[name]	TCRGEN	[TYPE={MTDI | MTST}]	
		[TRANS={STDUC | STDLC | name | NOTRAN}]	Valid only with TYPE=MTST specification.
		[EDIT={EDITD | EDITR | NOEDIT}]	Valid only with TYPE=MTDI specification.
		[VERCHK={NOCHK | VOKCHK}]	Valid only with TYPE=MTDI and either an EDIT=EDITD or EDIT=EDITR specification.
		[MINLN=m]	Valid only with TYPE=MTDI and either an EDIT=EDITD or EDIT=EDITR specification.
		[MAXLN=n]	Default=120
		[REPLACE={X'19' | X'xx'}]	
		[ERROPT={NORMAL | NOERR}]	This operand is ignored if a user routine is specified for the ERROR operand in the EXITS utility control statement.

TYPE= MTDI | MTST

specifies the device on which the magnetic tape cartridge(s) was written. MTDI specifies the input was created on a Magnetic Data Inscriber. MTST specifies the input was created on an IBM Magnetic Tape SELECTRIC Typewriter. MTST and MTDI cartridges must not be contained in the same input data set.

TRANS={STDUC / STDLC / name / NOTRAN}

is valid only when TYPE=MTST is specified, and is sued to specify the type of processing to be performed on MTST input by the program.

- TRANS=STDUC -- A translation from the MTST code to standard EBCDIC (see Figures 2 and 3), is performed by the program, with the exception that all alphabetic characters are translated to uppercase.

- TRANS=STDLC -- A translation from the MTST code to standard EBCDIC (see Figures 2 and 3) is performed by the program.

- TRANS=name -- This operand allows the user to specify his own translate table. The translate table must exist as a load module (with member name=name) 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/360 Principles of Operation, GA22-6821. This table is used by the program to perform the translation.

- TRANS=NOTRAN -- No translation and no special processing is to be performed by the utility. (Data is passed exactly as read from the cartridge.)

If STDUC, STDLC, or name is specified, certain of the MTST codes are processed in a special way before translation. That is, feed codes (FD), switch codes (SW), and autosearch codes (AS), both upper case 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), signals the program that all data on a cartridge has been read. Therefore, when a MTST cartridge to be processed by this program is created, the user must not use Stop Code for any purpose other than signaling the end of data on the cartridge. Stop Codes within meaningful data will cause any subsequent data on the cartridge to be lost since the cartridge will be rewound and unloaded upon encountering the Stop Code.

EDIT={EDITD / EDITR / NOEDIT}

is valid only when TYPE=MTDI is specified, and is used to specify the type of processing to be performed on MTDI input by the program.

- EDIT=EDITD -- The input is edited and the start-of-record (SOR) and end-of-record (EOR) codes are deleted and not included as part of the output record.

- EDIT=EDITR -- The input is edited and SOR and EOR codes are restrained as part of the output record.

- EDIT=NOEDIT -- No editing is to be performed by the utility. (Data, including any group separator codes (GS), is passed exactly as read from the cartridge.)

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-justified and leading zeros are inserted where necessary. Left-zero start codes are deleted from the records. Group separator codes and records which start with cancel record codes are bypassed.

VERCHK={NOCHK | VOKCHK}

is valid only when TYPE=MTDI and either EDIT=EDITD or EDIT=EDITR are specified. This parameter can be used when the program is to check for record verification on the MTDI input.

- VERCHK=NOCHK -- The program is not to check for record verification. Either a record mark (RM) or a verify OK (VOK) code is considered a valid end-of-record code.

- VERCHK=VOKCHK -- The program is to check for record verification. A record that does not contain a verify OK code is to be considered an error record.

MINLN=m

is valid only when TYPE=MTDI and either EDIT=EDITD or EDIT=EDITR are specified. This parameter can be used to specify in bytes the length of the shortest valid edited record. If the program encounters a record shorter than this specified length, the record is considered an error record. Refer to Table 3 for the values that can be specified for MINLN. If MINLN is omitted, no minimum length checking is performed.

MAXLN=n

is valid when either TYPE=MTDI or TYPE=MTST is specified.

When TYPE=MTST or TYPE=MTDI without editing is specified, all records but the last record passed to the program's output routine will contain the number of bytes specified by MAXLN, plus four for the record descriptor word when variable length records are specified (RECFM=V). The program will not indicte the end of data from one cartridge and the beginning of data from the next. Usually this transition of the data from one cartridge to another will occur within an output record. The last record passed to the output routine (the record containing the last data from the final cartridge of the input data set) will contain 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 will depend on the record length (LRECL) specified for the output data set.

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 can not be determined within this length, an end of record condition is forced and the record is considered an error record. Table 3 lists the values that can be specified for MAXLN. If MAXLN is omitted, a value of 120 is assumed.

IEBTCRIN Table 3. Values for MINLN and MAXLN

TCRGEN	MINLN	MAXLN
TYPE=MTST TYPE=MTDI EDIT=NOEDIT	Not Applicable	User specifies the number of bytes to be passed as a record.
TYPE=MTDI EDIT=EDITD	Should equal the number of bytes in the shortest valid record after editing (excluding SOR and EOR codes).	Should equal the number of bytes in the longest valid record after editing (excluding SOR and EOR codes).
TYPE=MTDI EDIT=EDITR	Should equal the number of bytes in the shortest valid record after editing (including SOR and EOR codes).	Should equal the number of bytes in the longest valid record after editing (including SOR and ERO codes).
Note: The values for MINLN and MAXLN should not include the 4-byte record descriptor word added to a variable length record.		

REPLACE={X'19' / X'xx'}

valid when either TYPE=MTDI or TYPE=MTST is specified, can be used to specify the hexadecimal representation of the character used by the program to replace error bytes. This gives the user the capability of identifying and possibly correcting error bytes in the user error exit routine (if specified) or in subsequent processing. To facilitate discovery of the error byte, the specified REPLACE character should be one that does not normally appear in the data.

X'19' is chosen as the default value because it is an end-of-data signal for either an MTST or MTDI cartridge. Therefore, it can never appear as a valid data byte.

For REPLACE=X'xx' the user can replace xx with hexadecimal characters of his own choosing. These choices can be made from Figure 1 if the input was created on an MTDI, or from Figure 2 if the input was created on an MTST. The replacement of error bytes is accomplished prior to any specified MTST translation.

Hexadecimal characters representing special purpose codes that must not be used as replacement bytes when MTDI with editing is specified are:

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)

Hexadecimal characters representing special purpose codes that must not be used as replacement bytes when MTST with translation is specified are:

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' thru X'FF'	

The special purpose codes listed above are used by the program when constructing records. Use of these codes causes a message to be issued and the utility to be terminated.

ERROPT={NORMAL|NOERR}

is valid when either TYPE=MTDI or TYPE=MTST is specified. This parameter can be used to specify the disposition of all error records. This operand is ignored if a user routine is specified for the ERROR operand in the EXITS utility control statement. (The ERROR operand allows the user to specify the disposition of each error record via a return code.) A definition of an error record is contained in "Error Records" in this section.

- ERROPT=NORMAL -- All error records, as determined by the program, are placed in the error data set (SYSUT3).
- ERROPT=NOERR -- All records (including error records) are placed in the normal output data set (SYSUT2). No records are placed in the error data set (SYSUT3).

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	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					|	¬	?	"									

This figure represents the character set and control codes as read from and MTDI created cartridge.

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

IEBTCRIN Figure 1. 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	l	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	,										

This figure represents the character set and control codes as read from an MTST created cartridge.

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

IEBTCRIN Figure 2. 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							?	"									

Note: The STDUC option permits translating both lowercase and uppercase alphabetic characters to uppercase.

TAB = Tab code
CR = Carrier return
BSP = Backspace
SRC = Search
STX = Stop transfer
SP = Space

IEBTCRIN Figure 3. MTST Codes After Translation by IEBTCRIN With TRANS=STDLC

The EXITS Statement

The EXITS statement is used to identify exit routines supplied by the user. All specified exit routines must exist in either the user job library or the link library.

Name	Operation	Operand	Comments
[name]	EXITS	[ERROR=routine name]	This exit is taken just prior to passing an error record to the error output data set (SYSUT3)
		[OUTREC=routine name]	This exit is taken just prior to passing a record to the normal output data set (SYSUT2).
		[OUTHDR2=routine name]	This exit is taken during the opening of the SYSUT2 data set.
		[OUTHDR3=routine name]	This exit is taken during the opening of the SYSUT3 data set.
		[OUTTLR2=routine name]	This exit is taken during the closing of the SYSUT2 data set.
		[OUTTLR3=routine name]	This exit is taken during the closing of the SYSUT3 data set.

Upon entry, a parameter list is supplied to the exit routine containing information as shown in Table 1 Appendix A.

Upon returning from the exit routine, the user must provide an acceptable return code according to the entries shown in Table 2 Appendix A.

ERROR=routine name
specifies the symbolic name of a routine that handles error records identified by the IEBTCRIN utility program. This operand nullifies any ERROPT operand present. This routine can be used to analyze and, if possible, correct the error record.

If MTDI is specified with either EDITD or EDITR, a 4-byte Error Description Word (EDW) is appended to the front of each error record describing the error condition. For further definition of the EDW, see "Error Records" in this section. If the SYSUT3 data set has specified variable length records, a 4-byte Record Descriptor Word (RDW) is also appended to the front of the record. For further description of the RDW, see IBM System/360 Supervisor and Data Management Services, GC28-6646.

The user can examine and modify any byte in the record or EDW. He can also change the record length, 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 workareas 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, since it will be re-created if the record is to be output by the utility. However, if the user does his own output from this routine, he must ensure that the RDW is correct for the record.

If the utility program is to output the record, the length of the output record will depend on the RECFM specification:

(1) Fixed length records - may have a maximum length equal to LRECL. Records larger than this will be truncated.

(2) Variable length records - may have a maximum length equal to LRECL. Records larger than this will be truncated.

(3) Undefined length records - may have a maximum length equal to BLKSIZE. Records larger than this will be 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 into the following record. However, it is not used for record length checking of subsequent records.

The user is cautioned that any modifications to the EDW, record, or record length may affect the editing of subsequent records. This is because the EDW, record, and record length returned from this exit are used during the editing of the following records.

If MTST or MTDI with NOEDIT is specified, the user can examine and modify any byte in the record. The record length can also be changed, subject to the same restrictions as for MTDI with EDITD or EDITR.

OUTREC=routine name
specifies the symbolic name of a routine that receives access to a record just before the record is passed to the normal output data set (SYSUT2). In this exit routine, the user can process the record and/or 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 since the record returned from this exit will be used to accomplish data duplication in the following record.

If the SYSUT2 data set has specified variable length records, a 4 byte RDW will be appended to the front of the record. The same restrictions apply to record modifications as for the ERROR exit described above.

OUTHDR2=routine name
specifies the symbolic name of a routine that creates user output header labels for the normal output data set (SYSUT2).

OUTHDR3=routine name
specifies the symbolic name of a routine that creates user output header labels for the error data set (SYSUT3).

OUTTLR2=routine name
specifies the symbolic name of a routine that creates user output trailer labels for the normal output data set (SYSUT2).

OUTTLR3=routine name
specifies the symbolic name of a routine that creates user output trailer labels for the error data set (SYSUT3).

Error Records

Once the program has determined a record to be in error, the record is passed to the user error exit routine (if the ERROR operand is specified in the EXITS utility control statement). If this ERROR exit is not specified, the action to be taken will be determined by the ERROPT option of the TCRGEN utility control statement.

MTST INPUT OR MTDI INPUT WITH NO EDITING

When either MTST input or MTDI input with no editing has been 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 the REPLACE operand of the TCRGEN statement. The record is considered an error record, but since a data check is the only error that can occur, no error description word is appended to the error record.

MTDI INPUT WITH EDITING

When TYPE=MTDI and either EDIT=EDITD or EDIT=EDITR have been specified, the program maintains information about each record as it is being edited. This information is summarized in the Error Description Word (EDW) described in Figure 4. When the EDW contains a nonzero value 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.

The Error Description Word (EDW)

The Error Description Word consists of four bytes which are appended to the start of an error record.

ERROR DESCRIPTION WORD[1]			
Byte 0	Byte 1	Byte 2	Byte 3
Level status	Type status	SOR	EOR
(0)=for any error record which will not cause questionable data to be in the following record(s). (1)=for any error which may cause questionable data in the following record(s). (2)=for any error record which has questionable data due to a nonzero error level of the preceding record(s) and may cause questionable data to be in the following record(s).	(0)=No identifiable error. (1)=Start-of-record (SOR) or End-of-record (EOR) in error. (2)=Length error. (4)=Field error. (8)=Data Check error. Note: Error records may have hex combinations of the above error types e.g., a (C) would indicate a Data Check and a Field error.	(1)=P1 (2)=P2 (3)=P3 (4)=P4 (5)=P5 (6)=P6 (7)=P7 (8)=P8 (E)=None of the above. Start-of-record (SOR) in error.	(U)=Unverified Record. (V)=Verified Record. (E)=Neither of the above. End-of-record (EOR) in error.
[1]The error description word is in EBCDIC format, e.g., a (2) is represented as X'F2'; a (C) is represented as X'C3'.			

IEBTCRIN Figure 4. Byte Values of the Error Description Word

Level Status (Byte 0): The level status indicator identifies error records that result from interrecord dependency and that cannot be identified in the type status byte.

The level status is presented with each error record and has a value of:

0 For any error record that will not cause questionable data in following record(s). A nonzero Type Status will accompany this byte.

1 For any error record that may cause questionable data in following record(s), and for which the level status of the previous record was 0.

2 For any error record that has questionable data within its content 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 record(s), and for which the level status of the previous record was 1 or 2.

A level status of other than 0 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.

- The record is longer than the user-specified maximum length record (MAXLN).
- 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 which contains no errors as identified in the type status, with the possible exception of being shorter than the specified minimum length (MINLN).
- The record has a data duplication dependency on a previous record with one of the above errors.

The level status is set to 0 whenever the IEBTCRIN utility encounters: a record without one of the previous errors, a canceled record, or the first record of a cartridge.

Type Status (Byte 1): The type status indicator identifies records in error because of SOR, EOR, length, field, and/or data check error conditions.

The type status is presented with each error record and has a value of:

0 For any record that contains none of the following identifiable error(s), but contains questionable data due to a level status of other than zero (0) (see "Level Status" in this section).

1 For any record that has:
- A SOR character of other than P1 through P8 or a GS code.
- An EOR character of other than a VOK code for records when the user has specified VERCHK=VOKCHK.
- An EOR character of other than a VOK or RM code for records when the user has specified VERCHK=NOCHK.

2 For any record that has an incorrect length because it is:
- Longer than the specified maximum (MAXLN), or
- Shorter than the specified minimum (MINLN), or
- Not equal to the length of the first valid record of the same program level encountered on this cartridge.

4 For any record that has a field error(s). A field error occurs when duplication and/or left zero justification functions did not occur in a field due to an error condition. See "MTDI Editing Criteria" in this section.

8 For 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/or 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): This byte contains an indication of the start-of-record (SOR) character associated with this record.

End-of-Record (Byte 3): This byte contains an indication of the end-of-record (EOR) character associated with this record.

MTDI Editing Criteria

The cartridges created on the IBM 50 Magnetic Data Inscriber contain a continuous stream of data bytes (i.e., there are no interblock gaps). Therefore when editing has been specified, it is the responsibility of the program to extract records one at a time from this data stream. To accomplish this, the program scans for control codes written by the MTDI. The following criteria are used by this program to accomplish editing.

START-OF-RECORD AND END-OF-RECORD LOCATIONS

The start-of-record (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 end-of-record (EOR) location.
- The location of an SOR code.
- The location of a 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 start-of-record character was a valid GS code.
2. The location of the first encountered RM or VOK code, if that location is within the length of the maximum user specified record size (MAXLN).
3. The location of any code preceding either a valid SOR code, or the end of media code, if that location is within the length of the maximum user specified record size (MAXLN).
4. The location determined in 2 or 3, regardless of the maximum user-specified record size (MAXLN), 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 MAXLN.

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 and EOR code and will be bypassed.

SPECIAL CONSIDERATIONS

- 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 zero.
- 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 (MAXLN) can result in an input record of this size.

- Data duplication is accomplished by replacing the DUP 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 which 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 will be 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 (MAXLN) is exceeded before encountering the valid end of a left-zero field.

End of Cartridge

Unique codes, written by the MTST or the MTDI device, signal to 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, this end of cartridge code is the End Data code (ED).

The program terminates input from a cartridge upon encountering the end of cartridge code and then rewinds the cartridge. The program 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 has been pressed, and the end of cartridge for the last cartridge has been recognized. An end-of-file indication will be passed to the OUTREC and/or ERROR exits if specified by setting register 1 equal to zero.

Sample Error Records

Figure 5 illustrates a stream of data bytes read sequentially from the tape cartridge reader. The program will construct records from the stream as shown in Figure 6.

These records show some of the errors that can occur during processing and their effect on the error description word. The following coding shows the parameters specified for these records:

```
TCRGEN   TYPE=MTDI,EDIT=EDITR,VERCHK=VOKCHK,                    X
         MAXLN=50,REPLACE=X'5B'
```

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.

Records 2 through 9 are classified as error records by the program.

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 onto the following record to be questionable. The type status (9) 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 since it requires duplication from the previous record that had a level status of 1.

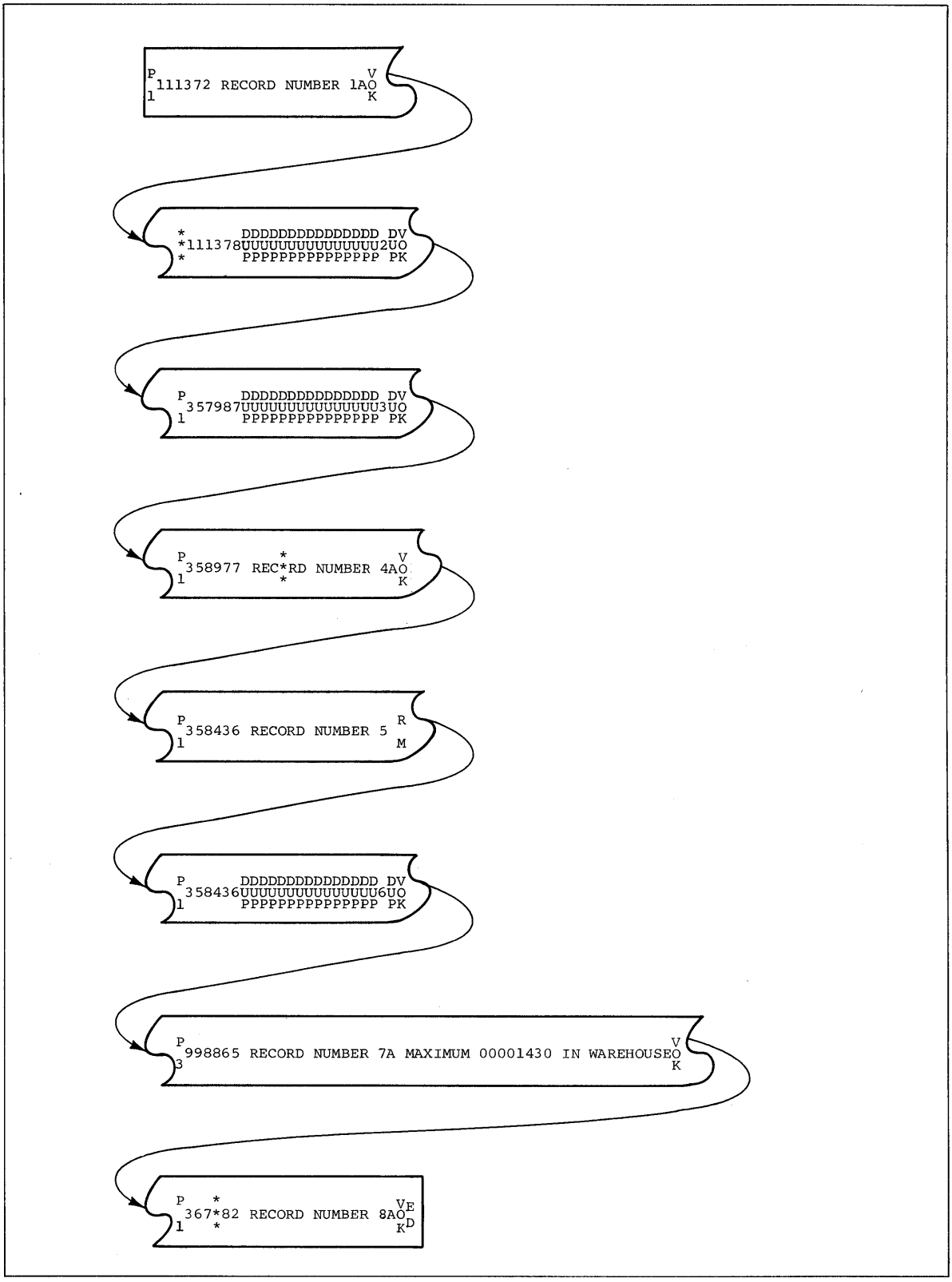
Record 4 has a data check. Since it contained no DUP codes, the level status is set to 0.

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). Because the program 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.

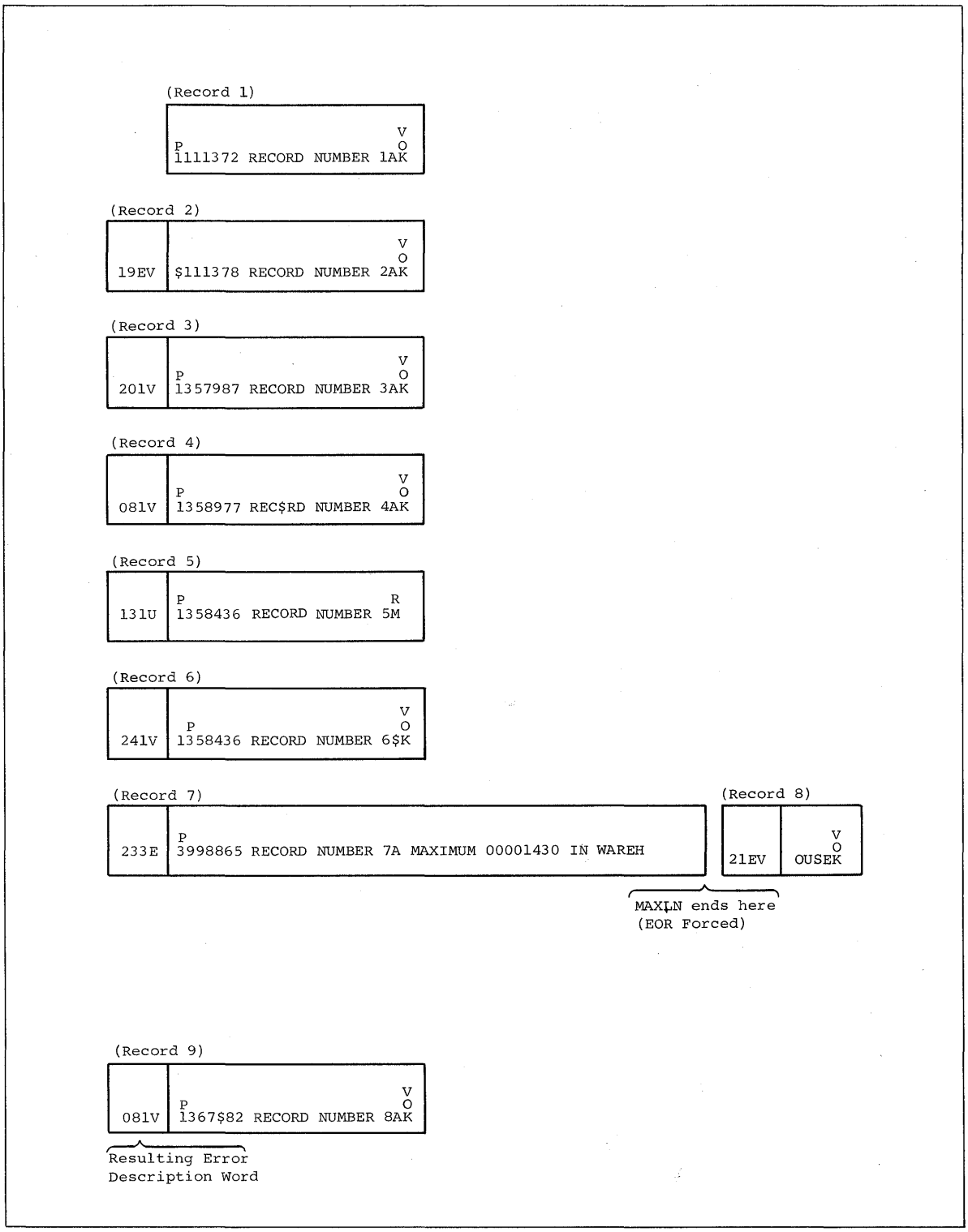
The seventh input record is longer than the maximum specified record length (MAXLN). Note that it is passed as two records. The first record (Record 7) indicates an EOR error and a length error; the second (Record 8) indicates an SOR error. Because Record 7 is an error record, its length (50 bytes) is not established as the valid length for all program level 3 records on this cartridge.

Record 9 has a data check. Since it contained no DUP codes, the level status is set to 0.



* Indicates a data check

IEBTCRIN Figure 5. Tape Cartridge Reader Data Stream



IEBTCRIN Figure 6. Record Construction

IEBTCRIN Examples

The following examples illustrate some of the uses of the IEBTCRIN program.

IEBTCRIN Example 1

Operation	Data Set Organization	Input Device	Output Device	Comments
Produce sequential output with editing from a tape cartridge reader	INPUT - N/A OUTPUT - SEQ	2495 tape cartridge reader	normal: DISK-2314	Fixed Blocked Output
			error: TAPE 9-TRACK	1. undefined output. 2. ERROR exit routine specified.

In this example, input from a tape cartridge is to be edited with normal records written to a 2314, and error records written to a 9-Track tape.

- The SYSUT1 DD statement: defines the input tape cartridge data set. A console message will instruct the operator to mount a set of cartridges named MYTAPE. The two input buffers will each be 3000 bytes long (BUFL). The UNIT parameter assumes that TCR has been SYSGENed as a UNITNAME for the Tape Cartridge Reader.

- The SYSUT2 DD statement: defines a sequential data set for the normal records. The logical record length=100 bytes and the blocksize=1000 bytes. The data will be written on a 2314 direct access storage facility.

- The SYSUT3 DD statement: defines a sequential data set for the error records. The records will be undefined with maximum blocksize=104 bytes including a 4 byte error description word appended by the program.

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

- The TCRGEN statement: indicates the input was written by an IBM 50 Magnetic Data Inscriber. 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 hex '5B' ($). The VERCHK operand is defaulted to NOCHK. Minimum record length checking is not requested (no MINLN).

- The EXITS statement: indicates that a user has provided an exit routine to handle error records. Since no job library has been specified, the exit routine (MYERR) must be contained in the link library.

```
//JOBNAME  JOB     0,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=2314,          X
//                 VOLUME=SER=111222,SPACE=(TRK,(10,10)),              X
//                 DCB=(LRECL=100,BLKSIZE=1000,RECFM=FB)
//SYSUT3   DD      DSNAME=ERRSET,UNIT=2400,                            X
//                 VOLUME=SER=000001,DISP=(NEW,KEEP),                  X
//                 DCB=(BLKSIZE=104,RECFM=U)
//SYSIN    DD      *
           TCRGEN  TYPE=MTDI,EDIT=EDITD,MAXLN=100,REPLACE=X'5B'
           EXITS   ERROR=MYERR
/*
```

IEBTCRIN Example 1. Producing Sequential Output With Editing

IEBTCRIN Example 2

Example 2 shows the coding required to invoke the IEBTCRIN utility program 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 a halfword boundary)
OPTLIST DC   H'0'           (Length must be zero for IEBTCRIN)

        CNOP 2,4            (DDNAME list must be on a 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)
```

IEBTCRIN Example 2. Invoking IEBTCRIN from a Problem Program Via the LINK Macro

Program Applications

The IEBUPDTE utility program incorporates both IBM and user-generated source language modifications into sequential data sets or into partitioned data sets. The input and output data sets contain blocked or unblocked logical records with record lengths of up to 80 bytes. Exits are provided at appropriate places for user routines that process user header and trailer labels.

The program can:

- Add, copy, and replace members or data sets.
- Add, delete, replace, and renumber the records within an existing member or data set.
- Assign sequence numbers to the records of a member or data set.
- Convert sequential input into partitioned output or vice versa.

The IEBUPDTE program is 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 the program, the highest return code encountered within the program is passed to the calling program.

Creating and Updating Symbolic Libraries

The IEBUPDTE program can create a library of partitioned members consisting of 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 of members.

Incorporating Changes to Partitioned Members or Sequential Data Sets

The IEBUPDTE program can modify an existing partitioned data set or sequential data set used as input to the program. Logical records can be replaced, deleted, renumbered, or added to the member or data set.

A sequential data set residing on a magnetic tape volume can be used to create a new master (i.e., 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, according to the program application, 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, according to the program application, by modifying the data set directly on the volume on which it resides.

Changing the Organization of a Data Set

The IEBUPDTE program 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.

Inputs and Outputs

IEBUPDTE Table 1 lists the major inputs to and outputs from the IEBUPDTE program.

IEBUPDTE Table 1. Data Sets Used (Input) and Produced (Output) by the IEBUPDTE Program

Inputs	Input Data Set: This data set (also called the old master data set) is to be modified or used as source data for a new master. The input is either sequential or a member of a partitioned data set. Control Data Set: This data set contains utility control statements and, if applicable, input data. The data set is required for each application of the IEBUPDTE program.
Outputs	Output Data Set: This data set is the result of the IEBUPDTE operation. The data set can be either sequential or partitioned. It can be either a new data set (i.e., created during the present job step) or an existing data set, modified during the present job step. Message Data Set: This data set is sequential. It contains a listing of: • Utility program identification. • Control statements used in the job step. • Error or warning messages, if applicable. • Modifications made to the input data set.

ADDITIONAL OUTPUTS

The IEBUPDTE program provides a return code to indicate the results of program execution. The return codes and their interpretations are as follows:

00 -- successful completion.
04 -- 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 (as indicated by a utility "function" statement).
12 -- an unrecoverable error occurred. The job step is terminated.
16 -- a label processing code of 16 was received from a user label processing routine. The job step is terminated.

Control

The IEBUPDTE program is controlled by job control statements and utility control statements. The job control statements are required to execute or invoke the IEBUPDTE 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 the IEBUPDTE program and, in certain cases, to supply new or replacement data.

JOB CONTROL STATEMENTS

IEBUPDTE Table 2 shows the job control statements necessary for executing or invoking the IEBUPDTE program.

IEBUPDTE Table 2. Job Control Statements for the IEBUPDTE Program (Part 1 of 2)

Statement	Usage
JOB statement	This statement initiates the job.
EXEC statement	This statement specifies the program name (PGM=IEBUPDTE), or, if the job control statements for the IEBUPDTE program reside in a procedure library, the procedure name. The statement specifying PGM=IEBUPDTE also specifies PARM=([NEW / MOD][,inhdr][,intlr]) where NEW indicates that the input consists solely of the control data set. The input data set is not defined if NEW is specified. MOD indicates 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 specifies the symbolic name of a routine that processes the user header label on the volume containing the control data set. intlr specifies the symbolic name of a routine that processes the user trailer label on the volume containing the control data set. For a detailed discussion of the processing of user labels as data set descriptors, refer to "Appendix F: Utility Program Handling of User Labels." If a parameter is omitted from the PARM field, its absence must be indicated by a comma; e.g., PARM=(NEW,,intlr). If all three parameters are omitted, PARM need not be coded.

(Part 1 of 2)

IEBUPDTE Table 2. Job Control Statements for the IEBUPDTE Program (Part 2 of 2)

Statement	Usage
SYSPRINT DD	This statement 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. (This DD statement must be present for each execution or invocation of the IEBUPDTE program.)
SYSUT1 DD statement	This statement defines the input (old master) data set. It can define a sequential data set on a card reader, a magnetic tape volume, or a direct access volume. Or, it can define a partitioned data set on a direct access volume. This DD statement is not required if the input consists solely of the control data set. Note: If an UPDATE=INPLACE[1] operation is specified, the data set defined by the SYSUT1 DD statement is considered to be both the input data set and the output data set; i.e., the SYSUT2 DD statement need not be coded. If the ADD[1] function is specified, and the input data resides in the input stream, the SYSUT1 DD statement need not be coded.[2] In all other cases both SYSUT1 and SYSUT2 must be included.
SYSUT2 DD	This statement defines the output data set. It can define a sequential data set on a card punch, a printer, a magnetic tape volume, or a direct access device. Or, it can define a partitioned data set on a direct access device. Space must be allocated for an output data set that is to reside on a direct access device, unless the data set is an existing data set, in which case space should not be allocated. It must not be a DUMMY data set.
SYSIN DD	This statement defines the control data set. The control data set normally resides in the input stream; however, it can also be defined as being a member within a library of partitioned members. The SYSIN DD statement must be included for each execution or invocation of the IEBUPDTE program.
[1]Refer to "Function Statements" for a discussion of the UPDATE=INPLACE operation.	
[2]Refer to "Function Statements" for a discussion of the ADD operation.	

Notes: 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).

If both the SYSUT1 and SYSUT2 DD statements specify standard user labels (SUL), the IEBUPDTE program will copy user labels from SYSUT1 to SYSUT2. See "Appendix F: Utility Program Handling of User Labels" for a discussion of the available options for user label processing.

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.

When adding a member to an existing partitioned data set using the ADD function, the DCB parameters specified, if any, 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 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 name(s) themselves should not appear on the DD statements; they should be referenced only through the IEBUPDTE control statements.

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:

- The PARM parameter of the EXEC statement must imply or specify MOD.
- The DISP parameter of the SYSUT1 DD statement must specify OLD.
- The DISP parameter of the SYSUT2 DD statement must include the MOD subparameter.

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-length blocked (RECFM=FBS) or unblocked records. The control data set contains 80-byte, blocked or unblocked records.

UTILITY CONTROL STATEMENTS

The IEBUPDTE program is controlled by five types of utility control statements:

- Function statements.
- Detail statements.
- Data statements.
- LABEL statements.
- ALIAS and ENDUP statements.

The function statements are used to initiate the utility operation. The detail statements are used with the function statements for specific applications. The data statements are logical records of data used as new or replacement records in the output data set. The LABEL statements are used to indicate that the following data statements should be treated by the IEBUPDTE program as user labels. The ALIAS and ENDUP statements are used to assign alias names and to terminate the utility program.

Function Statements

A function statement is used to initiate the utility operation. At least one function statement must be provided for each member or data set to be processed. A statement contains:

1-2	Name	Operation	Operand	
./	[name]	ADD REPL CHANGE REPRO	[LIST=ALL] [SEQFLD=dd1] [NEW=PO] [NEW=PS] [MEMBER=cccccccc] [IGNORE=EOF] [COLUMN=dd] [UPDATE=INPLACE]	Applicable to partitioned or sequential organization.
			[INHDR=cccccccc] [INTLR=ccccccc] [OUTHDR=cccccccc] [OUTLR=cccccccc] [TOTAL=(routinename,size)]	Applicable to sequential organization only.
			[NAME=cccccccc] [LEVEL=hh] [SOURCE=x] [SSI=cccccccc]	Applicable to partitioned organization only.

./ (columns 1 and 2)
identifies the control statement.

name
specifies an optional name which, if coded, begins in column 3 and extends no further than column 10.

ADD, REPL, CHANGE, or REPRO
is included in the operation field, according to the function to be performed:

ADD: The ADD statement indicates that a member or a data set is to be added in its entirety onto an output master data set. The member name specified in the ADD statement must not already exist in the old master data set.

Note: If SYSUT1 is omitted, PARM=NEW must be coded in the EXEC job control statement.

REPL: The REPL statement indicates that a member or 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 specified in the REPL statement must already exist in the old master data set.

CHANGE: The CHANGE statement indicates that modifications are to be made to an existing member or data set. Use of the CHANGE statement without a NUMBER and/or DELETE and/or data statement causes an error condition.

REPRO: The REPRO statement indicates that a member or a data set is to be copied in its entirety onto a new master data set.

Notes: At least one blank must precede and follow the operation field.

A member or a data set can be added directly to an old master data set if the amount of space originally allocated to the old master is sufficient to incorporate that new member or data set.

When (1) a member replaces (REPL) an identically named member on the the old master data set, or (2) a member is changed (CHANGE) and rewritten on the old master, the alias name (if any) of the original member still refers to the original member. However, if an identical alias name is specified for the newly written member, the original alias name entry in the directory is changed to refer to the newly written member.

Members can be deleted from a copy of a library by being omitted from a series of REPRO 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 (REPRO) only the desired data sets onto a new volume. If the NEW subparameter is coded in the EXEC statement, the ADD function is the only function permitted.

keywords
are included in the operand field according to the function to be performed. The keywords are separated by commas.

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.
If LIST is omitted, the SYSPRINT data set contains modifications and control statements only.

SEQFLD=ddl
specifies, in decimal, the starting column (column 80 or below) and length (8 or less) of sequence numbers within existing logical records and subsequent data statements. (dd +1 cannot exceed LRECL + 1.)

The default for this parameter is SEQFLD=738; i.e., an eight byte sequence number beginning in column 73. Therefore, if existing logical records and subsequent data statements have sequence numbers in columns 73-80, this keyword need not be coded. In any case, sequence numbers on incoming data statements and existing logical records must be padded to the left with enough zeroes to fill the length of the sequence field.

NEW=PO
indicates 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.

NEW=PS
indicates 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.

The NEW keyword should not be specified unless the organization of the new master data set is different from the organization of the old master. Refer to IEBUPDTE Table 3 for the use of the NEW keyword with the NAME and MEMBER keywords.

MEMBER=cccccccc
assigns a member name to the member placed into the partitioned data set defined by the SYSUT2 DD statement. This keyword is used only when SYSUT1 defines a sequential data set, SYSUT2 defines a

partitioned data set, and NEW=PO is specified. (Refer to IEBUPDTE Table 3 for the use of the MEMBER keyword with the NEW keyword.)

IGNORE=EOF (valid only for card reader input in PCP system)
indicates that a delimiter statement (/*) is being included as input data. The IEBUPDTE program is terminated when two consecutive delimiter statements are encountered, or when an ENDUP statement is encountered. The IGNORE keyword is not applicable to the entire job step; therefore, it must be included with each function statement preceding a delimiter statement that is to be used as input data.

COLUMN=dd (used only with the CHANGE statement)
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, as follows:

1. The utility program 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 40-80 (assuming 80-byte logical records) of a subsequent data statement are to be used as replacement data for columns 40-80 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 detail (DELETE) 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.

UPDATE=INPLACE (used only with the CHANGE statement)
indicates 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.

When UPDATE=INPLACE 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 statement and one UPDATE=INPLACE keyword is 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.
- SSI information cannot be changed.

INHDR=cccccccc
specifies the symbolic name of the user routine that handles user input (SYSUT1) header labels, if any. When used with UPDATE=INPLACE, this routine assumes a special function. See "User Label Processing with UPDATE=INPLACE" for a discussion of this function.

INTLR=cccccccc
specifies the symbolic name of the user routine that handles user input (SYSUT1) trailer labels, if any.

OUTHDR=cccccccc
specifies the symbolic name of the user routine that handles user output (SYSUT2) header labels, if any.

OUTTLR=cccccccc
specifies the symbolic name of the user routine that handles user output (SYSUT2) trailer labels, if any.

TOTAL=(routinename,size)
specifies that exits to a user's routine are to be provided prior to writing each record. When the option is specified, the user must supply two parameters. The first is the name of the user's totaling routine. The other is the size (number of bytes) needed to contain 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 the section "Exit Routine Linkage" for a discussion of linkage conventions for user routines.

The above five parameters are valid only when the utility is processing sequential data sets, but not with UPDATE=INPLACE. For a detailed discussion of the processing of user labels as data set descriptors, and for a discussion of user label totaling, refer to "Appendix F: Utility Program Handling of User Labels."

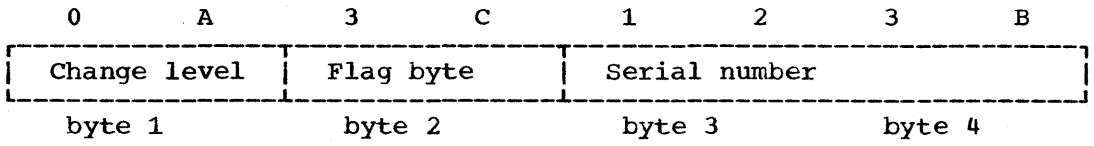
NAME=cccccccc
indicates the name of the member placed into the partitioned data set. The member name need not be specified in the DD statement itself.

A NAME keyword must be provided to identify each input member. Refer to IEBUPDTE Table 3 for the use of the NAME keyword with the NEW keyword.

LEVEL=hh
specifies the change (update) level in hexadecimal (00-FF). The level number is recorded in the directory entry of the output member.

SOURCE=x
specifies either user modifications (SOURCE=0) or IBM modifications (SOURCE=1). The source is recorded in the directory entry of the output member.

SSI=hhhhhhhh
specifies eight hexadecimal characters of system-status information (SSI) that is to be placed in the directory of the new master data set as four packed hexadecimal bytes of user data. The SSI information overrides any LEVEL or SOURCE keyword data given on the same function statement. The SSI information is packed as follows: (example) SSI=0A3C123B is stored as:



Refer to the publication IBM System/360 Operating System: Maintenance, Form C27-6918, for a detailed discussion of the format of the SSI information.

IEBUPDTE Table 3. NEW, MEMBER, and NAME Keywords

If SYSUT1 defines→	and SYSUT2 defines→	Use the combinations of keywords shown below.→
a partitioned data set	a partitioned data set	If the operation is an ADD operation, use NAME=cccccccc 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 desired, an ALIAS statement can also be used. If the operation is a CHANGE, REPL, or REPRO operation, use NAME=cccccccc 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.
----------	a new library (partitioned data set)	NAME=cccccccc with each ADD statement to assign a name for each member placed into the partitioned data set.
a partitioned data set	a sequential data set	NAME=cccccccc with the appropriate function statement to specify the name of the member in the partitioned data set defined by the SYSUT1 DD statement. also 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.)
a sequential data set	a partitioned data set	MEMBER=cccccccc with the appropriate function statement to assign a name to the member to be placed into the partitioned data set defined by the SYSUT2 DD statement. also use NEW=PO to specify the change in organization from sequential to partitioned.

Detail Statements

A detail statement is used with a function statement for certain applications, such as deleting or renumbering selected logical records. A detail statement contains:

1-2	Name		Operand	
./	[name]	{NUMBER} {DELETE}	[SEQ1=cccccccc] [SEQ2=cccccccc]	Used with the NUMBER or DELETE statement.
			[SEQ1=ALL] [NEW1=cccccccc] [INCR=cccccccc] [INSERT=YES]	Used only with the NUMBER statement.

./ (columns 1 and 2)
identifies the control statement.

name
specifies an optional name which, if coded, begins in columns 3 and extends no further than column 10.

NUMBER or DELETE
is included in the operation field, according to the operation:

NUMBER: The NUMBER statement is used with the CHANGE statement to change the sequence number of one or more logical records within a member or data set, and with ADD and REPL statements to assign sequence numbers to the records within new or replacement members or data sets. When used with the ADD or REPL statements, no more than one number statement is permitted for each ADD or REPL.

DELETE: The DELETE statement is used with a CHANGE statement to delete one or more logical records from a member or data set.

Notes: At least one blank must precede and follow the operation field.

Logical records cannot be deleted in part; i.e., a COLUMN=dd specification is not applicable to records that are to be deleted. Each specific sequence number is handled only once in any single operation.

keywords
are included in the operand field, according to the operation to be performed. The keywords are separated by commas.

SEQ1=cccccccc
specifies the sequence number of the first logical record to be renumbered or deleted. The SEQ1 keyword is not coded in a NUMBER statement that is used with an ADD or REPL function statement.

Note: When SEQ1 is used in an INSERT=YES operation, the SEQ1 value 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=YES keyword for a complete discussion.

SEQ2=cccccccc
specifies the sequence number of the last logical record to be renumbered or deleted. If only one record is to be deleted, the SEQ1 and SEQ2 specifications must be identical. The SEQ2 keyword

is not coded in a NUMBER statement that is used with an ADD or REPL function statement.

SEQ1=ALL (used only with a CHANGE statement/NUMBER statement combination)
specifies a renumbering operation for the entire input member or data set. This keyword must be coded if sequence numbers are to be assigned to existing logical records having blank sequence numbers. If this keyword is not coded, all existing logical records having blank sequence numbers are copied directly onto the output master data set.

When SEQ1=ALL is coded:

- The SEQ2 keyword need not be coded.
- One NUMBER statement is permitted per function statement.

Refer to the INSERT=YES keyword for additional SEQ1 qualifications with that keyword.

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 the NEW1 keyword must be equal to or greater than a value specified in the SEQ1 keyword (unless SEQ1=ALL is specified, in which case this rule does does not apply).

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.

Note: The SEQ1, SEQ2, NEW1, and INCR keywords (excluding SEQ1=ALL) specify 8-character (maximum) decimal numbers. Only the significant numbers of a value need be coded; e.g., SEQ1=00000010 can be shortened to SEQ1=10.

INSERT=YES
(used only with a CHANGE statement/NUMBER statement combination) 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.

When INSERT=YES is coded:

- The SEQ1 keyword specifies the existing logical record after which the insertion is to be made. (The SEQ2 keyword need not be coded.) SEQ1=ALL cannot be specified.

- The NEW1 keyword assigns a sequence number to the first logical record to be inserted.

- The INCR keyword is used to renumber as much as is necessary of the member or data set from the point of the first insertion; i.e., 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 statements, if any, must specify INSERT=YES. If a prior numbering operation renumbers the logical record specified in the SEQ1 keyword of a subsequent NUMBER statement, the NEW1 and INCR keyword specifications (if any) 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 keyword of a subsequent NUMBER statement, the latter statement must contain NEW1 and INCR keyword 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, an end of file indication, or a data statement containing a sequence number is encountered.

Data Statements

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 operation, a data statement contains new or replacement data, as follows:

- If the sequence number in the data statement is identical to 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.)

When used with an ADD or REPL operation, a data statement contains new data to be placed in the output master data set.

Notes: Sequence numbers within the old master data set are assumed to be in ascending order.

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 73-80; if the numbers are in columns other than these, the length and relative position must be specified in a SEQFLD keyword within a preceding function statement.)

LABEL Statements

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 the utility that the last label data statement of the group has been read. There can be no more than two LABEL statements per execution of the utility. 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=1 to 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 the IEBUPDTE program 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, the IEBUPDTE program passes a parameter list to the output label routines. This parameter list is described fully in "Appendix F: Utility Program Handling of User Labels." 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, via return codes to the utility, specifies 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, relative to other function statements, in the SYSIN data set indicates to the IEBUPDTE program which type of user label follows the LABEL statement:

- To create output header labels, place the LABEL statement and its associated label data statements before all other 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 all other 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 the IEBUPDTE program is to create output trailer labels.

User Label Processing With UPDATE=INPLACE

When UPDATE=INPLACE is specified, 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 F: Utility Program Handling of User Labels," for a complete discussion of the linkage between utility programs and user label processing routines. The return codes when the UPDATE=INPLACE function is used differ slightly from the standard codes discussed in Appendix F, as follows:

Routine Type	Return Code	System Response
Input header label	0	The system resumes normal processing; any additional user labels are ignored.
	4	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	The system writes the user label from the label buffer area and resumes normal processing.
	12	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 via the Keyword UPDATE=INPLACE.
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 discussed in Appendix F. 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, via return codes, the user specifies 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 becomes meaningless.

ALIAS Statement

An ALIAS statement creates or retains an alias in an output (partitioned) master directory. The ALIAS statement can be used with any of the function statements. Multiple alias names can be assigned to each member. The statement contains:

1-2	Name	Operation	Operand
./	[name]	ALIAS	NAME=cccccccc

./ (columns 1 and 2)
identifies the control statement.

name
specifies an optional name which, if coded, begins in column 3 and extends no further than column 10.

NAME=cccccccc
specifies an 8-character (maximum) alias name.

Note: At least one blank must precede and follow the operation field.

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 indication. The ENDUP statement follows the last group of SYSIN control statements and contains:

1-2	Name	Operation
./	[name]	ENDUP

./ (columns 1 and 2)
identifies the control statement

name
specifies an optional name which, if coded, begins in column 3 and extends no further than column 10.

Note: At least one blank must precede the operation field.

IEBUPDTE Examples

The following examples illustrate some of the uses of the IEBUPDTE program.

IEBUPDTE Example 1

Operation	Data Set Organization	Input Device	Output Device	Comments
ENTER a procedure into a cataloged procedure library	Input-PARTITIONED Output-PARTITIONED	DISK - 2311	DISK - 2311	1. The SYSUT1 and SYSUT2 DD statements define the same data set. 2. The new procedure is contained in the control data set.

In this example, a procedure is to be placed into the cataloged procedure library SYS1.PROCLIB. The example assumes that the new procedure (ERASE) can be accommodated within the space originally allocated to the procedure library; therefore, the update operation is performed directly to the old master.

- The SYSUT1 and SYSUT2 DD Statements: define the SYS1.PROCLIB data set.
- The SYSIN DD Statement: 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 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 NUMBER Statement: indicates that the new procedure is assigned sequence numbers. The first record of the procedure is assigned sequence number 10; the remaining two records are assigned sequence numbers 20 and 30, respectively.
- The //ERASE EXEC Statement, and the DD1 and SYSPRINT DD Statements: are placed in the cataloged procedure library SYS1.PROCLIB.

Note: The resulting procedure can be executed with an EXEC statement specifying PROC=ERASE. The publication IBM System/360 Operating System: Job Control Language Reference, GC28-6704, contains additional information on cataloged procedures.

```
//UPDATE   JOB  09#660,SMITH
//         EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=SYS1.PROCLIB,UNIT=2311,DISP=(OLD,KEEP),
//             VOLUME=SER=111111,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSUT2   DD   DSNAME=SYS1.PROCLIB,UNIT=2311,DISP=(OLD,KEEP),
//             VOLUME=SER=111111,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD   DATA
./        ADD   LIST=ALL,NAME=ERASE,LEVEL=01,SOURCE=0
./     NUMBER   NEW1=10,INCR=10
//ERASE    EXEC PGM=IEHPROGM
//DD1      DD   UNIT=190,DISP=(OLD,KEEP),VOLUME=SER=111111
//SYSPRINT DD   SYSOUT=A
/*
```

IEBUPDTE Example 1. Entering Job Control Statements Into a Cataloged Procedure Library

IEBUPDTE Example 2

Operation	Data Set Organization	Input Device	Output Device	Comments
CREATE a partitioned library	Input-N/A Output-PARTITIONED	------	DISK - 2311	1. Input data in control data set. 2. Output partitioned data set contains three members.

In this example, a three-member, partitioned library is to be created. The input data is contained solely in the control data set.

- The SYSUT2 DD Statement: defines the new partitioned master OUTLIB. Enough space is allocated to allow for subsequent modifications without creating a new master data set.

- The SYSIN DD Statement: 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 Statements: indicate that subsequent records (data statements) are to be placed as members in the output partitioned data set. Each ADD 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.

- The ENDUP Statement: signals the end of control data set input.

Note: Sequence numbers (other than blank numbers) are included within the data statements; therefore, no NUMBER statements are included in the example.

```
//UPDATE   JOB  09#770,SMITH
//         EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD   SYSOUT=A
//SYSUT2   DD   DSNAME=OUTLIB,UNIT=2311,DISP=(NEW,KEEP),
//              VOLUME=SER=111112,SPACE=(TRK,(100,,10)),
//              DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD   DATA
./         ADD   NAME=MEMB1,LEVEL=00,SOURCE=0,LIST=ALL
            .
            .
            MEMB1 data statements, sequence numbers in columns 73-80
            .
            .
./         ADD   NAME=MEMB2,LEVEL=00,SOURCE=0,LIST=ALL
            .
            .
            MEMB2 data statements, sequence numbers in columns 73-80
            .
            .
./         ADD   NAME=MEMB3,LEVEL=00,SOURCE=0,LIST=ALL
            .
            .
            MEMB3 data statements, sequence numbers in columns 73-80
            .
            .
./       ENDUP
/*
```

IEBUPDTE Example 2. Creating a Three-Member, Partitioned Library Using SYSIN Data as Input

IEBUPDTE Example 3

Operation	Data Set Organization	Input Device	Output Device	Comments
CREATE a partitioned data set	Input-PARTITIONED Output-PARTITIONED	DISK - 2311	DISK - 2311	1. Input from control data set and from existing partitioned data set. 2. Output partitioned data set contains four members.

In this example, a four-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).
- One replacement member (BLDL) for an existing member of the input partitioned data set.
- A new member (EXIT).

  The new member (EXIT) is contained in the control data set.

- The SYSUT1 DD Statement: defines the input partitioned data set SYS1.MACLIB.
- The SYSUT2 DD Statement: defines the output partitioned data set OUTLIB. Enough space is allocated to allow for subsequent modifications without creating a new master data set.
- The SYSIN DD Statement: defines the control data set.
- The REPRO 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 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 Statement: assigns sequence numbers to the data statements. (The data statements contain blank sequence numbers in columns 73-80.) The first record of the output member is assigned sequence number 10; subsequent records are incremented by 100.
- The REPL Statement: identifies a new member used as a replacement for an existing member. The subsequent NUMBER statement assigns sequence numbers to the records in the new member.
- The ENDUP Statement: signals the end of SYSIN data.

Note: The three named input members (ATTACH, DETACH, and BLDL) do not have to be specified in order of their collating sequence in the old master.

```
//UPDATE   JOB  09,770,SMITH
//         EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=SYS1.MACLIB,DISP=(OLD,KEEP),VOLUME=SER=111111,
//              UNIT=2311,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSUT2   DD   DSNAME=NEWMCLIB,DISP=(,KEEP),VOLUME=SER=111112,
//              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 the EXIT member
             .
             .
./        REPL   NAME=BLDL,LEVEL=01,SOURCE=1,LIST=ALL
./      NUMBER   NEW1=10,INCR=100
             .
             .
             data cards (replacement for existing BLDL member)
             .
             .
./       ENDUP
/*
```

IEBUPDTE Example 3. Creating a Partitioned Data Set Using SYSIN Data and Existing Data as Input

IEBUPDTE Example 4

Operation	Data Set Organization	Input Device Output Device	Comments
UPDATE INPLACE-renumber	Input-PARTITIONED Output-PARTITIONED	DISK - 2311	1. The input data set is considered to be the output data set, as well. (No SYSUT2 DD statement is required.)

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.

- The SYSUT1 DD Statement: defines the data set that is to be updated in place. (Note that the member name need not be specified in the DD statement.)
- The SYSIN DD Statement: defines the control data set.
- The CHANGE 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 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.

```
//UPDATE   JOB  09#770,SMITH
//         EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=PDS,UNIT=2311,DISP=(OLD,KEEP),VOLUME=SER=111112,
//              DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD   *
./     CHANGE   NAME=MODMEMB,LIST=ALL,UPDATE=INPLACE
./     NUMBER   SEQ1=ALL,NEW1=10,INCR=5
           data statement 1                                    00000020
           data statement 2                                    00000035
/*
```

IEBUPDTE Example 4. Updating a Member in Place

IEBUPDTE Example 5

Operation	Data Set Organization	Input Device	Output Device	Comments
CREATE sequential master from partitioned input & DELETE selected records	Input-PARTITIONED Output-SEQUENTIAL	DRUM - 2301	TAPE -9-track, standard label, 800 bits-per-inch density	1. Blocked output.

In this example, a sequential master data set is to be created from partitioned input, and selected logical records are to be deleted.

- The SYSUT1 DD Statement: defines the input partitioned data set PARTDS. Since no DCB parameters are specified in the SYSUT1 DD statement, the program assumes that the input data set consists of 80-byte, unblocked records. A warning message is issued to inform the user that this assumption has been made.

- The SYSUT2 DD Statement: defines the output sequential data set. The data set is to be written as the second data set on a 9-track magnetic tape volume. The data set is written at a density of 800 bits-per-inch.

- The SYSIN DD Statement: defines the control data set.

- The CHANGE Statement: 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 Statement: deletes logical records having sequence numbers equal to or between 223 and 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.

```
//UPDATE   JOB  09#770,SMITH
//         EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=PARTDS,UNIT=2301,DISP=(OLD,KEEP),
//             VOLUME=SER=111112
//SYSUT2   DD   DSNAME=SEQDS,UNIT=2400,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                                        00000123
./     DELETE   SEQ1=223,SEQ2=246
         data statement 2                                        00000224
/*
```

IEBUPDTE Example 5. Creating a Sequential Data Set From Partitioned Input -- Deleting Selected Records

IEBUPDTE Example 6

Operation	Data Set Organization	Input Device	Output Device	Comments
CREATE a partitioned data set from sequential input. DELETE records. UPDATE portions of records.	Input-SEQUENTIAL Output-PARTITIONED	TAPE 9-track, standard label, 800 bits-per-inch density	DISK - 2311	1. Sequence numbers in columns other than 73-80. 2. One member placed in the output partitioned data set.

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.

- The SYSUT1 DD Statement: defines the input sequential data set (OLDSEQDS). The data set was originally written at a density of 800 bits-per-inch on a 9-track magnetic tape volume.

- The SYSUT2 DD Statement: defines the output partitioned data set. Enough space is allocated to provide for members that might be added in subsequent job steps.

- The SYSIN DD Statement: defines the control data set.

- The CHANGE Statement: identifies the output member and indicates that a conversion from sequential input to partitioned output is to be made. The SEQFLD keyword indicates that a 5-byte sequence number is located in columns 60-64 of each data statement. The COLUMN keyword specifies the starting column of a field (within subsequent data statements) from which replacement information is obtained.

- The First Data Statement: Columns 40-80 of the first data statement replace columns 40-80 of the corresponding logical record. If no such logical record exists, the entire card image is inserted in the output member.

- The DELETE Statement: deletes all of the logical records having sequence numbers equal to or between 220 and 250.

- The Second Data Statement: the second data statement, whose sequence number falls within the range specified in the DELETE statement, is incorporated in its entirety in the output member.

- The Third Data Statement: the third data statement, which is beyond the range of the DELETE statement, is treated in the same manner as the first data statement.

- The ALIAS Statement: assigns the alias name MEMB1 to the output member PARMEM1.

```
//UPDATE   JOB  09#770,SMITH
//         EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=OLDSEQDS,UNIT=2400,LABEL=(,SL),
//              DISP=(OLD,KEEP),VOLUME=SER=001234,
//              DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSUT2   DD   DSNAME=NEWPART,UNIT=2311,DISP=(,KEEP),
//              VOLUME=SER=111112,SPACE=(TRK,(20,5,5)),
//              DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD   *
./     CHANGE   NEW=PO,MEMBER=PARMEM1,LEVEL=01,SOURCE=0,SEQFLD=605,    C
./              COLUMN=40
           data statement 1                                    00020
./     DELETE   SEQ1=220,SEQ2=250
           data statement 2                                    00230
           data statement 3                                    00260
./      ALIAS   NAME=MEMB1
/*
```

IEBUPDTE Example 6. Creating a Partitioned Data Set From Sequential Input -- Deleting Records -- Updating Portions of Records

IEBUPDTE Example 7

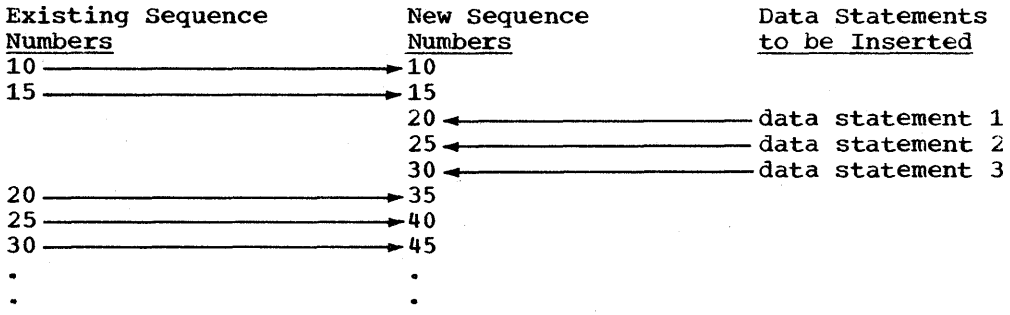
Operation	Data Set Organization	Input Device	Output Device	Comments
INSERT a block of logical records into existing member	Input-PARTITIONED Output-PARTITIONED	DISK - 2311	DISK - 2311	1. The SYSUT1 and SYSUT2 DD statements define the same data set.

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.

- The SYSUT1 and SYSUT2 DD Statements: define the partitioned data set PDS.
- The SYSIN DD Statement: defines the control data set.
- The CHANGE Statement: identifies the input member RENUM. The entire member is to be listed in the message data set.
- The NUMBER Statement: specifies the insert operation (INSERT=YES) and controls the renumbering operation.
- The Data Statements: are the logical records to be inserted. they contain blank sequence numbers. (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 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.



```
//UPDATE   JOB  09#770,SMITH
//         EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=PDS,UNIT=2311,DISP=(OLD,KEEP),
//              VOLUME=SER=111112,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSUT2   DD   DSNAME=PDS,UNIT=2311,DISP=(OLD,KEEP),
//              VOLUME=SER=111112,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//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
/*
```

IEBUPDTE Example 7. Inserting a Block of Logical Records Into an Existing Member

IEBUPDTE Example 8

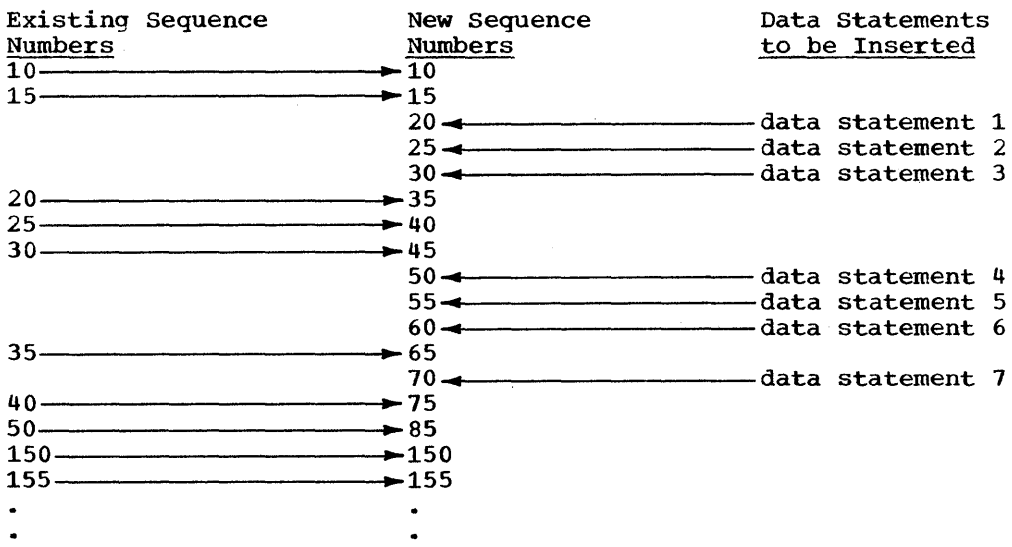
Operation	Data Set Organization	Input Device	Output Device	Comments
INSERT two blocks of logical records into existing member	Input-PARTITIONED Output-PARTITIONED	DISK - 2311	DISK - 2311	1. The SYSUT1 and SYSUT2 DD statements define the same data set.

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.

- The SYSUT1 and SYSUT2 DD Statements: define the partitioned data set PDS.
- The SYSIN DD Statement: defines the control data set.
- The CHANGE Statement: identifies the input member RENUM. The entire member is to be listed in the message data set.
- The NUMBER 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. They contain blank sequence numbers. (Sequence numbers are assigned when the data statements are inserted.)
- Data Statement 7: is a logical record to be inserted in the output member.

The existing logical records in this example have sequence numbers 10, 15, 20, 25, 30, 35, 40, 45, 50, 150, 155, 160, 165, etc. The insert and renumbering operations are performed 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. Data statement 4 is assigned sequence number 50 and inserted. (The SEQ1=30 specification in the second NUMBER statement places this data statement after existing logical record 30, which has become logical record 45.)
5. Data statements 5 and 6 are assigned sequence numbers 55 and 60 and are inserted after data statement 4.
6. Existing logical record 35 is assigned sequence number 65.
7. Data statement 7 is assigned sequence number 70 and is inserted.
8. The remaining logical records in the member are renumbered until logical record 150 is encountered. Since this record has a sequence number higher than the next number to be assigned, the renumbering operation is terminated.




```
//UPDATE   JOB  09#770,SMITH
//         EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=PDS,UNIT=2311,DISP=(OLD,KEEP),
//             VOLUME=SER=111112,DCB=(RECFM=F,BLKSIZE=80,LRECL=80)
//SYSUT2   DD   DSNAME=PDS,UNIT=2311,DISP=(OLD,KEEP),
//             VOLUME=SER=111112,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//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
./     NUMBER   SEQ1=30,INSERT=YES
           data statement 4
           data statement 5
           data statement 6
           data statement 7                                          00000038
/*
```

IEBUPDTE Example 8. Inserting Blocks of Logical Records Into an Existing Member

IEBUPDTE Example 9

Operation	Data Set Organization	Input Device	Output Device	Comments
Create a Sequential data set, with user labels, from card input.	Input-SEQUENTIAL Output-SEQUENTIAL	Card reader	DISK - 2311	1. Both user labels and data are in card form in input stream.

In this example, the IEBUPDTE program creates a sequential data set from card input. User header and trailer labels, also from the input stream, are placed on this sequential data set.

- The SYSUT2 DD Statement: defines and allocates space for the output sequential data set, which resides on a 2311 disk.

- The SYSIN DD Statement: 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 header label processing routine specified on the ADD statement.)

- The ADD Statement: indicates that the records (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 trailer label processing routine specified on the ADD statement.)

- The ENDUP Statement: signals the end of the control data set.

```
//LABEL    JOB  ,MSGLEVEL=1
//CREATION EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD   SYSOUT=A
//SYSUT2   DD   DSNAME=LABEL,VOLUME=SER=123456,
//              DISP=(NEW,KEEP),LABEL=(,SUL),
//              SPACE=(TRK,(15,3)),UNIT=2311
//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
/*
```

IEBUPDTE Example 9. Creating a Sequential Data Set From Card Input

IEBUPDTE Example 10

Operation	Data Set Organization	Input Device	Output Device	Comments
Copy a sequential data set from one direct access volume to another, permitting the user to process user labels with exit routines	Input-SEQUENTIAL Output-SEQUENTIAL	DISK - 2311	DISK - 2311	1. The input data set is simply copied onto the output data set, but user label exit routines are permitted to inspect and, if desired, change labels on both the input and output data sets.

In this example, the IEBUPDTE program copies a sequential data set from one direct access volume to another. User labels are processed by user exit routines.

- The SYSUT1 DD Statement: defines the input sequential data set, which resides on a 2311 disk.
- The SYSUT2 DD Statement: defines the output sequential data set, which will reside on a 2311 disk.
- The SYSIN DD Statement: defines the control data set.
- The REPRO Statement: indicates that the existing input sequential data set is to be copied onto the output data set. This output data set is to be listed on the message data set. The user label processing routines are to be given control when header or trailer labels are encountered on either the input or the output data set.
- The ENDUP Statement: indicates the end of the control data set.

```
//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=2311
//SYSUT2   DD   DSNAME=NEWMAST,DISP=(NEW,KEEP),LABEL=(,SUL),
//              UNIT=2311,VOLUME=SER=XB182,SPACE=(TRK,(10,10))
//SYSIN    DD   DSNAME=INPUT,DISP=OLD,LABEL=(,SUL),
//              VOLUME=SER=222222,UNIT=2311
/*
                    INPUT DATA SET
./       REPRO  LIST=ALL,INHDR=SSSSSS,INTLR=TTTTTT,OUTHDR=XXXXXX,      C
./              OUTTLR=YYYYYY
./       ENDUP
```

IEBUPDTE Example 10. Copying a Sequential Data Set With User Labels

Program Applications

The IEBISAM program can copy an indexed sequential data set directly from one direct access volume to another. Alternatively, the IEBISAM program can reorganize an indexed sequential data set into a sequential (unloaded) data set and place that data set on a direct access volume or on a magnetic tape volume. The unloaded data set is in a form that can be subsequently loaded; that is, it can be converted back into an indexed sequential data set.

Optionally, the IEBISAM program can be used to print the records of an indexed sequential data set.

The program is used to:

- Copy an indexed sequential data set (COPY operation).
- Create a sequential back-up (transportable) copy of source data from an indexed sequential data set (UNLOAD operation).
- Create an indexed sequential data set from an unloaded data set (LOAD operation).
- Print an indexed sequential data set (PRINTL operation).

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

Copying an Indexed Sequential Data Set (COPY Operation)

The IEBISAM program can copy an indexed sequential data set directly from one direct access volume to another. Records marked for deletion are automatically deleted when the data set is copied. 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.

Creating a Sequential Back-Up Copy (UNLOAD Operation)

An "unloaded" sequential data set can be created to serve as a back-up 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), those records that were contained in the overflow area assigned to the original data set are moved sequentially into the primary area.

Unloaded Data Sets: An unloaded data set consists of 80-byte logical records. The data set contains:

- Fixed-length 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).

When the LOAD operation is specified, these characteristics can be overridden by new 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, since checks are made to ensure that:

1. The record format is the same as that of the original indexed sequential data set (either fixed or variable length).

2. The logical record length is less than or equal to that of the original indexed sequential data set.

3. For fixed length 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 length records, the blocksize is equal to or greater than the logical record length plus four.

4. The relative key position is equal to or less than the logical record length minus the key length.

5. The key length is less than or equal to 255 bytes.

Caution: If either RKP or KEYLEN is overridden, it may 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:

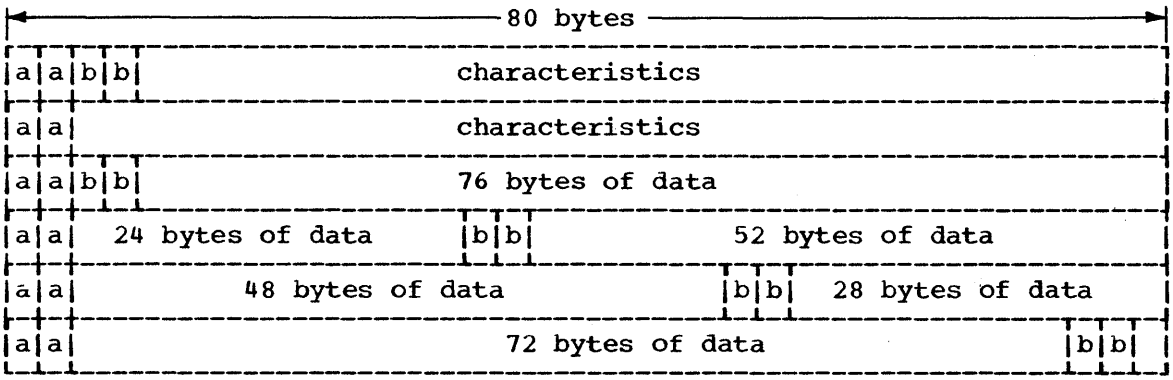
$$x = \frac{n(l+2) + 158}{78}$$

Where:

n = the number of records in the indexed sequential data set.
l = the length of a fixed-length record or the average length of the variable length records.
x = the number of 80-byte logical records created by the utility program.

IEBISAM Figure 1 shows the format of an unloaded data set for the first three fixed-length records (each 100 bytes long) of an indexed sequential data set. Each fixed-length record is preceded by two bytes (bb) that indicate the number of bytes in that record. (The last fixed-length 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.



IEBISAM Figure 1. An Unloaded Data set

Creating an Indexed Sequential Data Set From an Unloaded Data Set (LOAD Operation)

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 (PRINTL Operation)

The records of an indexed sequential data set can be printed or stored as a sequential data set for subsequent printing.

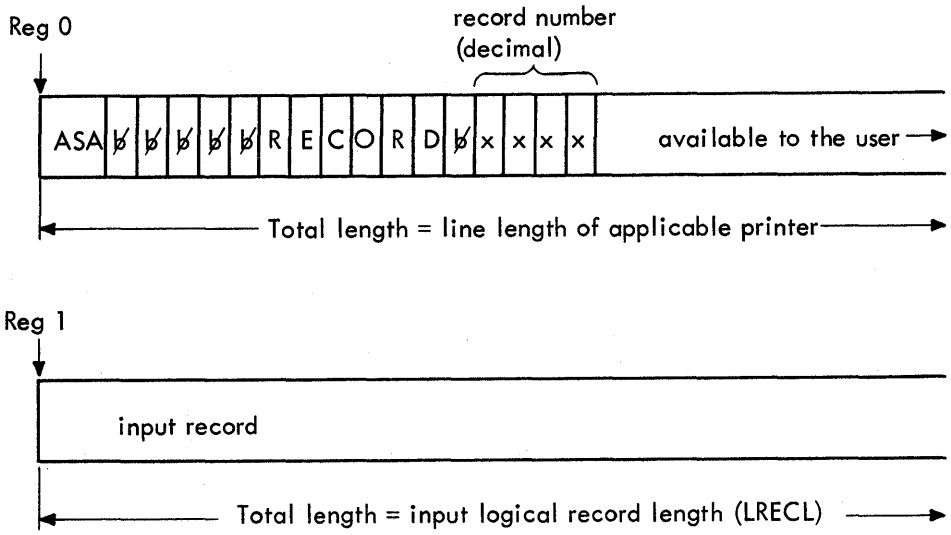
Each input record not marked for deletion is placed in a buffer from which it is printed or placed in a sequential data set. Each printed record is converted to hexadecimal unless specified otherwise by the user.

The IEBISAM program provides user exits so that the user can optionally include his own routines to:

- Modify records prior to 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 no user routines are provided, each record is identified in sequential order on the printout.

User Routines: Upon determining that a user routine is supplied for a PRINTL operation, the IEBISAM program 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."

These buffers contain:



The user returns control to the IEBISAM program by issuing a RETURN macro instruction (via Register 14) or by using a BR 14 instruction after restoring Registers 2 through 14. (Note: The user must save Registers 2 through 14 when control is given to the user routine.)

A user routine must place a return code in Register 15 prior to returning control to the IEBISAM program. The return code instructs the IEBISAM program how to handle records to be printed. The possible return codes and their meanings to the IEBISAM program are:

00 -- print the buffers.

04 -- print the buffers and terminate the operation.

08 -- continue processing; do not print this input record.

12 -- terminate the operation; do not print this input record.

Inputs and Outputs

The input to and output from the IEBISAM program is dependent upon the operation to be performed. IEBISAM Table 1 lists the major inputs to and outputs from the IEBISAM program for the COPY, UNLOAD, LOAD, and PRINTL operations.

IEBISAM Table 1. Data Sets Used (Input) and Produced (Output) by the IEBISAM Program

COPY operation	Input	Input Data Set: This data set is the indexed sequential data set that is to be copied.
	Outputs	Output Data Set: This data set is the result of the COPY operation. The data set is a reorganized, indexed sequential copy of the original data set. Message Data Set: This data set is a sequential data set containing informational messages and error messages, if applicable.
UNLOAD operation	Input	Input Data Set: This data set is the indexed sequential data set that is to be unloaded.
	Outputs	Output Data Set: This data set is the result of the UNLOAD operation. The data set is sequential, and it can be placed on a magnetic tape volume or a direct access volume. Message Data Set: This data set is a sequential data set containing informational messages and error messages, if applicable.
LOAD operation	Input	Input Data Set: This data set is an unloaded sequential version of an indexed sequential data set.
	Outputs	Output Data Set: This data set is the result of the LOAD operation. The data set is an indexed sequential data set residing on a direct access volume. Message Data Set: This data set is a sequential data set containing informational messages and error messages, if applicable.
PRINTL operation	Input	Input Data Set: This data set is an indexed sequential data set that is to be printed in logical sequence.
	Outputs	Output Data Set: This is a sequential data set that contains the logical records of an indexed sequential data set. Message Data Set: This data set is a sequential data set containing informational messages and error messages, if applicable.

ADDITIONAL OUTPUTS

The IEBISAM program provides a return code to indicate the results of program execution. The return codes and their interpretations are as follows:

00 -- successful completion.
04 -- a return code of 04 or 12 was passed to the IEBISAM program by a user routine.
08 -- the program terminated operation because an error condition was encountered during processing.
12 -- a return code other than 00, 04, 08, or 12 was passed from a user routine to the IEBISAM program. The job step is terminated.
16 -- the program terminated operation because an error condition was encountered during processing.

Control

The IEBISAM program is controlled by job control statements. No utility control statements are required.

JOB CONTROL STATEMENTS

IEBISAM Table 2 shows the job control statements necessary for executing or invoking the IEBISAM program.

IEBISAM Table 2. Job Control Statements for the IEBISAM Program

Statement	Usage
JOB Statement	This statement initiates the job.
EXEC Statement	This statement specifies the program name (PGM=IEBISAM). In addition, this statement specifies PARM={COPY / UNLOAD / LOAD / PRINTL or 'PRINTL,[N,] [EXIT=routinename]'} where COPY specifies the copy operation. UNLOAD specifies the unload operation. LOAD specifies the load operation. PRINTL specifies a print operation in which each record is converted to hexadecimal prior to printing. No user routines are provided. 'PRINTL,[N,] [EXIT=routinename]' specifies a print operation. If N is included, no conversion to hexadecimal is performed by the IEBISAM program. EXIT=routinename, if included, indicates that a user routine is to be given control prior to printing each record. (Exit routines must be included in either the job library or the link library.)
SYSUT1 DD statement	This statement defines the input data set.
SYSUT2 DD statement	This statement defines the output data set.
SYSPRINT DD statement	This statement defines a sequential message data set. The data set can be written onto a system output device (e.g., a printer), a magnetic tape volume, or a direct access device.

Note (COPY operation): The SYSUT2 DD statement must include a primary space allocation that is sufficient to accomodate 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.

Notes (UNLOAD operation): Specifications that are implied by default or included in the DCB parameter of the SYSUT2 DD statement (e.g., 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.

Notes (LOAD operation): If the input data set resides on an unlabeled tape, the SYSUT1 DD statement must specify a BLKSIZE of 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 accomodate 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.

Note (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.

IEBISAM Examples

The following examples illustrate some of the uses of the IEBISAM program.

IEBISAM Example 1

Operation	Data Set Organization	Input Device	Output Device	Comments
COPY	Input--IX SEQUENTIAL Output--SEQUENTIAL	2 DISK - 2311	2 DISK - 2311	1. Unblocked input - blocked output. 2. Prime and index separation.

In this example, an indexed sequential data set is to be copied from two 2311 disk volumes to two other 2311 disk volumes. The output data is blocked.

- The EXEC Statement: specifies the program name and the COPY operation.
- The SYSUT1 DD Statement: defines the input (indexed sequential) data set. The data set resides on two 2311 disk volumes.
- The SYSUT2 DD Statement: defines the output data set index area. (The index and prime areas are separated.)
- The // DD Statement: defines the output data set prime area. Ten cylinders are allocated for the prime area on each of the two 2311 disk volumes.

```
//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=(2311,2),
//               DCB=(DSORG=IS,LRECL=500,BLKSIZE=500)
//SYSUT2    DD   DSNAME=ISAM02(INDEX),UNIT=2311,DISP=(NEW,KEEP),
//               DCB=(DSORG=IS,BLKSIZE=1000),VOLUME=SER=444444,
//               SPACE=(CYL,(2))
//          DD   DSNAME=ISAM02(PRIME),UNIT=(2311,2),DISP=(NEW,KEEP),
//               DCB=(DSORG=IS,BLKSIZE=1000),SPACE=(CYL,(10)),
//               VOLUME=SER=(444444,555555)
/*
```

IEBISAM Example 1. Copying an Indexed Sequential Data Set

IEBISAM Example 2

Operation	Data Set Organization	Input Device	Output Device	Comments
UNLOAD	Input-IX-SEQUENTIAL Output-SEQUENTIAL	DISK - 2311	TAPE - 9-track, standard label, 800 bits-per-inch density	1. Blocked output.

In this example, indexed sequential input is to be converted into a sequential data set and the output is to be placed on a 9-track magnetic tape volume.

- The EXEC Statement: specifies the program name (IEBISAM) and the UNLOAD operation.
- The SYSUT1 DD Statement: defines the input (indexed sequential) data set. The data set resides on a 2311 Disk Storage Drive.
- The SYSUT2 DD Statement: defines the output (unloaded) data set. The data set consists of fixed-length, blocked records, and is to reside as the first or only data set on a 9-track, magnetic tape volume. The data set is to be written at a density of 800 bits-per-inch.

```
//STEP1    JOB  09#770,SMITH
//         EXEC PGM=IEBISAM,PARM=UNLOAD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=INDSEQ,UNIT=2311,DISP=(OLD,KEEP),
//              VOLUME=SER=111112
//SYSUT2   DD   DSNAME=UNLDSET,UNIT=2400,LABEL=(,SL),DISP=(,KEEP),
//              VOLUME=SER=001234,DCB=(RECFM=FB,LRECL=80,BLKSIZE=640)
/*
```

IEBISAM Example 2. Unloading an Indexed Sequential Data Set Onto 9-Track Magnetic Tape

IEBISAM Example 3

Operation	Data Set Organization	Input Device	Output Device	Comments
UNLOAD	Input-IX-SEQUENTIAL Output-SEQUENTIAL	DISK - 2311	TAPE 7-track, standard label, 800 bits-per-inch density, data conversion	1. Blocked output. 2. Data set written as second data set on output volume.

In this example, indexed sequential input is to be converted into a sequential data set and the output is to be placed on a 7-track, magnetic tape volume.

- The EXEC Statement: specifies the program name (IEBISAM) and the UNLOAD operation.
- The SYSUT1 DD Statement: defines the input (indexed sequential) data set. The data set resides on a 2311 Disk Storage Drive.
- The SYSUT2 DD Statement: defines the output (unloaded) data set. The data set consists of fixed-length, blocked records, and is to reside as the second data set on a 7-track magnetic tape volume. The data set is to be written at a density of 800 bits-per-inch.

```
//STEPA    JOB  09#770,SMITH
//         EXEC PGM=IEBISAM,PARM=UNLOAD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=INDSEQ,UNIT=2311,DISP=(OLD,KEEP),
//              VOLUME=SER=111112
//SYSUT2   DD   DSNAME=UNLDSET,UNIT=2400-2,LABEL=(2,SL),DISP=(,KEEP),
//              VOLUME=SER=001234,DCB=(DEN=2,RECFM=FB,LRECL=80,
//              BLKSIZE=1040,TRTCH=C)
/*
```

IEBISAM Example 3. Unloading an Indexed Sequential Data Set Onto 7-Track Magnetic Tape

IEBISAM Example 4

Operation	Data Set Organization	Input Device	Output Device	Comments
LOAD	Input-SEQUENTIAL Output-IX-SEQUENTIAL	TAPE 9-track, standard label, 800 bits-per-inch density	DISK - 2311	1. Input data set is second data set on magnetic tape volume.

In this example, an unloaded data set is to be converted into the form of the original indexed sequential data set.

- The EXEC Statement: specifies the program name (IEBISAM) and the LOAD operation.
- The SYSUT1 DD Statement: defines the input sequential (unloaded) data set. The data set is the second data set on a 9-track magnetic tape volume.
- The SYSUT2 DD Statement: defines the output (indexed sequential) data set. One cylinder of space is allocated for the data set on a 2311 Disk Storage Drive.

```
//STEPA    JOB  09#770,SMITH
//         EXEC PGM=IEBISAM,PARM=LOAD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=UNLDSET,UNIT=2400,LABEL=(2,SL),
//              DISP=(OLD,KEEP),VOLUME=SER=001234
//SYSUT2   DD   DSNAME=INDSEQ,UNIT=2311,DISP=(,KEEP),SPACE=(CYL,(1)),
//              VOLUME=SER=111112,DCB=(DSORG=IS)
/*
```

IEBISAM Example 4. Reconstructing an Indexed Sequential Data Set

IEBISAM Example 5

Operation	Data Set Organization	Input Device	Output Device	Comments
PRINTL	Input--IX SEQUENTIAL Output--SEQUENTIAL	DISK - 2311	System Output Device (Printer Assumed)	1. Blocked input. 2. Output not converted.

In this example, the logical records of an indexed sequential data set are to be printed on a system output device.

- The EXEC Statement: specifies the program name and the PRINTL operation. The output records are not converted to hexadecimal prior to printing.
- The SYSUT1 DD Statement: defines the input indexed sequential data set. The data set resides on a 2311 disk volume.
- The SYSUT2 DD Statement: defines the output (printed) sequential data set. A logical record length (LRECL) of 121 bytes is assumed.

```
//PRINT    JOB  09#770,SMITH
//         EXEC PGM=IEBISAM,PARM='PRINTL,N'
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=ISAM03,UNIT=2311,DISP=OLD,
//             VOLUME=SER=222222
//SYSUT2   DD   SYSOUT=A
/*
```

IEBISAM Example 5. Printing Logical Records From an Indexed Sequential Data Set

Program Applications

The IEBEDIT data set utility program can 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.

Input to the IEBEDIT program is obtained from a sequential data set. The input data set can reside on any IBM supported input device (e.g., magnetic tape, direct access, or card reader). The program can select JOB statements, JOBLIB statements, and job steps from the input data set and can include them in the output data set.

Selectively Copying a Job Stream

The IEBEDIT program 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 associated JOBLIB statements, if any.
- Portions of a job (i.e., selected job steps), including the JOB statement and its associated JOBLIB statement, if any.

All selected JOB statements, JOBLIB statements, jobs, or job steps are placed in the output data set in the same order as they exist in the input data set.

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

Note: A JOBLIB statement is copied only if it follows a selected JOB statement.

IEBEDIT Figure 1 shows a few examples of how the program can selectively copy a job stream provided as input to the IEBEDIT program.

INPUT	SOME OF THE POSSIBLE OUTPUTS			
JOBA STEPA STEPB STEPC STEPD JOBB STEPE STEPF STEPG JOBC JOBLIB STEPH STEPJ	JOBA STEPA STEPB STEPC	JOBA STEPC JOBB STEPE JOBC JOBLIB STEPJ	JOBB STEPG JOBC JOBLIB STEPH STEPJ	JOBC JOBLIB STEPJ

IEBEDIT Figure 1. Selectively Copying a Job Stream

Inputs and Outputs

IEBEDIT Table 1 shows the major inputs to and outputs from the IEBEDIT program.

IEBEDIT Table 1. Data Sets Used (Input) and Produced (Output) by the IEBEDIT Program

Inputs	Input Data Set: This data set is a sequential data set consisting of a job stream. The data set is used as source data in creating an output sequential data set. Control Data Set: This data set contains utility control statements used to specify the organization of jobs and job steps in the output data set.
Outputs	Output Data Set: This data set is a sequential data set consisting of a resultant job stream. Message Data Set: This data set is a sequential data set consisting of: • A listing of applicable control statements. • A listing of the output data set (optional). • Error messages, if applicable.

ADDITIONAL OUTPUTS

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

00 -- successful completion.

04 -- an error occurred. The output data set may or may not be usable as a job stream. Processing continues.

08 -- an unrecoverable error occurred while attempting to process the input, output, or control data set. The job step is terminated.

Control

The IEBEDIT program 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

IEBEDIT Table 2 shows the job control statements necessary for the execution or invocation of the IEBEDIT program.

IEBEDIT Table 2. Job Control Statements for the IEBEDIT Program

Statement	Usage
JOB statement	This statement initiates the job.
EXEC statement	This statement specifies the program name (PGM=IEBEDIT) or, if the job control statements for the IEBEDIT program reside in a procedure library, the procedure name.
SYSPRINT DD statement	This statement 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 statement	This statement defines the input data set. It can define a sequential data set on a card reader, a magnetic tape volume, or a direct access device.
SYSUT2 DD statement	This statement defines the output data set. It can define a sequential data set on a card punch, a printer, a magnetic tape volume, or a direct access device. Space must be allocated for an output data set that is to reside on a direct access volume.
SYSIN DD statement	This statement defines the control data set. The data set normally is included in the input stream; however, it can alternatively be defined as being a member of a procedure library or as being a sequential data set existing somewhere other than in the input stream.
The blocksize for the SYSPRINT (message) data set must be a multiple of 121. The blocksize for the SYSIN (control) data set must be a multiple of 80. Any blocking factor can be specified for these blocksizes. The SYSUT1 (input) and SYSUT2 (output) data sets can also have any blocking factor.	

The IEBEDIT program is controlled through the use of an EDIT utility control statement.

The 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.

Name	Operation	Operand
[name]	EDIT	[START=jobname] [TYPE={POSITION / INCLUDE / EXCLUDE}] [STEPNAME=({name / name-name}[,{name / name-name}],...)] [NOPRINT]

START=jobname
: specifies the name of the input job to which the EDIT statement applies.

  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.

TYPE=POSITION
: specifies that the output is to consist of a JOB statement, the job step specified in the STEPNAME keyword, and all steps that follow it. All job steps preceding the specified step are omitted from the operation.

TYPE=INCLUDE
: specifies that the output data set is to contain a JOB statement and all job steps specified in the STEPNAME keyword.

TYPE=EXCLUDE
: specifies that the output data set is to contain a JOB statement and all jobs steps belonging to job except those steps specified in the STEPNAME keyword.

  If the TYPE keyword is omitted, TYPE=POSITION is assumed.

STEPNAME
: With TYPE=POSITION: STEPNAME=name
  specifies the first job step to be placed in the output data set. Job steps preceding this step are not copied into the output data set.

With TYPE=INCLUDE or TYPE=EXCLUDE:

STEPNAME=({name / name-name} [,{name / name-name}],...)

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 and including STEPL, and STEPZ are to be included in or excluded from the operation.

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.

NOPRINT

specifies that the message data set is not to include a listing of the output data set.

If NOPRINT is omitted, the resultant output is listed in the message data set.

Note: Any JOBLIB DD statement that follows a selected JOB statement is automatically copied into the output data set.

IEBEDIT Examples

The following examples show some of the ways in which data sets to be used as job streams can be constructed.

IEBEDIT Example 1

Operation	Input Data Set Contains	Comments
Copy JOBA into the output data set.	JOBA STEPA STEPB STEPC STEPD JOBB STEPE STEPF STEPG JOBC STEPH STEPJ	1. The input data set resides on a labeled 9-track (800 bpi) magnetic tape volume. 2. The output data set resides on a labeled 9-track (800 bpi) magnetic tape volume.

This example copies JOBA, including all of its job steps (A, B, C, and D), into the output data set.

- The SYSUT1 DD Statement: defines the input data set. The data set resides on a 9-track magnetic tape volume (001234) having standard labels.
- The SYSUT2 DD Statement: defines the output data set. The data set is to reside as the first data set on a 9-track magnetic tape volume (001235) having standard labels.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream.
- The EDIT Utility Control Statement: indicates that JOBA is to be copied in its entirety.

```
//EDIT1    JOB  09#440,SMITH
//         EXEC PGM=IEBEDIT
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=2400,DISP=(OLD,KEEP),VOLUME=SER=001234,
//              DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSUT2   DD   UNIT=2400,DISP=(NEW,KEEP),VOLUME=SER=001235,
//              DCB=(RECFM=F,LRECL=80,BLKSIZE=80),DSNAME=OUTTAPE
//SYSIN    DD   *
         EDIT  START=JOBA
/*
```

IEBEDIT Example 1. Placing a Job in an Output Data Set

IEBEDIT Example 2

Operation	Input Data Set Contains	Comments
Copy one job step from each of three jobs	JOBA STEPA STEPB STEPC STEPD JOBB STEPE STEPF STEPG JOBC STEPH STEPJ	1. The input data set resides as the first or only data set on a labeled, 7-track (556 bpi) magnetic tape volume. 2. The output data set is to reside as the second data set on a labeled, 7-track (556 bpi) magnetic tape volume.

This example copies:

- The JOBA JOB statement and STEPC from that job.
- The JOBB JOB statement and STEPE from that job.
- The JOBC JOB statement and STEPJ from that job.

- The SYSUT1 DD Statement: defines the input data set. The data set resides on a 7-track magnetic tape volume (001234) having standard labels.
- The SYSUT2 DD Statement: defines the output data set. The data set is to reside as the second data set on a 7-track magnetic tape volume (001235) having standard labels.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream.
- The EDIT Utility Control Statements: copy the indicated JOB statements and job steps.

```
//EDIT2    JOB  09#440,SMITH
//         EXEC PGM=IEBEDIT
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=2400-2,DISP=(OLD,KEEP),VOLUME=SER=001234,
//              DCB=(DEN=1,RECFM=F,LRECL=80,BLKSIZE=80,TRTCH=C)
//SYSUT2   DD   DSNAME=OUTSTRM,UNIT=2400-2,DISP=(NEW,KEEP),LABEL=(2,SL),
//              DCB=(DEN=1,RECFM=F,LRECL=80,BLKSIZE=80,TRTCH=C)
//SYSIN    DD   *
         EDIT  START=JOBA,TYPE=INCLUDE,STEPNAME=STEPC
         EDIT  START=JOBB,TYPE=INCLUDE,STEPNAME=STEPE
         EDIT  START=JOBC,TYPE=INCLUDE,STEPNAME=STEPJ
/*
```

IEBEDIT Example 2. Placing Selected JOB Statements and Job Steps in an Output Data Set

IEBEDIT Example 3

Operation	Input Data Set Contains	Comments
Include and exclude job steps	JOBA STEPA STEPB STEPC STEPD JOBB STEPE STEPF STEPG JOBC STEPH STEPJ	1. The input data set resides on a 2311 disk volume. 2. The output data set is to reside as the first or only data set on an unlabeled, 9-track (800 bpi) magnetic tape volume.

This example copies:

- The JOBB JOB statement and STEPG from that job.
- The JOBC JOB statement and STEPH from that job (STEPJ is excluded).

- The SYSUT1 DD Statement: defines the input data set. The data set resides on a 2311 disk volume (231100).
- The SYSUT2 DD Statement: defines the output data set. The data set is to reside as the first or only data set on an unlabeled 9-track (800 bpi) magnetic tape volume.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream.
- The EDIT Statements: copy selected JOB statements and job steps.

```
//EDIT3    JOB  09#440,SMITH
//         EXEC PGM=IEBEDIT
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=INSET,UNIT=2311,DISP=(OLD,KEEP),
//              VOLUME=SER=231100
//SYSUT2   DD   DSNAME=OUTTAPE,UNIT=2400,LABEL(N,NL),
//              DISP=(,KEEP),DCB=(DEN=2,RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD   *
         EDIT   START=JOBB,TYPE=INCLUDE,STEPNAME=STEPG
         EDIT   START=JOBC,TYPE=EXCLUDE,STEPNAME=STEPJ
/*
```

IEBEDIT Example 3. Including and Excluding Job Steps

IEBEDIT Example 4

Operation	Input Data Set Contains	Comments
Copy the latter portion of a job stream.	JOBA STEPA STEPB STEPC STEPD STEPE STEPF STEPG STEPH STEPJ STEPK STEPL	1. The input data set resides on a 2314 disk volume. 2. The output data set is to reside on a 2314 disk volume.

This example copies the JOBA JOB statement, the job step STEPF, and all the steps that follow it. Job steps STEPA through STEPE are not included in the output data set.

- The SYSUT1 DD Statement: defines the input data set. The data set resides on a 2314 disk volume (231400).
- The SYSUT2 DD Statement: defines the output data set. The data set is to reside on a 2314 disk volume (231401). Two tracks are allocated for the output data set.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream.
- The EDIT Statement: copies the JOB statement and job steps STEPF through STEPL.

```
//EDIT4    JOB  09#440,SMITH
//         EXEC PGM=IEBEDIT
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=INSTREAM,UNIT=2314,DISP=(OLD,KEEP),
//              VOLUME=SER=231400
//SYSUT2   DD   DSNAME=OUTSTREM,UNIT=2314,DISP=(,KEEP),SPACE=(TRK,2)
//              VOLUME=SER=231401,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD   *
         EDIT   START=JOBA,TYPE=POSITION,STEPNAME=STEPF
/*
```

IEBEDIT Example 4. Placing the Latter Portion of a Job Stream in the Output Data Set

IEBEDIT Example 5

Operation	Input Data Set Contains	Comments
Copy job control statements from the input stream.	Job stream	1. All records in the defined input data set are copied into the output data set; however, the record containing ..* in column 1-3 is converted to a termination (/*) statement in the output stream. 2. The output data set resides on a labeled 9-track magnetic tape volume.

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

- The SYSUT2 DD Statement: defines the output data set. The data set is to reside as the first data set on a 9-track magnetic tape volume (001234).
- The SYSIN DD Statement: defines a dummy control data set. This statement must precede the SYSUT1 DD statement.
- The SYSUT1 DD Statement: defines the input data set, which follows in the input stream. The job is terminated when the termination statement (/*) is encountered.

```
//EDIT5    JOB
//         EXEC PGM=IEBEDIT
//SYSPRINT DD   SYSOUT=A
//SYSUT2   DD   DSNAME=OUTTAPE,UNIT=2400,DISP=(NEW,KEEP),
//              VOLUME=SER=001234,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD   DUMMY
//SYSUT1   DD   DATA
//BLDGDGIX JOB
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=2311,VOLUME=SER=111111,DISP=OLD
//SYSIN    DD   *
         BLDG  INDEX=A.B.C,ENTRIES=10,EMPTY
..*
/*
```

IEBEDIT Example 5. Copying a Job From the Input Stream

The IEBUPDAT Program

The IEBUPDAT utility program incorporates both IBM- and user-generated source language modifications into symbolic libraries. (A symbolic library is a partitioned data set containing 80-byte records, such as SYS1.PROCLIB, and SYS1.MACLIB.) It is executed or invoked with the symbolic name IEBUPDAT. The program can:

- Add, copy, and replace members.
- Add, delete, replace, and renumber the records within an existing membr.
- Assign sequence numbers to the records of a new member.

The IEBUPDAT program uses two input data sets. A DD statement named SYSUT1 defines an old-master partitioned data set. The DD statement named SYSIN defines a sequential data set containing all of the transactions that are to be applied to the old master.

IEBUPDAT also uses two output data sets. A DD statement named SYSUT2 defines a new-master partitioned data set. The DD statement named SYSPRINT defines a sequential data set that reflects either the latest changes applied to the old master or the entire new master.

The input data set defined by SYSUT1 and the output data set defined by SYSUT2 can contain either blocked or unblocked records with a logical record length of 80 bytes. The output data set can have a blocking factor different from the input data set.

Notes: If the DD statements SYSUT1 and SYSUT2 define the same data set, the user can make modifications to the old master without creating a new master.

If enough space cannot be allocated for reblocked output records, the update request will be terminated.

EXEC Statement Control Information

The IEBUPDAT program obtains control information through the EXEC statement and the SYSIN data set. The EXEC statement for this program should contain the parameter:

```
PARM=(input,inhdr,intlr)
```

input
specifies either NEW or MOD, as follows:

Input	Meaning
NEW	The input consists of the SYSIN data set.*
MOD	The input consists of both the SYSIN and SYSUT1 data sets.

*Note: The SYSUT1 data set need not be defined if NEW is specified.

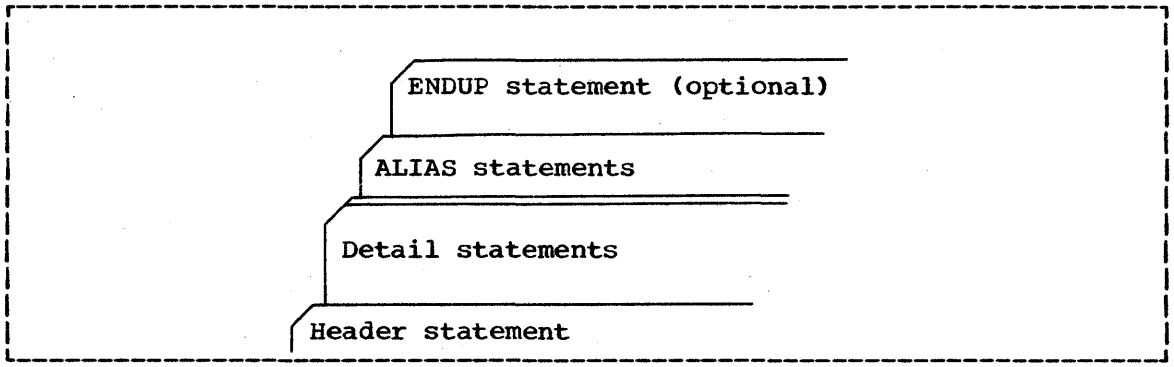
If the input is any other value, an error is indicated and the operation terminated. If an input is not specified, MOD is assumed.

inhdr
specifies the symbolic name of a routine that processes the user header label on the SYSIN data set.

intlr
specifies the symbolic name of a routine that processes the user trailer label on the SYSIN data set.

If a value is omitted from the PARM field, its absence must be indicated by a comma (e.g., PARM=(input,,intlr).

All other control information needed to execute the IEBUPDAT program is entered through the SYSIN data set. This information is supplied on a header statement and one more detail statements and ALIAS statements. An ENDUP statement can be used to indicate the end of the SYSIN input to this program. IEBUPDAT Figure 1 illustrates the order of the SYSIN control statements.



IEBUPDAT Figure 1. SYSIN Control Statements

The SYSIN data set can contain any number of header statements and ALIAS statements, each followed by a group of detail statements and ALIAS statements.

Header Statement

A header statement must be provided for each member to be processed. The statements must be in binary collating sequence by member name. The header statement contains:

Columns	Contents
1-2	Period-slash (./)
10	{ADD, REPL, CHNGE, REPRO}
16	membername,level, source,list,ssi

ADD
indicates that the named member is to be added in its entirety to the new master.

REPL
indicates that the named member is being entered in its entirety as a replacement for a member in the old master.

CHNGE
indicates that modifications are to be made within the named member.

REPRO
indicates that the entire named member is to be copied onto the new master. Members are deleted from a library by being omitted from a series of REPRO operations.

The following parameters, separated by commas, are written beginning in column 16.

membername
specifies the name of the member to which the update transactions are to be applied.

level
specifies the current run number, a 2-digit number from 00 to 99.

source
specifies either 0 or 1, as follows:
0 - indicates user modifications.
1 - indicates IBM modifications.

list
specifies either 0 or 1, as follows:
0 - indicates that the SYSPRINT data set is to contain only modifications and control statements.
1 - indicates that the SYSPRINT data set is to contain the entire updated member and control statements.

ssi
specifies eight hexadecimal characters of new system status index information that is to be placed in the directory of the new master as the first four hexadecimal bytes of user data. If ssi information is not specified, the user data is copied as it exists in the directory of the old master. System status index information is discussed in detail in the publication IBM System/360 Operating System: Maintenance, GC27-6918.

Detail Statements

The detail statements contain information to be applied to the member that is named in the header statement. These statements and their formats are discussed in the following paragraphs.

The NUMBR statement is used with CHNGE operations to change the sequence number of one or more logical records within a member, and with ADD and REPL operations to assign sequence numbers to the records within new and replacing members. This statement affects only those sequence numbers that fall in the specified range. A NUMBR statement contains:

Columns	Contents
1-2	A period-slash (./).
10-14	NUMBR
16-23	The sequence number of the first record to be renumbered in CHNGE operations. This field is ignored in ADD and REPL operations.
24	A comma (,).
25-32	The sequence number of the last record to be renumbered in CHNGE operations. This field is ignored in ADD and REPL operations.
33	A comma (,).
34-41	The first new sequence number.
42	A comma (,).
43-50	The increment value of successive new sequence numbers.

Note: All of the sequence numbers must be 8-digit, alphameric fields.

The DELET statement is used to delete one or more logical records within a member. It is used only in conjunction with a CHNGE header statement. A DELET statement contains:

Columns	Contents
1-2	A period-slash (./).
10-14	DELET
16-23	The sequence number of the first logical record to be deleted.
24	A comma (,).
25-32	The sequence number of the last logical record to be deleted.

The logical record statements contain the data to be added to, or used to replace, existing logical records. They are used in conjunction with ADD, REPL, or CHNGE header statements. The logical record statements contain:

Columns	Contents
1-72	The data to be added or used as a replacement.
73-80	The sequence number of the record.

ALIAS Statements

The ALIAS statements create or retain aliases in the new master directory. They can be used in conjunction with any of the header statements. One ALIAS statement must be supplied for each alias. It contains:

Columns	Contents
1-2	A period-slash (./).
10-14	ALIAS
16	An alias name.

Note: If provided, system status index information for the member name is also inserted into the alias entry in the directory of the new master.

ENDUP Statement (Optional)

The ENDUP statement can be used to indicate the end of the SYSIN input to this program; it serves as an end-of-data indication if there is no other indication. The ENDUP statement follows the last group of SYSIN control statements and contains:

Columns	Contents
1-2	A period-slash (./).
10-14	ENDUP

IEBUPDAT Examples

IEBUPDAT Example 1 illustrates the cataloging of a set of job control statements in the cataloged procedure library. PROC6 WILL BE ADDED TO THE CATALOGED PROCEDURE LIBRARY DEFINED IN SYSUT2. The first record of this procedure will contain the sequence number 00000010 with successive records numbered 00000020, 00000030,.... The resulting cataloged procedure can be executed with an EXEC statement specifying PROC=PROC6. The publication IBM System/360 Operating System: Job Control Language Reference, GC28-6704, contains additional information on cataloged procedures.

IEBUPDAT Example 2 illustrates the replacement of a member in the old master by a new member. In this example, MBR7 (alias ALS7) will replace MBR7 of the library named in SYSUT1. Successive records in the new member will contain the sequence numbers 00000010, 00000020,....

Sample Coding Form

```
//UPD1      JOB  09#770,D.P.BROWN
//          EXEC PGM=IEBUPDAT,PARM=NEW
//SYSPRINT  DD   SYSOUT=A
//SYSUT2    DD   (Parameters defining the cataloged procedure library.)
//SYSIN     DD   DATA
./          ADD  PROC6,05,0,1
./        NUMBR  00000000,00000000,00000010,00000010
//STEP1     EXEC ---
//DD1       DD   ---
             .
             .
//STEP      EXEC ---
             .
             .
/*
```

IEBUPDAT Example 1. Cataloging Job Control Statements in the Cataloged Procedure Library

Sample Coding Form

```
//UPD2      JOB  09#770,D.P.BROWN
//          EXEC PGM=IEBUPDAT,PARM=MOD
//SYSPRINT  DD   SYSOUT=A
//SYSUT1    DD   (Parameters defining the old master data set.)
//SYSUT2    DD   (Parameters defining the new master data set.)
//SYSIN     DD   *
./          REPL MBR7,06,0,1
./        NUMBR  00000000,00000000,00000010,00000010
             Logical Record Statements
             .
             .
./        ALIAS  ALS7
/*
```

IEBUPDAT Example 2. Replacing a Member of a Symbolic Library

IEBUPDAT Example 3 illustrates the deletion of logical records 00000050 through 00000090 from a member of a symbolic library.

IEBUPDAT Example 4 illustrates the copying of a member from the old master onto the new master.

IEBUPDAT Example 5 illustrates the creation of a three-member library. The members are named LIBMEMB1, LIBMEMB2, and LIBMEMB3.

Sample Coding Form

```
//UPD3     JOB  09#770,D.P.BROWN
//         EXEC PGM=IEBUPDAT
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   (Parameters defining the old master data set.)
//SYSUT2   DD   (Parameters defining the new master data set.)
//SYSIN    DD   *
./       CHNGE  MBRS,13,0,1
./       DELET  00000050,00000090
./       ALIAS  ALS5
/*
```

IEBUPDAT Example 3. Deleting a Record From a Symbolic Library

Sample Coding Form

```
//UPD4     JOB  09#770,D.P.BROWN
//         EXEC PGM=IEBUPDAT,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   (Parameters defining the old master data set.)
//SYSUT2   DD   (Parameters defining the new master data set.)
//SYSIN    DD   *
./       REPRO  MBRZ,12,0,1
/*
```

IEBUPDAT Example 4. Copying a Member of a Symbolic Library

Sample Coding Form

```
//UPD5     JOB  09#770,D.P.BROWN
//         EXEC PGM=IEBUPDAT,PARM=NEW
//SYSPRINT DD   SYSOUT=A
//SYSUT2   DD   (Parameters for creating a master data set)
//SYSIN    DD   *
./        ADD   LIBMEMB1,01,0,1,1234ABCD
           Logical Record Statements
           .
           .
           .
./        ADD   LIBMEMB2,01,0,1,1234EFAB
           Logical Record Statements
           .
           .
           .
./        ADD   LIBMEMB3,01,0,1,1234ABAB
           Logical Record Statements
           .
           .
           .
/*
```

IEBUPDAT Example 5. Creating a Three Member Library

Program Applications

The IEBDG (data generator) program provides a "pattern" of test data to be used as a programming debugging aid. An output (test) 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.

Generating Test Data

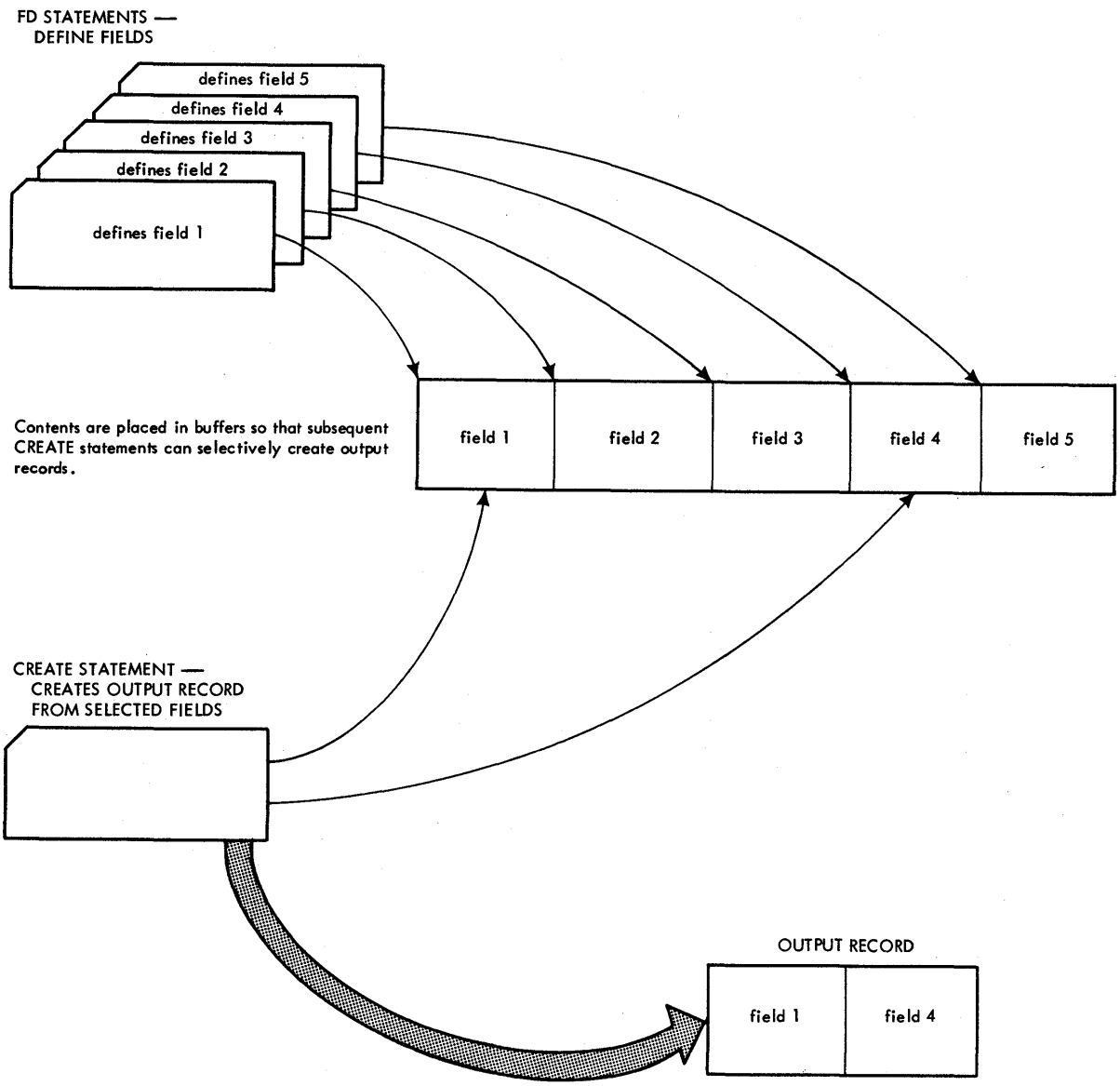
To generate test data, the user constructs a pattern of data that he can analyze quickly for predictable results. Test data is generated through the use of five utility control statements. (See IEBDG Table 1.)

IEBDG Table 1. Utility Control Statements

Statement	Use
The DSD statement	Refers to DD statements defining input to and output from the IEBDG program. One DSD statement is included per "set" of utility control statements.
The FD statement The CREATE statement The REPEAT statement	Work together to define the content and number of records in a data set.
The END statement	Signals the end of a set of utility control statements. Any number of sets can be included in a single job step; each set defines one data set.

USING THE DSD STATEMENT: The DSD statement marks the beginning of a set of utility control statements. This statement indicates to the IEBDG program the data sets that it will use for input and output. The DSD statement must indicate one and only one output data set for each application of the IEBDG program; any number of input data sets can be indicated on the same statement if they are needed.

USING THE FD STATEMENT: The FD statement defines the contents of a field. A defined field is used only if it is referred to, by name, by a subsequent CREATE statement. IEBDG Figure 1 shows how the contents of five fields are placed in buffer areas so that subsequent CREATE statements can assign selected fields to specific output records.



IEBDG Figure 1. Defining and Selecting Fields for Output Records

Field Selection

A specific field within the input logical record(s) may be selected for use in the output record(s). This field is defined by the NAME, LENGTH, STARTLOC, INPUT, and FROMLOC keywords of the FD statement.

INPUT
identifies the input data set containing the input records.

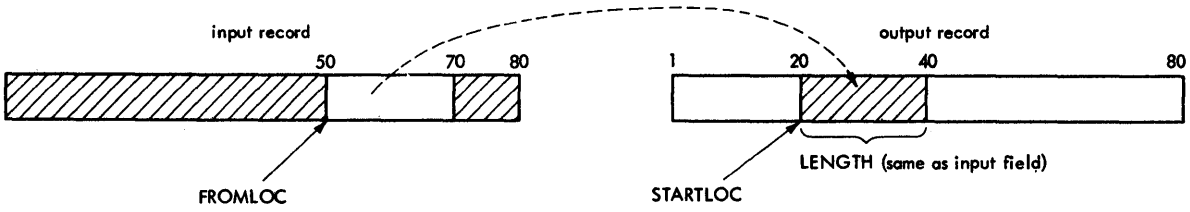
FROMLOC
specifies the field's offset (in bytes) from the beginning of the input record.

LENGTH
specifies the length (in bytes) of the selected field (applicable to both input and output).

STARTLOC
specifies where the field will be moved (offset in bytes from the beginning of the output record).

NAME
allows the selected field to be given a symbolic name.

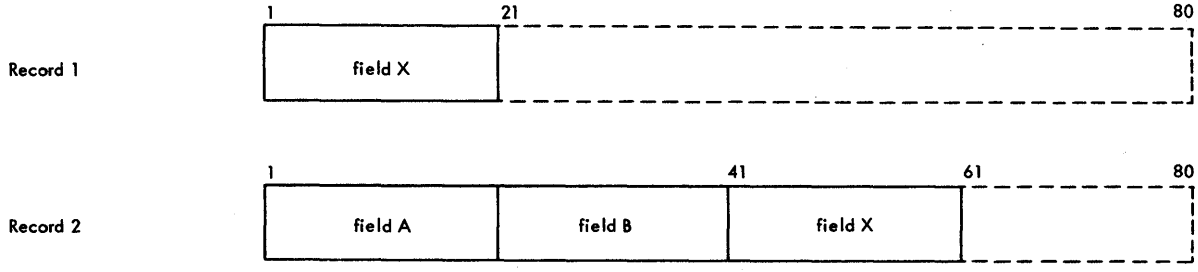
The field select operation is shown in IEBDG Figure 2.



IEBDG Figure 2. Field Selection Operation

If the user does not specify a starting location in an FD statement, the starting location is the first available byte of an output record.

IEBDG Figure 3 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.



IEBDG Figure 3. Default Placement of Fields Within an Output Record

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

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

IBM-Supplied Format: IBM supplies seven patterns. The user may choose one of them when he defines the contents of a field. These patterns are:

- Alphameric
- Alphabetic
- Zoned decimal
- Packed decimal
- Binary number
- Collating sequence
- Random number

IEBDG Table 2 shows the seven IBM-supplied formats. In this table, logical continuations within a pattern are shown by dotted lines (---). Repetition of a pattern is shown by a vertical bar (|).

All patterns except the binary and random number patterns will repeat themselves in a given field, provided that the defined field length is sufficient to permit repetition. For example, the alphabetic pattern is:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ ABCDEFG---
```

The user can specify a starting character when defining an alphameric, alphabetic, or collating sequence field. For example, a 10-byte alphabetic field for which "H" is specified as the starting character would appear as:

```
HIJKLMNOPQ.
```

The same 10-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 number field. If no sign is specified, the field is assumed to be positive.

IEBDG Table 2. IBM-Supplied Formats

Type	Expressed in Hexadecimal	Expressed in Printable Characters
Alphameric	C1 C2 C3---E9 F0---F9|C1---	ABC---Z0---9|ABC---
Alphabetic	C1 C2 C3---E9|C1 C2---	ABC---Z|ABC---
Zoned decimal	F0F0---F0F1	00---01
Packed Decimal	0000---001C (positive pattern requested) 0000---001D (negative pattern requested)	N/A
Binary number	00---01 (positive pattern requested) FF---FF (negative pattern requested)	N/A
Collating Sequence	40---F9	bc.<(+|&!$*);¬-/,%_>?:#@'=" A through Z 0 through 9
Random Number	random hexadecimal digits	N/A
Note: A packed decimal or binary number is right justified in the defined field.		

User-Supplied Picture: Instead of selecting an IBM-supplied format, the user can supply the IEBDG program with the picture he wishes to place in the defined field. The user can provide:

- An EBCDIC character string.

- A decimal number to be converted to packed decimal by the IEBDG program.

- A decimal number to be converted to a binary equivalent by the IEBDG program.

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

Action: The IEBDG program can 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
- Wave

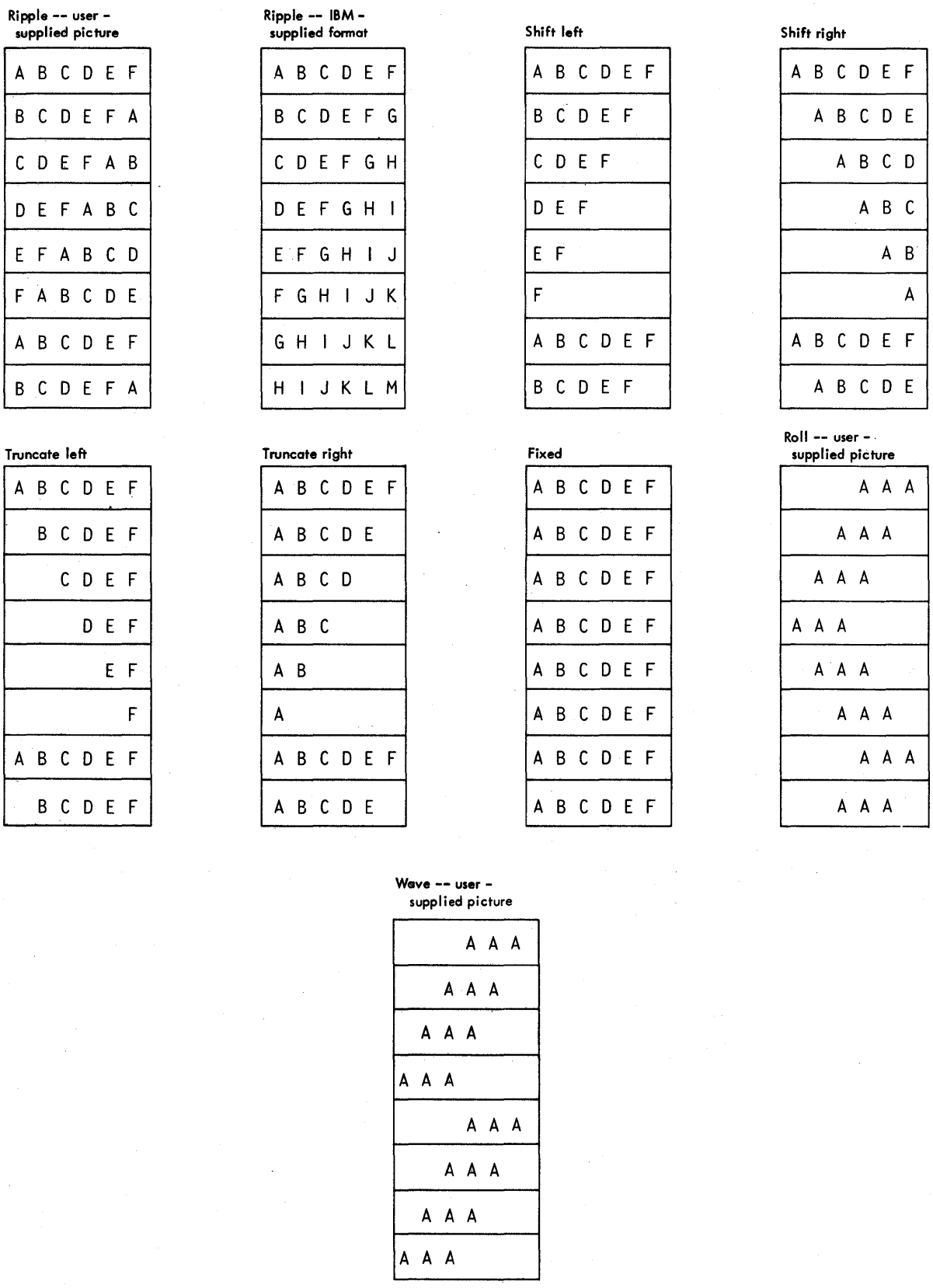
If no action is selected, or if the specified action is not compatible with the format, the "fixed" action is assumed by the IEBDG program.

IEBDG Figure 4 shows the effects of each of the actions on a 6-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 or user supplied -- is present.

The user can also indicate that a numerical 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.

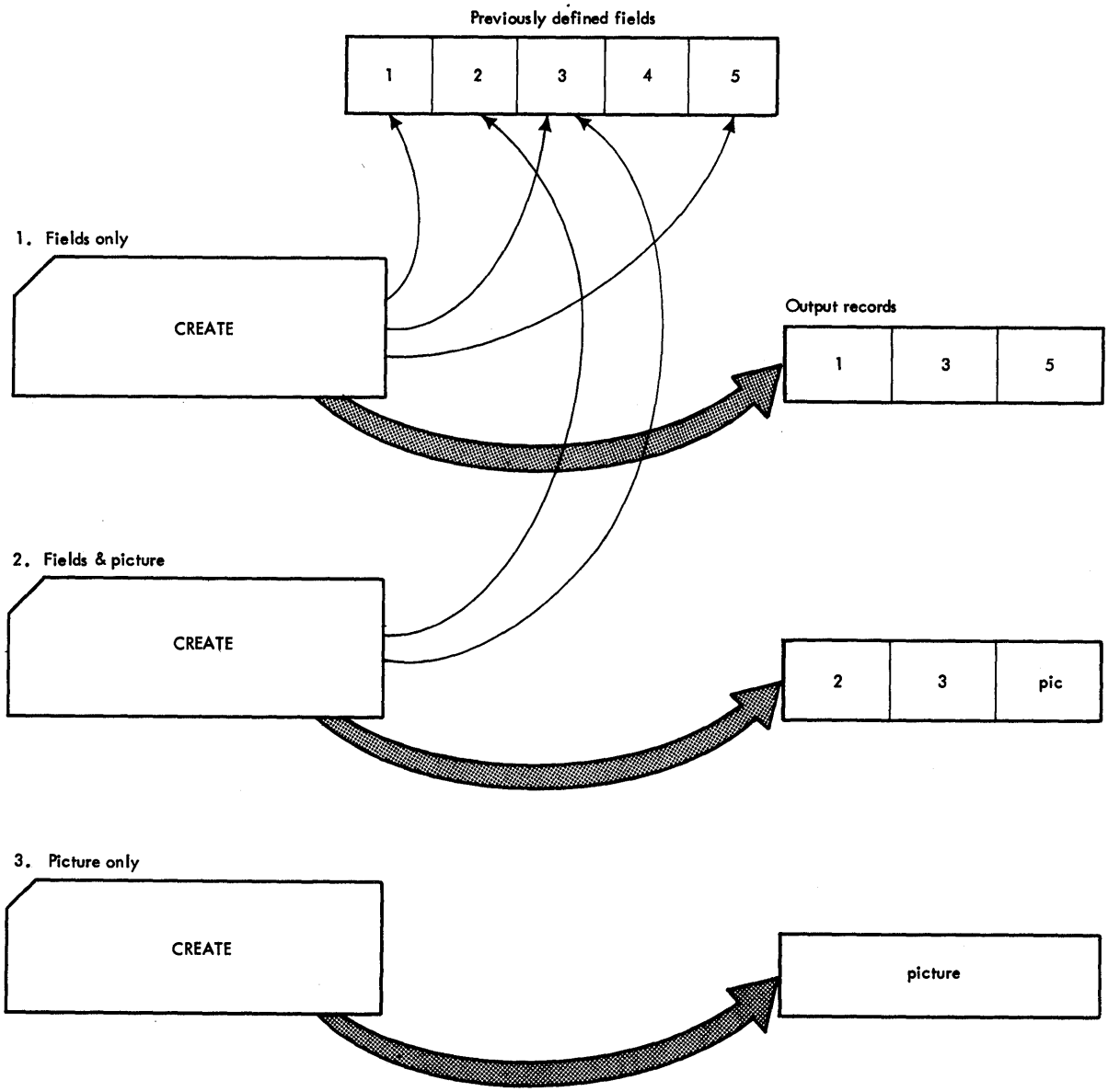
USING THE CREATE STATEMENT: The CREATE statement constructs an output record by referring, by name, to previously defined fields and/or by providing a picture to be placed in the record. The user can generate multiple records with a single CREATE statement.

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-justified.)



IEBDG Figure 4. IEBDG Actions

IEBDG Figure 5 shows three ways in which output records can be created from utility control statements.



IEBDG Figure 5. Creating Output Records With 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.

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. An input record, if any is provided, is placed left-justified in the output record.

3. FD fields, if any, are placed in the output record in the order of the appearance of their names in the CREATE statement.

4. A CREATE statement picture, if any, is placed in the output record.

The IEBDG program 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.

USING THE REPEAT STATEMENT: The REPEAT statement indicates how many times a CREATE statement or a group of CREATE statements is to be used repetitively by the IEBDG program. The REPEAT statement precedes the CREATE statement(s) to which it applies.

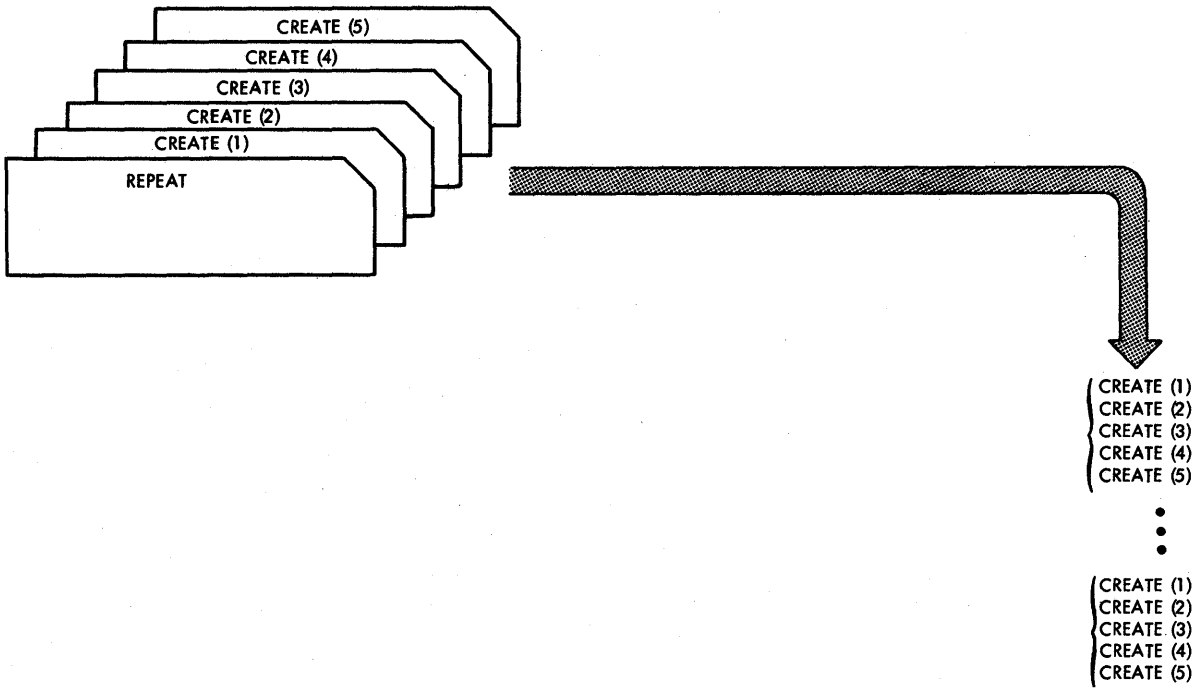
IEBDG Figure 6 shows a group of five CREATE statements repeated n times.

USING THE END STATEMENT: The END statement must be used to signal the end of a set of utility control statements. Each set of control statements can pertain to:

- Any number of input data sets.
- A single 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 keyword 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 this case, defined fields for field selection are not usable across sets of utility control statements. The FD card may be duplicated and used in more than one set of utility control statements within the job step.



IEBDG Figure 6. Order of Repetition Due to the REPEAT Statement

Inputs and Outputs

IEBDG Table 3 lists the major inputs to and outputs from the IEBDG program.

IEBDG Table 3. Data Sets Used (Input) and Produced (Output) by the IEBDG Program

Inputs	Input Data Set(s): The input data set (or sets) contain records that are to be used in the construction of an output data set or partitioned data set member. The input data set(s) are optional; that is, output records can be created entirely from utility control statements. Control Data Set: This data set contains utility control statements, which are used to determine the contents of the output records. The control data set can contain any number of sets of utility control statements.
Outputs	Message Data Set: This data set contains informational messages, the contents of applicable utility control statements, and error messages, if any. Output (Test) Data Set: An output data set is created as the result of an IEBDG operation. One output data set is created by each set of utility control statements included in the job step.
Note: Input and Output data sets may be sequential, indexed sequential, or partitioned data set members. (BDAM is not supported.)	

ADDITIONAL OUTPUTS

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

00 -- successful completion.
04 -- a user routine returned a code of 16 to the IEBDG program. (The job step is terminated at the user's request.)
08 -- an error occurred while processing a set of utility control statements. No data is generated from the point of the error. Processing continues normally with the next set of utility control statements, if any.
12 -- an error occurred while processing an input or output data set. The job step is terminated.
16 -- an error condition occurred from which no recovery was possible. The job step is terminated.

Control

The IEBDG program is controlled by job control statements and utility control statements. The job control statements are used to:

- Execute or invoke the program.
- Define the input data sets or partitioned data set members.
- Define the message data set.
- Define the output data sets or partitioned data set members.

Utility control statements are used to control the functions of the program and to define the contents of the output records.

JOB CONTROL STATEMENT INVOKING THE IEBDG PROGRAM

IEBDG Table 4. Job Control Statements for the IEBDG Program (Part 1 of 2)

Statement	Usage
JOB statement	This statement initiates the job.
EXEC statement	This statement specifies the program name (PGM=IEBDG) or, if the job control statements for the program reside in a procedure library, the procedure name. 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. PARM='LINECNT=nnnn' (where nnnn is a 4-digit decimal number) specifies that nnnn number of lines (0000 to 9999) are 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 the IEBDG program is invoked, the line count option can be passed in a parameter list that is referred to by the option addr subparameter of the LINK or ATTACH macro instruction. In addition, a page count can be passed in a 6-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."
SYSPRINT DD statement	This statement defines a sequential message data set. The data set can be written on a system output device, a magnetic tape volume, or a direct access volume. (This DD statement should be present for each execution or invocation of the IEBDG program.) If it is omitted, no messages will be written.
SYSIN DD statement	This statement 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 alternatively be defined as being a sequential data set or a member of a library of partitioned members.

//seqinset DD DSNAME=setname,UNIT=xxxx,DISP=(OLD,KEEP), C
VOLUME=SER=xxxxxx,DCB(applicable subparameters, C
including DSORG),LABEL=(...,...)
↑ for magnetic tape only
	This DD statement defines an optional sequential or indexed sequential data set used as input to the IEBDG program. The data set can reside on a magnetic 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 job step. Each DD statement is subsequently referred to by a DSD utility control statement.

(Part 1 of 2)

IEBDG Table 4. Job Control Statements for the IEBDG Program (Part 2 of 2)

Statement	Usage
//parinset DD DSNAME=setname(membername),UNIT=xxxx,DISP=(OLD, // KEEP),VOLUME=SER=xxxxxx,DCB=(applicable subparameters)	C
This DD statement 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 DD statement is referred to by a DSD utility control statement.	
//seqout DD DSNAME=setname, UNIT=xxxx, VOLUME=SER=xxxxxx, // DISP=(,KEEP),DCB=(applicable subparameters, including DSORG) // LABEL=(...,...),SPACE=(applicable subparameters) └magnetic tape only └direct access only	C
This statement defines an output (test) sequential or indexed sequential data set. It can define a data set on magnetic tape volume, a card punch, a printer, or a direct access volume. Any number of "seqout" DD statements can be included per job step; however, only one seqout (or parout) statement is applicable per set of utility control statements. Note: On an MVT system, the ddname of this statement should not be SYSPRINT.	
//parout DD DSNAME=setname (membername), UNIT=xxxx, // DISP=(,KEEP), VOLUME=SER=xxxxxx,DCB=(applicable // subparameters, including DSORG), SPACE=(applicable subparameters) └used only when creating the first member to be placed in a partitioned data set.	C C
This statement 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(or seqout) statement is applicable per set of utility control statements.	
Notes: The SYSPRINT (message) data set and the SYSIN (control) data set can have any blocking factor. Both input and output data sets can contain fixed-length, variable-length, or undefined-length records. Input data set record type must agree with the output data set record type. IMPORTANT: The DSORG subparameter must be included in the DCB subparameters of the output DD cards. Refer to the publication IBM System/360 Operating System: Supervisor and Data Management Services, GC28-6646, for information on estimating space allocations. Refer to the IEBDG examples for typical uses of the job control statements.	

UTILITY CONTROL STATEMENTS

The IEBDG program is controlled by five utility control statements:

- The DSD statement.
- The FD statement.
- The CREATE statement.
- The REPEAT statement.
- The END statement.

The DSD Statement

The DSD statement marks the beginning of a set utility control statements.

Name	Operation	Operand
[name]	DSD	OUTPUT=(ddname) [INPUT=(ddname,...)]

OUTPUT=(ddname)
specifies the ddname of the DD statement defining the output data set. The ddname must be enclosed in parentheses.

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 in the INPUT keyword.

The FD (Field Definition) Statement

The FD statement defines the contents and length of a field that will subsequently be used by a CREATE statement (or statements) in the formation of output record (or records).

Name	Operation	Operand
name	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]

NAME=name
specifies the name of the field defined by this FD statement.

LENGTH=length-in-bytes
specifies the length (in bytes) of the defined field.

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.
If the STARTLOC keyword is omitted, 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).

Note: For V-type records the starting location is the first byte after the length descriptor.

FILL='character'
specifies an EBCDIC character which is to be placed in each byte of the defined field prior to any other operation in the construction of a field.

FILL=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 defined field prior to any other operation in the construction of a field.

If neither FILL='character' nor FILL=X'2-hexadecimal-digits' is coded binary zeros are placed initially in the field.

FORMAT=pattern
specifies an IBM supplied pattern that is to be placed in the defined field. The patterns are:

```
FORMAT=AN -- alphameric.
FORMAT=ZD -- zoned decimal.
FORMAT=PD -- packed decimal.
FORMAT=CO -- collating sequence.
FORMAT=BI -- binary.
FORMAT=AL -- alphabetic.
FORMAT=RA -- random binary number.
```

If both the FORMAT and the PICTURE keywords are omitted, the specified FILL character appears in each byte of the defined field.

CHARACTER=character (used only with the FORMAT keyword)
specifies the starting character of a field. For example, if FORMAT=AL and CHARACTER=F are specified for a 10-byte field, the result is:

```
FGHIJKLMNO
```

If CHARACTER=character is omitted, the starting character is as shown in IEBDG Table 2.

PICTURE=length, { 'character-string' / P'decimal-number' / B'decimal-number' }
specifies the length and contents of a user-supplied field (FD) picture.

length: specifies the number of characters in the FD picture.

'character-string': specifies an EBCDIC character string that is to be placed in the defined field. The character string is left-justified in the field. A character string may be broken in column 71 and must be continued in column 4. (Note that double quotation marks must not be coded to represent a single quotation mark within a character string.)

P'decimal-number': specifies a decimal number that is to be converted to a packed decimal equivalent and placed right-justified in the defined field.

B'decimal-number': specifies a decimal number that is to be converted to a binary equivalent and placed right-justified in the defined field. In all cases the number of characters within the quotation marks must equal the number specified in the length subparameter.

If both the PICTURE and the FORMAT keywords are omitted, the FILL character appears in each byte of the defined field.

SIGN=sign
specifies a mathematical sign (SIGN=+ or SIGN=-). This keyword is used when defining a packed decimal or binary number field.

If the SIGN keyword is omitted, the sign is assumed to be positive.

ACTION=action
specifies that the contents of a defined field be altered after the field's inclusion in an output record. The actions are:

ACTION=SL -- shift left.
ACTION=SR -- shift right.
ACTION=TL -- truncate left.
ACTION=TR -- truncate right.
ACTION=RO -- roll.
ACTION=WV -- wave.
ACTION=FX -- fixed.
ACTION=RP -- ripple.

If the ACTION keyword is omitted, the fixed (FX) action is assumed.

INDEX=number
specifies a number to be added to this field whenever a specified number of records have been written. (The number of records is specified in the CYCLE keyword.)

If the INDEX keyword is omitted, no indexing is performed.

CYCLE=number (Used only with the INDEX keyword)
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 keyword. For example, if CYCLE=3, output records might appear as 111 222 333 444 etc.

If the CYCLE keyword is omitted and the INDEX keyword is coded, a CYCLE value of 1 is assumed; that is, the field is indexed after each inclusion in a potential output record.

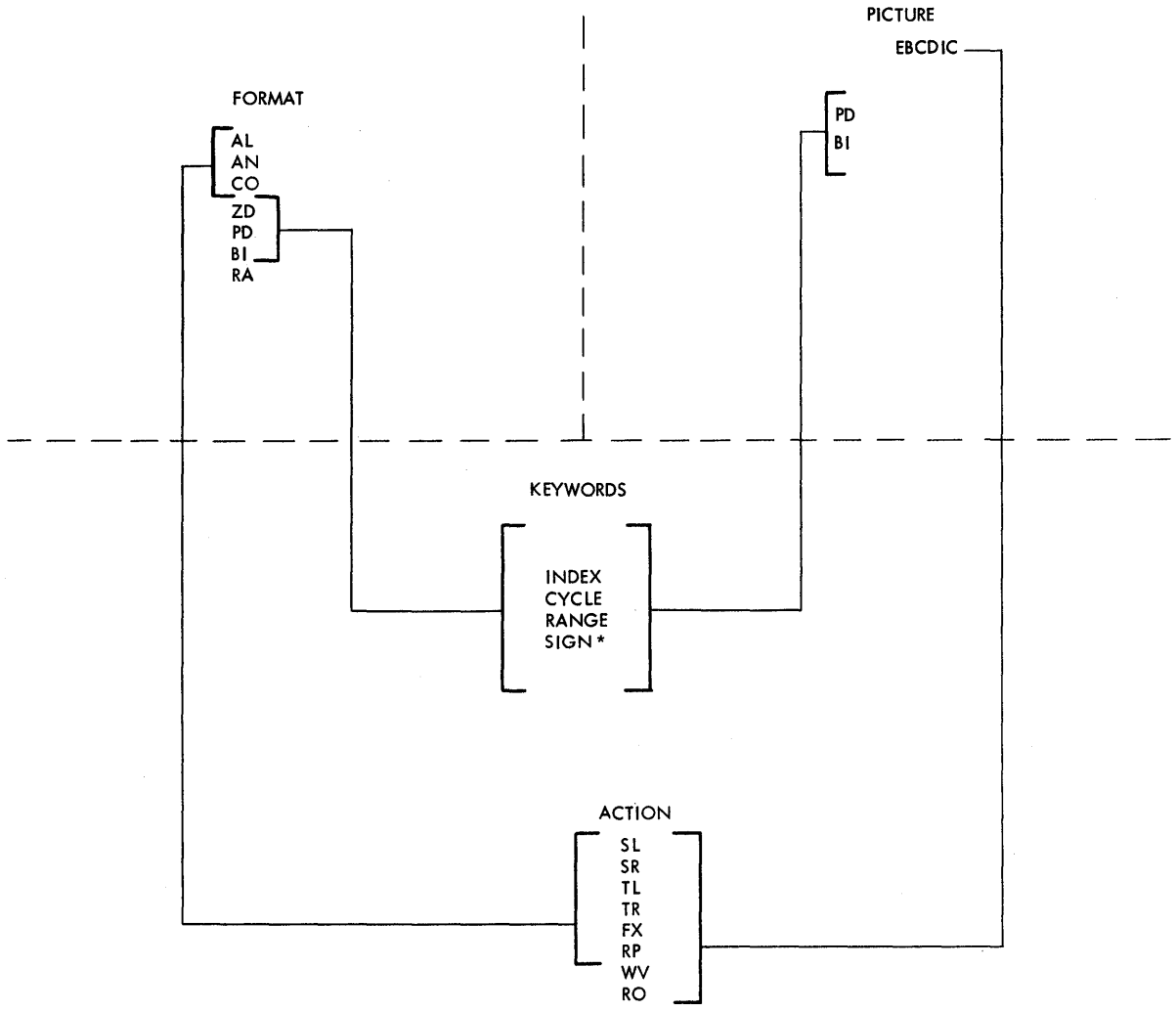
RANGE=number (Used with the INDEX keyword)
specifies an absolute value which the contents of this field can never exceed.

If an INDEX operation attempts to exceed the specified absolute value, the contents of the field as of the previous INDEX operation are used.

INPUT=ddname
specifies the ddname for the input data set.

FROMLOC=number
specifies the location of the selected field within the input logical record. The number represents the offset (in bytes) from the beginning of the input record. (For V-type records, the significant data begins on the first byte after the four-byte length descriptor.)

Note: Some of the FD keywords do not apply when certain formats or pictures are selected by the user; for example, the INDEX, CYCLE, RANGE, and SIGN keywords are used only with numeric fields. IEBDG Figure 7 shows which IEBDG keywords can be used with the applicable format or picture chosen by the user.



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

IEBDG Figure 7. Compatible Operations

The 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).

Name	Operation	Operand
[name]	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 | (name$_1$,...,name$_n$) | (name$_1$,(COPY=number,name$_2$,name$_3$,...))}]
		[EXIT=routinename]

QUANTITY=number
specifies the number of records that this CREATE statement is to generate.

If the QUANTITY keyword is omitted, one record is created unless the INPUT keyword is also coded, in which case the number of records created is equal to the number of remaining input records to be processed plus the generated data up to the specified number.

FILL='character'
specifies an EBCDIC character which is to be placed in each byte of the output record prior to any other operation in the construction of record.

FILL=X'2-hexadecimal-digits'
specifies two dexadecimal digits (for example, FILL=X'40', or FILL=X'FF') to be placed in each byte of the output record prior to any other operation in the construction of the record.

If neither FILL='character' nor FILL=X'2-hexadecimal-digits' is coded, binary zeros are placed initially in the output record.

INPUT={ddname | SYSIN [(cccc)]}
defines an input data set whose records are to be used in the construction of output records.

INPUT=ddname: specifies the ddname of a DD statement defining an input data set.

INPUT=SYSIN[(cccc)]: indicates that the SYSIN data set (input stream) contains records (other than utility control statements) to be used in the construction of output records. If INPUT=SYSIN or INPUT=SYSIN[(cccc)] 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 is coded, the input records are

delimited from any additional utility control statements by a record containing $$$E in columns 1 through 4.

If INPUT=SYSIN[(cccc)] is coded, the input records are delimited by a record containing from one to four EBCDIC characters beginning in column 1 (cccc represents any combination of from one to four EBCDIC characters).

If neither INPUT=ddname nor INPUT=SYSIN[(cccc)] is coded, the output record(s) is created entirely from utility control statements.

CAUTION: If the INPUT keyword is coded, the QUANTITY keyword should also be coded, unless the remainder of the input records are all to be processed by this CREATE statement.

PICTURE=length,startloc,{'character-string' / P'decimal-number' / B'decimal-number'}

specifies the length, starting byte, and contents of a user-supplied picture (CREATE statement picture).

length: specifies the number of bytes that the picture will occupy.

startloc: 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-justified; that is, it begins in the defined starting byte. A character string may be broken in column 71 and must be continued in column 4.

P'decimal-number': specifies a decimal number that is to be converted to a packed decimal equivalent and placed right-justified (within the boundaries of the defined length and starting byte) in the output record(s).

B'decimal-number': specifies a decimal number that is to be converted to a binary equivalent and placed right-justified (within the boundaries of the defined length and starting byte in the output record(s).

If both the PICTURE and the NAME keywords are omitted, the FILL character specified in the CREATE statement appears in each byte of the applicable output record(s).

NAME={name / ($name_1$,...,$name_n$) / ($name_1$,(COPY=number,$name_2$,$name_3$,...)}

specifies the name or names of previously defined fields to be included in the applicable output record(s).

($name_1$,...): 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 the NAME keyword; however, sets of

parentheses cannot be imbedded. Within each set of inner parentheses, COPY=number must appear before the name of any field.

If both the NAME keyword and the PICTURE keyword are omitted, the FILL character specified in the CREATE statement appears in each byte of the applicable output record(s).

EXIT=routinename
specifies the name of a user routine that is to receive control from the IEBDG program prior to the writing of each output record. After processing each potential output record, the user routine provides a return code to instruct the IEBDG program how to handle the output record. The user codes are:

00 -- write the record.

04 -- do not write this record. Do not count the skipped record as a generated output record; however, continue the program action as though a record was written.

Note: If skips are requested through user exits and input records are supplied, each skip will cause 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 will be processed.

12 -- bypass the processing of the remainder of this set of utility control statements. Continue processing with the next DSD statement.

16 -- halt all processing.

Note: Standard conventions are used when an exit routine is loaded and when the user returns control to the IEBDG program. Refer to Appendix A for details of register contents at these times.

The REPEAT Statement

The REPEAT statement specifies the number of times that a following CREATE statement or group of CREATE statements is to be used repetitively in the generation of output records.

Name	Operation	Operand
[name]	REPEAT	QUANTITY=number[,CREATE=number]

QUANTITY=number
specifies the number of times the defined group of CREATE statements is to be used repetitively. This number cannot exceed 65535.

CREATE=number
specifies the number of following CREATE statements to be included in the group.

If the CREATE keyword is omitted, only one CREATE statement is repeated.

The END Statement

The END statement signals the end of a set of utility control statements.

Name	Operation
[name]	END

IEBDG Examples

The following examples illustrate some of the uses of the IEBDG program.

IEBDG Example 1

Operation	Data Set Organization	Input Device	Output Device	Comments
Place binary zeros in selected fields	Input-SEQUENTIAL Output-SEQUENTIAL	TAPE- 9-track unlabeled	TAPE- 9-track, unlabeled	1. Blocked input and output.

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 20-29 and 50-59.

- The SEQIN DD Statement: defines an input sequential data set (INSET). The data set was originally written on a 9-track, unlabeled magnetic tape volume.
- The SEQOUT DD Statement: defines the test data set (OUTSET). The output records are identical to the input records, except for locations 20-29 and 50-59, which contain binary zeros at the completion of the operation.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream.
- The DSD Utility Control Statement: 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 Utility Control Statements: create two 10-byte fields (FIELD1 and FIELD2) that contain binary zeros. The fields are to begin in the 20th and 50th bytes of each output record.
- The CREATE Utility Control Statement: constructs 100 output rcords in which the contents of previously defined fields (FIELD1 and 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.
- The END Utility Control Statement: signals the end of a set of utility control statements.

```
//CLEAROUT JOB  ,,MSGLEVEL=1
//         EXEC PGM=IEBDG
//SYSPRINT DD   SYSOUT=A
//SEQIN    DD   DSNAME=INSET,UNIT=2400,DISP=(OLD,KEEP),LABEL=(,NL),
//              VOLUME=SER=240000,DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SEQOUT   DD   DSNAME=OUTSET,UNIT=2400,VOLUME=SER=240001,LABEL=(,NL),
//              DISP=(,KEEP),DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//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
/*
```

IEBDG Example 1. Placing Binary Zeros in Selected Fields of Copied Records

IEBDG Example 2

Operation	Data Set Organization	Input Device	Output Device	Comments
Ripple on Alphabetic Pattern	Input-SEQUENTIAL Output-SEQUENTIAL	TAPE- 9-track, standard labeled	DISK - 2311	1. Blocked input and output.

In this example, a 10-byte alphabetic pattern is to be rippled. At the end of the job step the output records contain

	first byte	11th byte	80th byte
Record 1	ABCDEFGHIJ	data in locations 11-80 from input record	
Record 2	BCDEFGHIJK	data in locations 11-80 from input record	
Record 3	CDEFGHIJKL	data in locations 11-80 from input record	
Record 4	DEFGHIJKLM	data in locations 11-80 from input record	

etc.

- The SEQIN DD Statement: defines an input sequential data set (INSET). The data set was originally written on a 9-track, standard labeled magnetic tape volume.

- The SEQOUT DD Statement: 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 2311 disk storage volume.

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

- The DSD Utility Control Statement: 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 Utility Control Statement: Creates a 10-byte field in which the pattern ABCDEFGHIJ is placed. The data is rippled after each output record is written.

- The CREATE Utility Control Statement: 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.

- The END Utility Control Statement: signals the end of a set of utility control statements.

```
//RIPPLE   JOB  ,,MSGLEVEL=1
//         EXEC PGM=IEBDG
//SYSPRINT DD   SYSOUT=A
//SEQIN    DD   DSNAME=INSET,UNIT=2400,DISP=(OLD,KEEP),
//              VOLUME=SER=240000,DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SEQOUT   DD   DSNAME=OUTSET,UNIT=2311,VOLUME=SER=231100,DISP=(,KEEP),
//              SPACE=(TRK,(20,10)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN    DD   *
       DSD      OUTPUT=(SEQOUT),INPUT=(SEQIN)
       FD       NAME=FIELD1,LENGTH=10,FORMAT=AL,ACTION=RP
       CREATE   QUANTITY=100,INPUT=SEQIN,NAME=FIELD1
       END
/*
```

IEBDG Example 2. Rippling an Alphabetic Pattern

IEBDG Example 3

Operation	Data set Organization	Output Device	Comments
Create output records from utility control statements only.	Input -- N/A Output -- SEQUENTIAL	DISK-2311	1. Blocked output.

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 indexed by one after each output record is written. At the end of the job step, the output records contain:

first byte	31st byte	61st byte	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
	etc.	packed dec.	

- The SEQOUT DD Statement: defines the test output data set. Twenty tracks of primary space and ten tracks of secondary space are allocated for the sequential data set on a 2311 disk storage volume.

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

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

- The FD Utility Control Statements: 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 10-byte field containing a packed decimal number (1234567890) to be indexed by one after each record is written.

- The CREATE Utility Control Statement: constructs 100 output records in which the contents of previously defined fields (FIELD1, FIELD2, and FIELD3) are included.

- The END Utility Control Statement: signals the end of a set of utility control statements.

```
//UTLYONLY JOB  ,,MSGLEVEL=1
//         EXEC PGM=IEBDG
//SYSPRINT DD   SYSOUT=A
//SEQOUT   DD   DSNAME=OUTSET,UNIT=2311,DISP=(,KEEP),
//              VOLUME=SER=240000,DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//              SPACE=(TRK,(20,10))
//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=20,STARTLOC=61,                    C
              PICTURE=10,P'1234567890',INDEX=1
       CREATE   QUANTITY=100,NAME=(FIELD1,FIELD2,FIELD3),FILL=X'FF'
       END
/*
```

IEBDG Example 3. Creating a Test Sequential Data Set From Utility Control Statements Only

IEBDG Example 4

Operation	Data Set Organization	Input Device	Output Device	Comments
Modifying input records from two partitioned members and supplying from the input stream	Input-PARTITIONED and SEQUENTIAL Output-PARTITIONED	2311 - DISK	2311 - DISK	1. Reblocking (from BLKSIZE=800 to BLKSIZE=960) is performed. 2. Each block of output records contain 10 modified partitioned input records and two input stream records

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. At the end of the job step, the output partitioned member contains:

```
DEPARTMENT 21     (Rightmost 67 bytes of INSET1(MEMBA)     record 1)
DEPARTMENT 21     (Rightmost 67 bytes of INSET1(MEMBA)     record 2)
   .                                                                  1st block of
   .                                                                  12 output
   .                                                                  records.
DEPARTMENT 21     (Rightmost 67 bytes of INSET1(MEMBA)     record 10)
   input record 1  from input stream
   input record 2  from input stream

DEPARTMENT 21     (Rightmost 67 bytes of INSET1(MEMBA)     record 11)
   .
   .
   .                                                                  2nd block of
DEPARTMENT 21     (Rightmost 67 bytes of INSET1(MEMBA)     record 20)  12 output
   input record 3  from input stream                                  records.
   input record 4  from input stream
   .
   .
   .
DEPARTMENT 21     (Rightmost 67 bytes of INSET1(MEMBA)     record 100)
   input record 19 from input stream
   input record 20 from input stream
DEPARTMENT 21     (Rightmost 67 bytes of INSET2(MEMBA)     record 1)
   .
   .
   .
   etc.
```

- The PARIN1 DD Statement: defines one of the input partitioned members.
- The PARIN2 DD Statement: defines the second of the input partitioned members. (Note that the members are from different partitioned data sets.)
- The PAROUT DD Statement: 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.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream.

- The DSD Utility Control Statement: 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 Utility Control Statement: creates a 13-byte field in which the picture DEPARTMENT 21 is placed.

- The First REPEAT Utility Control Statement: indicates that the following group of two CREATE statements is to be repeated 10 times.

- The First CREATE Utility Control Statement: creates 10 output records. Each output record is constructed from an input record (from partitioned data set INSET1) and previously defined FIELD1.

- The Second CREATE Utility Control 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 utility control statement.

  The next REPEAT statement group is identical to the preceding group, except that records from a different partitioned member are used as input.

- The END Utility Control Statement: signals the end of a set of utility control statement.

```
//MIX      JOB  ,,MSGLEVEL=1
//         EXEC PGM=IEBDG
//SYSPRINT DD   SYSOUT=A
//PARIN1   DD   DSNAME=INSET1(MEMBA),UNIT=2311,DISP=OLD,
//              VOLUME=SER=231100,DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//PARIN2   DD   DSNAME=INSET2(MEMBA),UNIT=2311,DISP=OLD,
//              VOLUME=SER=231101,DCB(RECFM=FB,LRECL=80,BLKSIZE=960)
//PAROUT   DD   DSNAME=PARSET(MEMBA),UNIT=2311,DISP=(,KEEP),
//              VOLUME=SER=231102,SPACE=(TRK,(20,10,5)),
//              DCB(RECFM=FB,LRECL=80,BLKSIZE=960)
//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 record 1
                 .
                 .
                 .
           input record 20
$$$E            (delimiter)
       REPEAT   QUANTITY=10,CREATE=2
       CREATE   QUANTITY=10,INPUT=PARIN2,NAME=FIELD1
       CREATE   QUANTITY=2,INPUT=SYSIN
           input record 21
                 .
                 .
           input record 40
$$$E            (delimiter)
       END
/*
```

IEBDG Example 4. Reblocking/Modifying Records From Two Partitioned Data Set Members and Supplying Additional Records From the Input Stream

IEBDG Example 5

Operation	Data Set Organization	Output Device	Comments
Create three partitioned data set members entirely from utility control statements	Input - N/A Output-PARTITIONED	2311-DISK	1. Block output. 2. Three sets of utility control statements (one set for each member) are used.

In this example, output records are to be created entirely 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. At the end of the job step, the partitioned data set members contain:

MEMBA

FIELD1 1st byte	FIELD3 31st byte	FIELD2 51st byte	71st byte
ABCDEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQRST	0000000001000000001	fill
BCDEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQRS	0000000001000000001	fill
CDEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQR	0000000001000000001	fill
DEFGHIJKLMNOPQRSTUVWXYZABCD	ABCDEFGHIJKLMNOPQ	0000000001000000001	fill
etc.			BINARY ZEROS

MEMBB

3 1st byte	3 21st byte	3 41st byte	2 61st byte
ABCDEFGHIJKLMNOPQRST	ABCDEFGHIJKLMNOPQRST	ABCDEFGHIJKLMNOPQRST	0000000001000000001
ABCDEFGHIJKLMNOPQRS	ABCDEFGHIJKLMNOPQRS	ABCDEFGHIJKLMNOPQRS	0000000001000000001
ABCDEFGHIJKLMNOPQR	ABCDEFGHIJKLMNOPQR	ABCDEFGHIJKLMNOPQR	0000000001000000001
ABCDEFGHIJKLMNOPQ	ABCDEFGHIJKLMNOPQ	ABCDEFGHIJKLMNOPQ	0000000001000000001
etc.			

MEMBC

4 1st byte	1 31st byte	61st byte
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123	ABCDEFGHIJKLMNOPQRSTUVWXYZABCD	fill
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123	BCDEFGHIJKLMNOPQRSTUVWXYZABCD	fill
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123	CDEFGHIJKLMNOPQRSTUVWXYZABCD	fill
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123	DEFGHIJKLMNOPQRSTUVWXYZABCD	fill
etc.		BINARY ZEROS

- The PAROUT1 DD Statement: 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.

- The PAROUT2 and PAROUT3 DD Statements: defines the second and third members, respectively, of the output partitioned data set. Note that each DD statement specifies DISP=OLD and UNIT=AFF=PAROUT1.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream.
- Each DSD Utility Control Statement: 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.
- Each FD Utility Control Statement: defines the contents of a field that is used in the subsequent construction of output records.
- Each CREATE Utility Control Statement: constructs 4 records from combinations of previously defined fields.
- Each END Utility Control Statement: signals the end of a set of utility control statements.

```
//UTSTS    JOB  ,,MSGLEVEL=1
//         EXEC PGM=IEBDG
//SYSPRINT DD   SYSOUT=A
//PAROUT1  DD   DSNAME=PARSET(MEMBA),UNIT=2311,DISP=(,KEEP),
//              VOLUME=SER=231100,SPACE=(TRK,(20,10,5)),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//PAROUT2  DD   DSNAME=PARSET(MEMBB),UNIT=AFF=PAROUT1,DISP=OLD,
//              VOLUME=SER=231100,DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//PAROUT3  DD   DSNAME=PARSET(MEMBC),UNIT=AFF=PAROUT1,DISP=OLD,
//              VOLUME=SER=231100,DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//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   QUANTIFY=4,NAME=(FIELD4,FIELD1)
       END
/*
```

IEBDG Example 5. Creating Three Partitioned Data Set Members Entirely From Utility Control Statements

IEBDG Example 6

Operation	Data Set Organization	Output Device	Comments
Rolling and waving user supplied patterns	Input - N/A Output- SEQUENTIAL	2311-DISK	1. Output records are created entirely from utility Control statements.

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. At the end of the job step, the output records contain:

FIELD1	FIELD2	FIELD3	FIELD4	FIELD5	FIELD6	FIELD7	FIELD8	FIELD9	FIELD10
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
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

* <u>The OUTSET DD Statement</u>: defines the output sequential data set on a 2311 disk volume. Twenty tracks of primary space and ten tracks of secondary space are allocated to the data set.

* <u>The SYSIN DD Statement</u>: defines the control data set, which follows in the input stream.

* <u>The DSD Utility Control Statement</u>: marks the beginning of a set of utility control statements and refers to the DD statement defining the output data set.

* <u>Each FD Utility Control Statement</u>: defines a field that will 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.

* <u>The CREATE Utility Control Statement</u>: constructs 300 records from the contents of the previously defined fields.

* <u>The END Utility Control Statement</u>: signals the end of a set of utility control statements.

```
//ROLLWAVE JOB  ,,MSGLEVEL=1
//         EXEC PGM=IEBDG
//SYSPRINT DD   SYSOUT=A
//OUTSET   DD   DSNAME=SEQSET,UNIT=2311,DISP=(,KEEP),
//              VOLUME=SER=2311,SPACE=(TRK,(20,10)),
//              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,FIELD4,FIELD5, C
                FIELD6,FIELD7,FIELD8,FIELD9,FIELD10)
       END
/*
```

IEBDG Example 6. Rolling and Waving User-Supplied Pictures

IEBDG Example 7

Operation	Data Set Organization	Output Device	Comments
Create an indexed sequential data set using field selection for input and output records, plus data generation	Input - SEQUENTIAL Output - INDEXED SEQUENTIAL	2311 - DISK	1. Indexed output records are created by augmenting selected input fields with generated data.

In this example, the first 10 bytes of the output record contains zoned decimal format generated data. 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.

- The TAPEIN DD Statement: defines the input sequential data set.
- The DISKOUT DD Statement: defines the output indexed sequential data set.
- The SYSIN DD Statement: defines the control data set, which follows in the input stream.
- The DSD Utility Control Statement: marks the beginning of a set of utility control statements and refers to the DD statement defining the output data set.
- Each FD Utility Control Statement: 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 eleven of the output logical record. The second FD statement defines and locates the 10-byte key field.
- The CREATE Utility Control Statement: constructs a 90-byte output record by referring to the previously defined fields.
- The END Utility Control Statement: signals the end of a set of utility control statements.

```
//CREATEIS JOB  MSGLEVEL=1
//BEGIN    EXEC PGM=IEBDG
//TAPEIN   DD   DSNAME=TAPEIT,DCB=(BLKSIZE=80,LRECL=80,RECFM=F),
//              DISP=(OLD,KEEP),UNIT=2400,LABEL=(,SL),VOL=SER=MASTER
//DISKOUT  DD   DSNAME=CREATIS,DCB=(BLKSIZE=270,LRECL=90,RECFM=FB,
//              DSORG=IS,NTM=2,OPTCD=MY,RKP=0,KEYLEN=10,CYLOFL=1),
//              DISP=(NEW,KEEP),UNIT=2311,SPACE=(TRK,2,,CONTIG),
//              VOL=SER=231100
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
       DSD      OUTPUT=(DISKOUT),INPUT=(TAPEIN)
       FD       NAME=DATAFD,LENGTH=80,FROMLOC=1,STARTLOC=11,INPUT=TAPEIN
       FD       NAME=KEYFD,LENGTH=10,STARTLOC=1,FORMAT=ZD,INDEX=1
       CREATE   INPUT=TAPEIN,NAME=(KEYFD,DATAFD)
       END
/*
```

IEBDG Example 7. Creating an Indexed Sequential

Section 3: Independent Utilities

Independent utility programs operate outside, and in support, of the IBM System/360 Operating System. They include:

- IBCDASDI (DASDI) -- a program that initializes and assigns alternate tracks to a direct access volume.
- IBCDMPRS (DUMP/RESTORE) -- a program that dumps and restores the data contents of a direct access volume.
- IBCRCVRP (RECOVER/REPLACE) -- a program that recovers usable data from a defective track, assigns an alternate track, and merges replacement data with the recovered data onto the alternate track.

Utility Control Statement Requirements

Independent utility programs require a JOB statement, a MSG statement, a group of statements that describe the function to be performed, and an END statement.

The JOB statement indicates the beginning of a job.

Name	Operation	Operand
[name]	JOB	[user information]

The MSG statement defines an output device for operator messages. It follows the JOB statement and precedes a group of function-defining statements that are associated with one of the independent utility programs.

Name	Operation	Operand
[name]	MSG	TODEV=xxxx TOADDR=cuu [MODE=mm]

TODEV=xxxx
specifies the device type of the message output device.

TOADDR=cuu
specifies the channel number (c) and unit number (uu) of the message output device.

MODE=mm (used only with RECOVER/REPLACE and when the message output device is a 7-track tape)
specifies the mode in which the message output tape is to be written. Valid modes are shown in Independent Utilities Table 1.

If MODE is omitted, 6B is assumed.

Independent Utilities Table 1. Valid 7-Track Tape Unit Modes

Mode (mm)	Density (bits per inch)	Translator	Data Converter	Parity
13	200	Off	On	Odd
23	200	Off	Off	Even
33	200	Off	Off	Odd
2B	200	On	Off	Even
3B	200	On	Off	Odd
53	556	Off	On	Odd
63	556	Off	Off	Even
73	556	Off	Off	Odd
6B	556	On	Off	Even
7B	556	On	Off	Odd
93	800	Off	On	Odd
A3	800	Off	Off	Even
B3	800	Off	Off	Odd
AB	800	On	Off	Even
BB	800	On	Off	Odd

The END Statement denotes the end of job. It appears after the last function-defining statement.

Name	Operation	Operand
[name]	END	[user information]

Methods of Operation

Independent utility programs are 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 DASDI, DUMP/RESTORE, and RECOVER/REPLACE:

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 depressing 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 two manners:

   a. Depress the REQUEST key of the console typewriter. The message DEFINE INPUT DEVICE is printed.

      Enter the following message from the console typewriter:

      INPUT=xxxx cuu

   where xxxx is the device type, c is the channel address, and uu is the unit address. The device type can be 1402, 1442, 2400 or 2540.In addition, the RECOVER/REPLACE program can use the 1052 as a device type.

b. (Use only if the console typewriter is not available.) Enter one of the following numbers at storage location 0110 (hexadecimal):

   1cuu for a 1442 Card Read Punch.
   2cuu for a 2400 Nine-Track Magnetic Tape Drive.
   0cuu for a 2540 Card Read Punch.

   Depress 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.

   If the job executes the RECOVER/REPLACE program and the message output device is a tape, the console lights display the hexadecimal value DDDD at a normal end of the job, and EEEE at an abnormal end of job. If a machine check occurs, 00E2 is displayed.

IBCDASDI–Initializing and Assigning Alternate Tracks on Direct Access Volumes

The IBCDASDI (DASDI) independent utility program performs two separate functions: it initializes direct access volumes for use with the operating system, and assigns alternate tracks on nondrum, direct access storage volumes. A single job can initialize one volume or assign alternates for specified tracks on one volume. DASDI jobs can be performed continuously by stacking complete sets of control statements. DASDI is not supported on MP65 with the mode switch set to MS; the mode switch must be set to 65.

Initializing a Direct Access Volume

The first function of the DASDI program is to initialize a direct access volume. A volume can be initialized with or without a surface analysis (i.e., a test for defective tracks); however, a surface analysis should be included when a volume is initialized for the first time.

Note: A 2321 volume is automatically initialized with a surface analysis.

Initialization With Surface Analysis: The DASDI program:

- Checks for tracks that have been previously designated as defective (flagged) and have had alternates assigned. The program automatically assigns alternates (disk devices only). This test must be suppressed when a disk recording surface is being initialized for the first time.
- Performs a surface analysis of each track and automatically assigns alternates, if necessary (non-drum storage volumes only). Tracks that are available for disposition as alternates are checked first.
- Writes a standard home address, a track descriptor record (record 0), and erases the remainder of each track.
- Writes IPL records on track 0 (records 1 and 2).
- Writes volume label on track 0 (record 3) and provides space for additional records, if requested.
- Constructs and writes a volume table of contents (VTOC).
- Writes IPL program, if requested, on track 0 (2301 or 2314) or track 1 (2303 or 2311).

Initialization Without Surface Analysis: The DASDI program:

- Checks for tracks that have been previously designated as defective (flagged) and have had alternates assigned. The program automatically assigns alternates (disk devices only). This test must not be suppressed.
- Writes a standard home address, a track descriptor record (record 0), and erases the remainder of each track.
- Writes IPL records on track 0 (records 1 and 2).

- Writes volume label on track 0 (record 3) and provides space for additional records, if requested.
- Constructs and writes a volume table of contents (VTOC).
- Writes IPL program, if requested, on track 0 (2301 or 2314) or track 1 (2303 or 2311).

The DASDI program requires the following function-defining statements to initialize direct access volumes:

1. DADEF Statement
2. VLD Statement
3. VTOCD Statement
4. IPLTXT Statement (optional)
5. LASTCARD Statement (optional)

These statements must appear in this sequence.

Note: A DASDI job that initializes a 2321 data cell cannot follow a DASDI job that initializes a different device type unless the DASDI program is reloaded.

DADEF Statement

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

Name	Operation	Operand	
[name]	DADEF	TODEV=xxxx	1
		TOADDR=cuu	
		[IPL=YES]	
		{VOLID=serial }	
		{VOLID=SCRATCH}	
		[BIN=d]	
		[FLAGTEST=NO]	2
		[PASSES=n]	
		[BYPASS=YES]	3

Notes:
[1]Applicable to initialization with or without surface analysis.
[2]Applicable to initialization with surface analysis.
[3]Applicable to initialization without surface analysis.

TODEV=xxxx
specifies the device type of the direct access device.

TOADDR=cuu
specifies channel number (c) and unit number (uu) of the device.

IPL=YES
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.

If IPL is omitted, no IPL program is written.

VOLID=serial
specifies the volume serial number of the volume to be initialized.

If "serial" matches the volume serial number found on the volume to be initialized, the operation proceeds. If it does not match, the operator is notified.

VOLID=SCRATCH
specifies that no volume serial number check is to be made.

FLAGTEST=NO (applicable with surface analysis)
specifies that the program is not to check for previously flagged tracks before surface analysis is attempted on this device. (FLAGTEST=NO applies only to disk storage devices, and should be specified when the disk recording surface is initialized for the first time.)

Note: Since no check is ever made for previously flagged tracks on drum volumes or on 2321 volumes, FLAGTEST=NO is not coded when these devices are initialized.

PASSES=n (applicable with surface analysis)
specifies that the program's defective-track checking feature is to make n number of passes (from 1 to 255) per track.

If PASSES is omitted, one pass is made per track. The PASSES option is not applicable to 2321 volumes.

BYPASS=YES
specifies that the program's defective-track checking feature is to be bypassed.

If BYPASS is omitted, tracks will be checked and those found defective will automatically be assigned alternates. The BYPASS option is not applicable to the 2321 Data Cell Drive.

BIN=d
specifies a decimal bin number (0-9). This parameter is applicable only to the 2321 Data Cell Drive.

VLD Statement

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

Name	Operation	Operand
[name]	VLD	NEWVOLID=serial
		{VOLPASS=1 / VOLPASS=0}
		[OWNERID=xxxxxxxxxx]
		[ADDLABEL=n]

NEWVOLID=serial
specifies a 1- to 6-character volume serial number.

VOLPASS=1
specifies that the volume security bit is to be set to 1.

VOLPASS=0
specifies that the volume security bit is to be set to 0.

If VOLPASS is omitted, the volume security bit will be set to 0.

OWNERID=xxxxxxxxxx
specifies a 1- to 10-character field that identifies the owner of the volume.

If OWNERID is omitted, no identification is given.

ADDLABEL=n
specifies a number between one and seven that indicates the total number of additional labels for which space is to be allocated.

If ADDLABEL is omitted, 0 is assumed.

VTOCD Statement

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

Name	Operation	Operand
[name]	VTOCD	STRTADR=nnnnn EXTENT=nnnn

STRTADR=nnnnn
specifies the 1- to 5-byte track address, relative to the beginning of the volume, at which the volume table of contents is to begin. The VTOC cannot occupy track 00 or any alternate track.

EXTENT=nnnn
specifies the length of the volume table of contents in tracks. The number of entries per track for each type of device is given below.

Device	VTOC Entries/Track
2301	63
2314	25
2302	22
2303	17
2311	16
2321	8

IPLTXT Statement

The IPLTXT statement separates utility control statements from IPL program text statements. It is required only when IPL text is included. The statement consists of the operation IPLTXT, followed with blanks.

When IPL text is included, END must start in column 2.

LASTCARD Statement

The LASTCARD statement is required only when a DASDI job or a series of stacked DASDI jobs is followed by other statements on the control statement input device. It must follow the last END statement applying to a DASDI job. It consists of the operation LASTCARD, followed with blanks.

Assigning an Alternate Track

The second function of the DASDI program is used to (1) test a track and, if necessary, to assign an alternate, or (2) to bypass testing, and automatically assign an alternate.

Assigning an Alternate (With Testing): An alternate track will be assigned for a track specified for testing and found defective. If the defective track has had an alternate previously assigned, a new alternate is assigned. 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 found to be "not defective," no alternate is assigned. The operator is notified by a message.

Assigning an Alternate (Without Testing): The program's defective track checking feature can be bypassed, and 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.

GETALT Statement

Any number of alternate tracks on a volume can be assigned in a single job by including one GETALT statement for each track.

Name	Operation	Operand
[name]	GETALT	TODEV=xxxx
		TOADDR=cuu
		TRACK=ccccchhhh
		VOLID=serial
		[FLAGTEST=NO]
		[PASSES=n]
		[BYPASS=YES]
		[BIN=d]

TODEV=xxxx
specifies the device type of the direct access device.

TOADDR=cuu
specifies the channel number (c) and unit number (uu) of the direct access device.

TRACK=ccccchhhh
specifies the address of the track for which an alternate is requested, where cccc is the cylinder number and hhhh is the head number.

VOLID=serial
specifies the volume serial number of the volume to which an alternate track is to be assigned.
If "serial" matches the volume serial number found on this volume, the alternate track assignment proceeds. If it does not match, the operator is notified.

FLAGTEST=NO (used when testing prior to assigning an alternate)
specifies that the program will not check for a previously flagged track before a surface analysis is attempted on this track (disk storage devices only).

PASSES=n (used when testing prior to assigning an alternate)
specifies that the program's defective track checking feature is to make n number of passes (from 1 to 255) when performing a surface analysis on this track.
If PASSES is omitted, one pass will be made on this track.

BYPASS=YES
specifies that the program's defective track checking feature is to be bypassed.

If BYPASS is omitted, the program assigns an alternate only if it finds that the specified track is defective.

BIN=d
specifies a decimal bin number (0-9). This parameter is applicable only to the 2321 Data Cell Drive.

Notes: A list of defective tracks is provided with new IBM Disk Storage volumes. This list should be referred to the first time the DASDI program is to be used. After initialization, the GETALT function can then be included in a DASDI job to assign an alternate track for each track on the list. Subsequent DASDI jobs will "remember" those defective tracks, unless the FLAGTEST=NO option is specified for those jobs.

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.

IBCDASDI Examples

IBCDASDI Example 1 illustrates the first initialization of a disk storage volume. A surface analysis is performed with the initialization.

```
                                Sample Coding Form
INIT      JOB  'INITIALIZE 2311'
            MSG TODEV=1403,TOADDR=00E
          DADEF TODEV=2311,TOADDR=190,VOLID=SCRATCH,FLAGTEST=NO
            VLD NEWVOLID=111111
          VTOCD STRTADR=50,EXTENT=10
            END
```

IBCDASDI Example 1. Initializing a Disk Storage Volume With Surface Analysis

IBCDASDI Example 2 illustrates an initialization (other than the first) of a drum storage volume. No surface analysis is performed with the initialization.

```
                                Sample Coding Form
INIT      JOB  'INITIALIZE 2301'
            MSG TODEV=1403,TOADDR=00E
          DADEF TODEV=2301,TOADDR=1C0,VOLID=SCRATCH,BYPASS=YES
            VLD NEWVOLID=230100
          VTOCD STRTADR=1,EXTENT=3
            END
```

IBCDASDI Example 2. Initializing a Drum Storage Volume Without Surface Analysis

IBCDASDI Example 3 illustrates the initialization of an IBM 2311 disk storage volume for later use as a system residence volume. An IPL program is included as a load module in standard TXT format.

```
                                Sample Coding Form
INIT      JOB  'INITIALIZE 2311'                 COMMENTS
            MSG TODEV=1403,TOADDR=00E            MESSAGE OUTPUT
          DADEF TODEV=2311,TOADDR=108,IPL=YES    VOLUME DEFINITION
            VLD NEWVOLID=P1,OWNERID=BROWN,       VOLUME LABEL          C
                ADDLABEL=2                        DEFINITION
          VTOCD STRTADR=2,EXTENT=9               VTOC DEFINITION
        IPLTXT                                   DELIMITER
 TXT                                             IPL PROGRAM
 .                                               .
 .              IPL Text                         .
 .                                               .
 TXT
 END            (IPL Text END card)
```

IBCDASDI Example 3. Initializing a Direct Access Volume

IBCDASDI Example 4 illustrates the assignment of three alternate tracks to a disk storage volume, without re-initialization of the volume. The program's defective-track checking feature is bypassed when the first two of the three tracks are assigned.

Sample Coding Form

```
ALTRK     JOB  'ASSIGN ALTERNATE TRACKS ON 2311' COMMENTS
              MSG TODEV=2400,TOADDR=180             MESSAGE OUTPUT
STMT1   GETALT TODEV=2311,TOADDR=190,TRACK=006F0001,                  C
               BYPASS=YES,VOLID=P2
STMT2   GETALT TODEV=2311,TOADDR=190,TRACK=00910004,                  C
               BYPASS=YES,VOLID=P2
STMT3   GETALT TODEV=2311,TOADDR=190,TRACK=004B0007,                  C
               VOLID=P2
              END
```

IBCDASDI Example 4. Assigning Alternate Tracks on a Disk Storage Volume

IBCDMPRS—Dumping and Restoring a Direct Access Volume

The IBCDMPRS (DUMP/RESTORE) program dumps and restores the data on direct access volumes. The data contents of a direct access volume (all data except the home address) can be "dumped" onto IBM 2311 or 2314 disk storage volumes or onto magnetic tapes, and restored onto a direct access volume that resides on the same type of device as the source volume. Both the source volume and the volume onto which data is to be restored must have been initialized to IBM System/360 Operating System specifications. This utility is useful for preparing transportable copies and backup copies of direct access volume contents. DUMP/RESTORE is not supported on MP65 with the mode switch set to MS; the mode switch must be set to 65.

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 is dumped, including any data on alternate tracks. If both the source and receiving volumes reside on 2311 Disk Storage Drives or on 2314 Disk Storage Drives, the data on the receiving volume is an exact replica of the source data and need not be restored.

Note: When a standard label restore tape created by IBCDMPRS is restored by IEHDASDR, the DD card describing the tape for IEHDASDR must specify LABEL=(,BLP). Bypass-label-processing must have been sysgened by specifying OPTIONS=BYLABEL on the SCHEDULR control card. If bypass-label-processing is not available, any standard label tape created by IBCDMPRS cannot be restored by IEHDASDR, but IBCDMPRS.

Name	Operation	Operand
[name]	DUMP	FROMDEV=xxxx
		FROMADDR=cuu
		TODEV=xxxx
		TOADDR=cuu
		[VOLID=serial=list]
		[MODE=mm]
		[BIN=d]

FROMDEV=xxxx
specifies the device type of the source device.

FROMADDR=cuu
specifies channel number (c) and unit number (uu) of the source device.

TODEV=xxxx
specifies the device type of the receiving device.
If the receiving device is a magnetic tape drive 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.)

TOADDR=cuu
specifies the channel number (c) and unit number (uu) of the receiving device.

VOLID=serial[,serial]...
specifies the volume serials of the receiving volumes onto which data is to be dumped. (VOLID is required when the receiving volume has been initialized to operating system specifications.)

If "serial" matches the volume serial number found on the receiving volume, the dump operation proceeds. If it does not match, the operator is notified.

If VOLID is not specified and the receiving volume contains a volume serial number, the operator is notified.

MODE=mm
specifies the bit density for data written onto the receiving magnetic tape volume. This parameter is applicable to 7-track tape drives and to 9-track tape drives with density selections of 800 and 1600 bits-per-inch. Valid 7-track modes are shown in Independent Utilities Table 1. (Only those modes which set converter on are accepted.)

For 9-track tape drives with density selections of 800 and 1600 bits-per-inch, the mode settings are:

MODE=CB for 800 bpi.
MODE=C3 for 1600 bpi.

If the receiving device is not a magnetic tape drive, the MODE parameter is ignored. If the receiving device is a tape drive but no mode is specified, the data is written at the highest density supported by the device.

BIN=d
specifies a decimal bin number (0-9). This parameter is applicable only to the 2321 Data Cell Drive.

Note: When dumping from direct access to magnetic tape, "dump time" can be minimized by specifying different channel selections in the TOADDR=cuu and FROMADDR=cuu keywords. For example,

```
DUMP FROMDEV=2311,FROMADDR=190,TODEV=2400,TOADDR=282
```

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.

Name	Operation	Operand
[name]	VDRL	BEGIN=nnnnn [END=nnnnn]

BEGIN=nnnnn
specifies a 1- to 5-byte relative track address that identifies the first track to be dumped.

END=nnnnn
specifies the relative 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.

If END is omitted, the last track of the volume, excluding those tracks reserved as alternates, is assumed to be the upper limit.

RESTORE Statement

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

Name	Operation	Operand
[name]	RESTORE	FROMDEV=xxxx
		FROMADDR=cuu
		TODEV=xxxx
		TOADDR=cuu
		VOLID=serial
		[MODE=mm]
		[BIN=d]

FROMDEV=xxxx
specifies the device type of the source device.

FROMADDR=cuu
specifies the channel number (c) and unit number (uu) of the source device.

TODEV=xxxx
specifies the device type of the receiving device. This device type must be the same as the device containing the volume originally dumped.

TOADDR=cuu
specifies the channel number (c) and unit number (uu) of the receiving device.

VOLID=serial
specifies the volume serial number of the receiving volume.

If "serial" matches the volume serial number found on the receiving volume, the restore operation proceeds. If it does not match, the operator is notified.

MODE=mm
specifies the bit density for data written onto the receiving magnetic tape volume. This parameter must match the mode specified when data was written onto the source volume. MODE should not be specified if the source or receiving volume is not a magnetic tape, or if MODE was not specified when data was written onto the source volume.

Valid 7-track modes are shown in Independent Utilities Table 1. (Only those modes which set converter on are accepted.) For 9-track tape drives with density selections of 800 and 1600 bits-per-inch, the mode settings are:

MODE=CB for 800 bpi.
MODE=C3 for 1600 bpi.

BIN=d
specifies a decimal bin number (0-9). This parameter is applicable only to the 2321 Data Cell Drive.

Note: When restoring from magnetic tape, "restore time" can be minimized by specifying different channel selections in the TOADDR=cuu and FROMADDR=cuu keywords. For example:

```
RESTORE FROMDEV=2400,FROMADDR=282,TODEV=cuu,TOADDR190
```

IBCDMPRS Examples

IBCDMPRS Example 1 illustrates the dumping of a direct access volume onto a tape volume.

IBCDMPRS Example 2 illustrates the restoring of dumped data onto a direct access volume.

```
                                Sample Coding Form
DUMP       JOB   DUMP 2311 ONTO TAPE              COMMENTS
           MSG   TODEV=1052,TOADDR=103            MESSAGE OUTPUT
          DUMP   FROMDEV=2311,FROMADDR=203,                              C
                 TODEV=2400,TOADDR=120
           END
```

IBCDMPRS Example 1. Dumping a Direct Access Volume

```
                                Sample Coding Form
RESTORE    JOB   RESTORE 2311 FROM TAPE
           MSG   TODEV=1052,TOADDR=10B
       RESTORE   FROMDEV=2400,FROMADDR=120,TODEV=2311,                    C
                 TOADDR=202,VOLID=PZ
           END
```

IBCDMPRS Example 2. Restoring a Direct Access Volume

IBCRCVRP–Recovering Data From a Defective Track

The IBCRCVRP (RECOVER/REPLACE) independent utility program retrieves usable data from a defective track, assigns an alternate track, and merges the usable data with replacement data onto the alternate track. (Alternate tracks must be assigned manually on drum devices.) A single job can retrieve or replace data on one or more volumes that contain defective tracks. RECOVERY/REPLACE is not supported on MP65 with the mode switch set to MS; the mode switch must be set to 65.

Note: The IEHATLAS utility program will perform these operations under control of the Operating System.

Recovering Usable Data

The RECOVER routine retrieves data from a defective track, writes this data on a receiving tape, and lists the bad records on the message output device. Each defective track requires a RECOVER statement; an optional LIST statement requests that both usable records and bad records be listed. The statements must appear in the sequence:

```
RECOVER
[LIST]
RECOVER
[LIST]
  .
  .
  .
```

RECOVER Statement

Each RECOVER statement in a job identifies (1) the direct access volume that contains the defective track, (2) the defective track, and (3) a unique receiving tape.

Name	Operation	Operand
[name]	RECOVER	FROMDEV=xxxx
		FROMADDR=cuu
		TODEV=xxxx
		TOADDR=cuu
		VOLID=serial
		TRACK=bbbbccccchhhh
		[MODE=mm]

FROMDEV=xxxx
: specifies the device type of the direct access device that contains the defective track, for example, 2311.

FROMADDR=cuu
: specifies the channel number (c) and unit number (uu) of the direct access device that contains the defective track.

TODEV=xxxx
: specifies the device type of the receiving tape volume.

If the receiving device is a magnetic tape drive 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.)

TOADDR=cuu
: specifies the channel number (c) and unit number (uu) of the receiving tape volume. This unit must be different from other receiving tapes in the same job. If this volume has no label or a nonstandard label, the RECOVER routine writes a tape mark preceding the data.

VOLID=serial
: specifies the volume serial number of the direct access volume that contains the defective track.

  If "serial" matches the volume serial number found on the specified volume, the program proceeds. If it does not match, the operator is notified and the job is terminated.

TRACK=bbbbccccchhhh
: specifies the hexadecimal bin-cylinder-head address of the defective track. If the specified track is one for which an alternate has been assigned, data is recovered from the alternate, and a message identifying both tracks is issued.

MODE=mm
: specifies the bit density for data written onto the receiving magnetic tape volume. This parameter is applicable to 7-track tape drives and to 9-track tape drives with density selections of 800 and 1600 bits-per-inch. Valid 7-track modes are shown in Independent Utilities Table 1. (Only those modes which set converter on are accepted.)

  For 9-track tape drives with density selections of 800 and 1600 bits-per-inch, the mode settings are:

  MODE=CB for 800 bpi.
  MODE=C3 for 1600 bpi.

  If no mode is specified, the data is written at the highest density supported by the device.

LIST Statement

The LIST statement requests that the entire contents (both usable records and bad records) of the defective track be listed, and identifies the output device on which records are listed (list device). If it is omitted, only bad records are listed on the message output device.

Name	Operation	Operand
[name]	LIST	TODEV=xxxx TOADDR=cuu [MODE=mm]

TODEV=xxxx
: specifies the device type of the list device.

TOADDR=cuu
specifies the channel number (c) and unit number (uu) of the list device.

MODE=mm (used only when the list device is a 7-track magnetic tape)
specifies the mode in which the list tape is to be written. Valid modes are shown in Independent Utilities Table 1.

If MODE is not specified, and the list device is different from the message output device, MODE=93 is assumed.

Notes: If the list device and the message output device are the same, the list mode will be the same as that of the message output device.

Neither the list device nor the message output device can be the same as the receiving tape identified in the RECOVER statement.

Tape volumes must have either a standard label or a tape mark in place of a label. The label or tape mark must be written in the same mode as the data.

Replacing Bad Data

The REPLACE routine merges data recovered from a defective track with replacement data, and writes the result on an assigned alternate track. (Alternate tracks must be assigned manually on direct access devices that are not disks.)

The REPLACE routine uses:

1. REPLACE statement
2. LIST statement (optional)
3. INSERT statement (one or more)

These statements must appear in the sequence shown above. When the same device is used to read both control statements and replacement records, a replacement record must follow the INSERT statement that describes it.

REPLACE Statement

The REPLACE statement identifies both the tape device containing recovered data (recover tape), and the direct access volume on which recovered data is merged with new replacement data.

Name	Operation	Operand
[name]	REPLACE	FROMDEV=xxxx
		FROMADDR=cuu
		TODEV=xxxx
		TOADDR=cuu
		VOLID=serial
		TRACK=bbbbccccchhhh
		[MODE=mm]

FROMDEV=xxxx
specifies the device type on which the recover tape is mounted.

If MODE is not specified in this statement, the REPLACE routine assumes that the recover tape was written in maximum density (default mode).

FROMADDR=cuu
specifies the channel number (c) and the unit number (uu) of the tape device on which the recover tape is mounted.

TODEV=xxxx
specifies the device type of the direct access device on which recovered data is to be merged with replacement data.

TOADDR=cuu
specifies the channel number (c) and unit number (uu) of the direct access device on which recovered data is to be merged with replacement data.

VOLID=serial
specifies the volume serial number of the direct access volume that contains the defective track.

TRACK=bbbbccccchhhh
specifies the hexadecimal bin-cylinder-head address of the defective track from which data was recovered.

MODE=mm
specifies the bit density at which data was written onto the source magnetic tape volume. MODE should not be specified if it was not specified when data was written onto the source volume.

Valid 7-track modes are shown in Independent Utilities Table 1. (Only those modes which set converter on are accepted.) For 9-track tape drives with density selections of 800 and 1600 bits-per-inch, the mode settings are:

MODE=CB for 800 bpi.
MODE=C3 for 1600 bpi.

LIST Statement

The LIST statement requests that both recovered data and replacement records be listed after they are merged. If it is omitted, only replacement records are listed on the message output device.

Name	Operation	Operand
[name]	LIST	TODEV=xxxx
		TOADDR=cuu
		[MODE=mm]

TODEV=xxxx
specifies the device type of the list device.

TOADDR=cuu
specifies the channel number (c) and unit number (uu) of the list device.

MODE=mm (used only when the list device is 7-track tape)
specifies the mode in which the list tape is to be written. Valid modes are shown in Independent Utilities Table 1.

If MODE is not specified, and the list device is different from the message output device, MODE=93 is assumed.

Notes: If the list device and the message output device are the same, the list mode will be the same as the message mode.

Neither the list device nor the message output device can be the same as the tape device containing recovered data.

Tape volumes must have either a standard label or a tape mark in place of a label. The label or tape mark must be written in the same mode as the data.

INSERT Statement

The INSERT statement identifies the device that contains each replacement record and describes the count field of that record. INSERT statements and corresponding data must be in sequence by record number (e.g., if records 3 and 5 are bad, the INSERT statement and replacement data for record 3 must precede the INSERT statement and data for record 5).

Name	Operation	Operand
[name]	INSERT	[FROMDEV=xxxx]
		[FROMADDR=cuu]
		{RECORD=yyy }
		{RECORD=LAST}
		COUNT=cccchhhhrrkkdddd
		[MODE=xx]
		[OVERFLOW=yes]

FROMDEV=xxxx
specifies the device type of the device that contains replacement data.

FROMDEV need not be written if the bad record did not contain key or data fields.

FROMADDR=cuu
specifies the channel number (c) and unit number (uu) of the device that contains replacement data.

FROMADDR need not be written if the bad record did not contain key or data fields.

RECORD=yyy
indicates the decimal record number of the original bad record. (This number is obtained from message IBC305I issued by the RECOVER routine.)

RECORD=LAST
indicates that this replacement record is to be the last physical record written on the alternate track. Records can be added after this record if the track capacity is not exceeded. With this feature, records near the end of a defective track that has missing address markers (and, thus, could not be recovered) can still be replaced.

COUNT=cccchhhhrrkkdddd
describes (in hexadecimal) the count field for the replacement record, where cccc is the cylinder number, hhhh is the head number, rr is the physical record number, kk is the key length (in bytes), and dddd is the data length (in bytes).

MODE=xx (used only when the replacement data is on a 7-track tape)
specifies the mode in which the input tape was written. Valid modes are shown in Independent Utilities Table 1. This tape volume must have either a standard label or a tape mark in place of a label. The label or tape mark must be written in the same mode as the data.

If MODE is omitted, 93 is assumed.

OVERFLOW=YES
indicates that the bad record, which is being replaced was a segment (other than the last segment) of an overflow record. The replacement record will be either the last record or the only record on the assigned alternate track.

Replacement Data Records

A replacement record is an 80-byte card image; it must be supplied if the key or data fields were found by the RECOVER routine to be bad. It contains replacement data for only the bad fields.

Columns	Contents
1-8	I/D=xxx (column 8 ignored) or I/D=LAST The value of I/D must be the same as that of the RECORD parameter of the associated INSERT statement.
9-10	blank
11-80	Replacement data in hexadecimal. Extends for as many bytes as specified in the COUNT parameter of the INSERT statement. Each card image can replace 35 bytes (columns 11-80) of data.

Additional replacement records are processed in sequential order. They must have the same format as the first record, except that columns 1-8 are ignored.

IBCRCVRP Examples

IBCRCVRP Example 1, illustrates a sequence that recovers data from defective tracks on IBM 2311 disk storage volumes 123456 and 222222. The entire contents of these tracks are listed on an IBM 1403 printer -- channel 0, unit 0E.

```
                                Sample Coding Form
 JOB            'RECOVER 2311 TRACKS'
 MSG            TODEV=1403,TOADDR=00E
 RECOVER        FROMDEV=2311,FROMADDR=190,TODEV=2400,TOADDR=280,        C
                VOLID=123456,TRACK=0000005E0008
 LIST           TODEV=1403,TOADDR=00E
 RECOVER        FROMDEV=2311,FROMADDR=191,TODEV=2400,TOADDR=281,        C
                VOLID=222222,TRACK=000000110005
 LIST           TODEV=1403,TOADDR=00E
 END
```

IBCRCVRP Example 1. Recovering Data From Defective Tracks

IBCRCVRP Example 2 illustrates a sequence that replaces bad data on IBM 2311 disk storage volume 123456. Replacement records 001 and 003 contain 32 (20 in hexadecimal) bytes and 40 bytes (including an 8-byte key) of replacement data, respectively. Record 003 must be continued onto an additional card image since the replacement data exceeds 35 bytes.

```
                                Sample Coding Form
 JOB            'REPLACE 2311 TRACK'
 MSG            TODEV=1403,TOADDR=00E
 REPLACE        FROMDEV=2400,FROMADDR=280,TODEV=2311,TOADDR=190,        C
                VOLID=123456,TRACK=0000005E0008
 LIST           TODEV=1403,TOADDR=00E
 INSERT         FROMDEV=1402,FROMADDR=00C,RECORD=001,                   C
                COUNT=005E000801000020
I/D=001     ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789
 INSERT         FROMDEV=1402,FROMADDR=00C,RECORD=003,                   C
                COUNT=005E000803080020
I/D=003     FFFFFFFFFFFFFFFFDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
            DDDDDDDDDD
 END
```

IBCRCVRP Example 2. Replacing Data on an Alternate Track

Appendix A: Exit Routine Linkage

Linking to an Exit Routine

Data set utility programs perform linkage operations by using the LINK macro instruction (except linkage to user label processing and/or totaling routines, and IEBCTRIN exits named OUTREC and ERROR, which is performed by the BALR instruction). This macro instruction contains the symbolic name of the entry point to an exit routine and, if required, a list of parameters. Linkage Table 1 shows the ordered parameter list for each available exit. (Format and contents of the DCB are presented in the publication IBM System/360 Operating System: Supervisor and Data Management Macro Instructions, GC28-6647.)

At the time of the linkage operation:

- General register 1 contains the starting address of the parameter list.
- For the OUTREC and ERROR exits, register 1 contains a zero to indicate end-of-file on the input data set (SYSUT1). No return code is necessary.
- 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 F: Utility Program Handling of 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 to these values before control is returned to the utility program.

The exit routine must be contained in either the job library or the link library.

Linkage Table 1. Parameter Lists for Exit Routines (Part 1 of 2)

Program	Exit	Parameters (In Order of Appearance in Parameter List)
COPYPDS	None	
GENERATE	INHDR	Address of input header label; address of DCB; address of status information; address of totaling area.
	OUTHDR	Address at which user output label is to be created; address of DCB; address of status information; address of totaling area.
	INTLR	Address of input trailer label; address of DCB; address of status information; address of totaling area.
	OUTLR	Address at which user output trailer label is to be created; address of DCB; address of status information; address of totaling area.
	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 (packed word).[1]
	TOTAL	Address of output buffer; address of DCB; address of status information; address of totaling area.
COMPARE	INHDR	Address of input header label; address of DCB; address of status information.
	INTLR	Address of input trailer label; address of DCB; address of status information.
	ERROR	Address of DCB for SYSUT1; address of DCB for SYSUT2.[2]
	PRECOMP	Address of SYSUT1 record; length of SYSUT1 record, address of SYSUT2 record; length of SYSUT2 record.
PRINT/PUNCH	INHDR	Address of input header label; address of DCB; address of status information.
	INTLR	Address of input trailer label; address of DCB; address of status information.
	INREC	Address of input record; length of input record.
	OUTREC	Address of output record; length of output record.
UPDATE	INHDR	Address of input header label for SYSIN data set; address of DCB; address of status information.
	INTLR	Address of input trailer label for SYSIN data set; address of DCB; address of status information.
	OUTHDR	Address at which user output label is to be created; address of DCB; address of status information; address of totaling area.
	OUTLR	Address at which user output trailer label is to be created; address of DCB; address of status information; address of totaling area.

(Part 1 of 2)

Linkage Table 1. Parameter Lists for Exit Routines (Part 2 of 2)

Program	Exit	Parameters (In Order of Appearance in Parameter List)
IEBTCRIN	ERROR OUTREC	Address of the record; address of a full word which contains the record length.
	OUTHDR2 OUTHDR3	Address of 80-byte buffer area in which output header label is to be created by the user; address of DCB; address of status information; reserved word.
	OUTTLR2 OUTTLR3	Address of 80-byte buffer area in which output header label is to be created by the user; address of DCB; address of status information; reserved word.

[1]For the packed-word format of this parameter, refer to the CHECK macro instruction, discussed in the publication IBM System/360 Operating System: Supervisor and Data Management Macro Instructions.

[2]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 the publication IBM System/360 Operating System: Supervisor and Data Management Macro Instructions.

Returning From an Exit Routine

An exit routine returns control to the utility program by means of the RETURN macro instruction in the exit routine.

Name	Operation	Operand
[name]	RETURN	[(r_1,r_2)] {RC=n RC=(15)}

(r_1,r_2)
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=n
specifies a return code to be placed in the 12 low-order bits of general register 15.

RC=(15)
specifies that general register 15 already contains a valid return code.

If RC is omitted, register 15 is loaded as specified by (r_1,r_2).

The utility programs examine the return code and respond as described in Linkage Table 2. Further information on the use of the LINK and RETURN macro instructions is contained in the publication IBM System/360 Operating System: Supervisor and Data Management Services, GC28-6646, and IBM System/360 Operating System: Supervisor and Data Management Macro Instructions, GC28-6647.

Linkage Table 2. Action on Return Codes

Type of Exit	Return Code	Action
Label processing exits	0,4,8	Return code is passed to the OPEN routine.
	16	Utility program is terminated.
	Any other number	Return code is passed to the OPEN 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	Terminate.
All other exits (except IEBTCRIN exits ERROR and OUTREC)	0-11 (Set to next lowest multiple of four; i.e., 0, 4, 8)	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.
	12-16 (Set to either 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	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.

Return Codes from IEBTCRIN

At job termination, the IEBTCRIN program produces a return code to indicate the results of program execution. The return codes and their interpretations are shown in Table 3.

Linkage Table 3. Return Codes from Job Termination

Code (decimal)	Interpretation
00	Normal termination.
04	Warning message issued; execution permitted. Conditions leading to issuance of this code are: 1. SYSPRINT, SYSIN, SYSUT2, or SYSUT3 DD statements missing. 2. DCB parameters missing in SYSUT2 or SYSUT3 DD statements.
12	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. 3. Invalid or conflicting utiltiy control statements.
16	Terminal error message issued; execution terminated. Conditions leading to issuance of this code are: 1. Permanent I/O 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. 5. User exit routine not found.

Appendix B: Invoking Utility Programs

Utility programs can be invoked by a problem program through the use of the ATTACH or LINK macro instructions. In addition, IEBTCRIN can be invoked through the use of LOAD or CALL.

The problem program must supply 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.

```
┌──────┬─────────┬──────────────────────────────────────────────────┐
│Name  │Operation│Operand                                           │
├──────┼─────────┼──────────────────────────────────────────────────┤
│[name]│ LINK    │EP=progname                                       │
│      │ ATTACH  │PARM=(optionaddr,                                 │
│      │         │[ddnameaddr],                                     │
│      │         │[hdingaddr])                                      │
│      │         │,VL=1                                             │
└──────┴─────────┴──────────────────────────────────────────────────┘
```

EP=progname
specifies the symbolic name of the utility program.

PARM
specifies, as a sublist, address parameters to be passed from the problem program to the utility program. The first fullword in the address parameter list contains the address of information that is usually specified in the PARM parameter of the EXEC statement. The second fullword contains the address of the ddname list. The third fullword contains the address of the page count list.

If standard ddnames are to be used and this is not the last parameter in the list it should point to a halfword of zeroes. If it is the last parameter it may be omitted.

optionaddr
specifies the address of a variable length list containing EBCDIC information usually specified in the PARM parameter of the EXEC statement. This address must be written for all utility programs.

The option list must begin on a halfword boundary (one 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, 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.

ddnameaddr
specifies the address of a variable length EBCDIC list containing 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 zeroes. If it is the last parameter it may be omitted.

The ddname list must begin on a halfword boundary (one 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 less than eight bytes must be left justified 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 8-byte entry must contain binary zeros. Names can be omitted from the end by merely shortening the list.

The sequence of the 8-byte entries in the ddname list is as follows:

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

hdingaddr
specifies the address of a 6-byte list containing an EBCDIC page count for the output device. The first two bytes of this list contain the length in bytes of the hding list. The remaining four bytes contain a page number that the utility program is to place on the first page of printed output.

VL=1
specifies that the sign bit of the last fullword of the address parameter list is to be set to 1.

LINK/ATTACH Figure 1 shows how a utility program can be invoked by a calling program. In this figure, the IEHINITT system utility program is used to place volume label sets onto a magnetic tape.

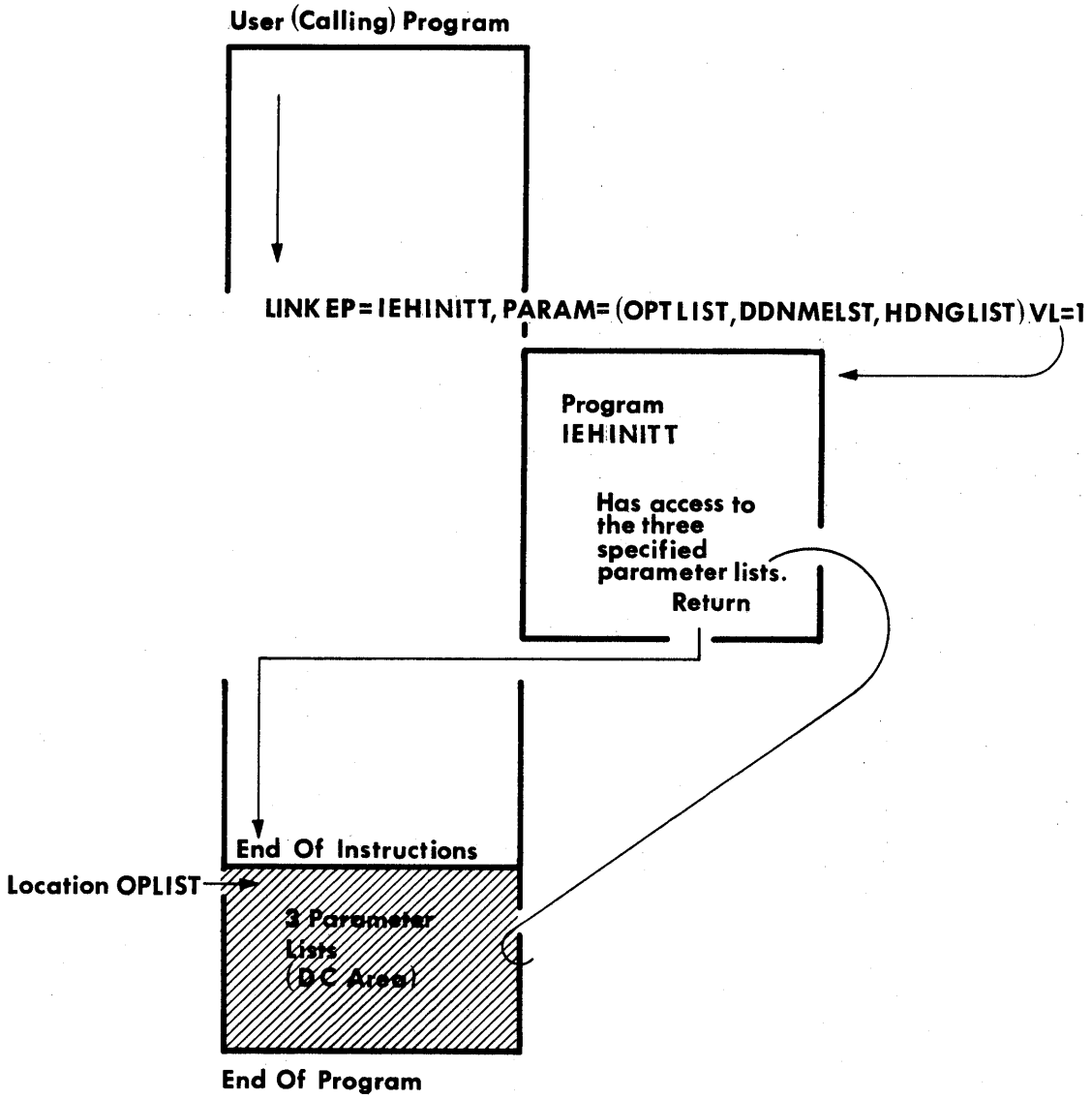
The PARAM parameter of the LINK macro instruction in the calling program provides the utility program with the symbolic addresses of three parameter lists: OPTLIST (the optionaddr list), DDNMELST (the ddnameaddr list), and HDNGLIST (the hdingaddr list). These lists, as they exist in the user DC area, are shown in LINK/ATTACH Figure 2.

- The optionaddr list: includes the number of bytes in the list (08 hexadecimal) and the NOVERIFY option.

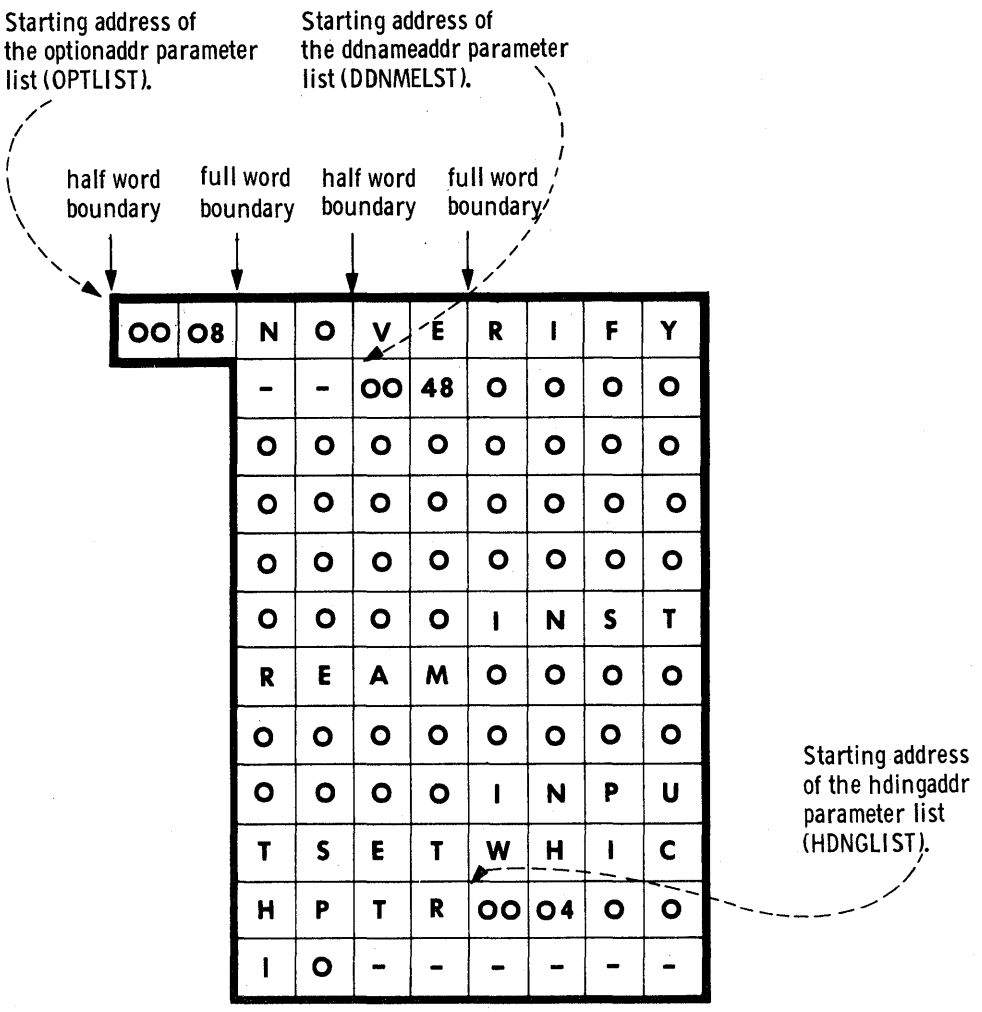
- The ddnameaddr list: includes the number of bytes in the list (48 hexadecimal) and alternative names for the SYSIN, SYSUT1, and SYSUT2 data sets (INSTREAM, INPUTSET, and WHICHPTR).

- The hdingaddr list: includes the number of bytes in the list (04 hexadecimal) and indicates the starting page number for printing operations controlled through the SYSPRINT data set.

Note that the symbolic starting addresses for the optionaddr and ddnameaddr parameter lists fall on halfword boundaries.



LINK/ATTACH Figure 1. Invoking a Utility Program



00	08	N	O	V	E	R	I	F	Y
		-	-	00	48	0	0	0	0
		0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0
		0	0	0	0	I	N	S	T
		R	E	A	M	0	0	0	0
		0	0	0	0	0	0	0	0
		0	0	0	0	I	N	P	U
		T	S	E	T	W	H	I	C
		H	P	T	R	00	04	0	0
		I	0	-	-	-	-	-	-

LINK/ATTACH Figure 2. Typical Parameter Lists

The IEBTCRIN utility can be invoked through use of the LOAD and CALL macro instructions as follows;

Name	Operation	Operand
[symbol]	LOAD	{EP=IEBTCRIN {EPLOC=address of name

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.

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.

Control can be passed to the load module via a CALL macro 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 ATTACH and LINK. The last parameter list address must contain an X'80' in byte 1 to indicate the last parameter in the list.

Name	Operation	Operand
[symbol]	CALL	IEBTCRIN(,optionaddr[,ddnameaddr][,hdingaddr]),VL

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, ddnameaddr, hdingaddr
: are the same as for ATTACH and LINK.

VL
: is written as shown. It causes the high order bit of the last address parameter in the macro expansion to be set to 1.

Appendix C: Control Statement Format and Notation

The control statements for the IBM System/360 Operating System utility programs have the following standard format:

Name	Operation	Operand
Optional symbolic name	Control statement type	Optional and required parameters

The name symbolically identifies the control statement and, with the exception of system utility program IEHINITT, can be omitted at the discretion of the utility user. 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 only be used 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.

A typical utility statement might appear as:

```
NAME OPERATION KEYWORD=information,...
```

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 can 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. An operand field with a nonblank character in column 71 will be treated as an interruption of a statement that exceeds 71 characters. 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 IEBCOPY, IEBPTPCH, IEBGENER, IEBCOMPR, and IEBDG utility programs permit certain exceptions to these requirements (see the applicable program writeup).

Notation for Defining Control Statements

The notation used to define control statements in this publication is described in the following paragraphs.

1. The set of symbols listed below are used to define control statements, but are never written in the actual statement.

hyphen	-
underscore	_
braces	{ }
brackets	[]
ellipsis	...

   The special uses of these symbols are explained in paragraphs 5-9.

2. Upper-case letters and words, numbers, and the set of symbols listed below are written in an actual control statement exactly as shown in the statement definition.

apostrophe	'
asterisk	*
comma	,
equal sign	=
parentheses	()
period	.

3. Lower-case letters, words, and symbols appearing in a control statement definition represent variables for which specific information is substituted in the actual statement.

   Example: If name appears in a statement definition, a specific value (for example, ALPHA) is substituted for the variable in the actual statement.

4. Stacked items represent alternatives. Only one such alternative should be selected.

   Example: The representation

   A
   B
   C

   indicates that either A or B or C should be selected.

5. Hyphens join lower-case letters, words, and symbols to form a single variable.

   Example: If member-name appears in a statement definition, a specific value (for example, BETA) is substituted for the variable in the actual statement.

6. An underscore indicates a default option. If an underscored alternative is selected, it need not be written in the actual statement.

Example: The representation

```
A
B
C
```

indicates that either A or B or C should be selected; however, if B is selected, it need not be written, because it is the default option.

7. Braces group related items, such as alternatives.

Example: The representation

```
         ⎧A⎫
ALPHA=( ⎨B⎬,D)
         ⎩C⎭
```

indicates that a choice should be made among the items enclosed within the braces. If A is selected, the result is ALPHA=(A,D). If B is selected, the result can be either ALPHA=(,D) or ALPHA=(B,D).

8. Brackets also group related items; however, everything within the brackets is optional and may be omitted.

Example: The representation

```
         ⎡A⎤
ALPHA=( ⎢B⎥,D)
         ⎣C⎦
```

indicates that a choice can be made among the items enclosed within the brackets or that the items within the brackets can be omitted. If B is selected, the result is: ALPHA=(B,D). If no choice is made, the result is: ALPHA=(,D).

9. An ellipsis indicates that the preceding item or group of items can be repeated more than once in succession.

Example:

```
ALPHA[,BETA]...
```

indicates that ALPHA can appear alone or can be followed by ,BETA any number of times in succession.

Appendix D: Defining Mountable Devices to be Used by System Utility Programs

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.

If the user is operating in a PCP environment, these considerations may not be critical. If however, the user is operating in an MFT or MVT environment, he must choose a combination of DD statement parameters that will ensure that volume integrity, as well as data set integrity, is maintained. In any case, extreme caution should be exercised when altering volumes which are permanently resident or reserved (e.g. volumes containing system data sets, non-dismountable devices, and volumes reserved through the PRESRES option).

Sharing Devices: Under normal conditions, the user will not want to "share" a mountable device with another job step; that is, if he is using a utility program to update a volume on a mountable device, he will want to ensure that the volume remain mounted until he is finished with it.

There are a number of ways in which the user can ensure that his mountable devices cannot be shared. One of these ways is by specifying DEFER in a DD statement defining a mountable device. Another is by specifying unit affinity on a second DD statement defining a mountable device. Additional methods are available, such as specifying a volume count (in the VOLUME parameter of a DD statement) that is greater than the number of mountable devices to be allocated. The following examples of DD statements use the DEFER parameter to ensure that a device is nonsharable. (The DEFER parameter is also used to support the deferred mounting of additional volumes on a single device.)

If the user is processing a single volume he can have the volume demounted and "signed out" to him by specifying PRIVATE in the DD statement defining the mountable device on which that volume is mounted. (The PRIVATE subparameter permits the demounting of a single volume at the end of a job step.)

If the user is processing multiple volumes through the use of a single DD statement, he must specify PRIVATE in the DD statement defining the mountable device on which the volumes are to be mounted, otherwise, the first volume that is mounted is processed correctly, but it cannot be demounted (i.e., the remaining volumes cannot be mounted).

If the public attribute is assigned to a volume (PRIVATE is not specified), that volume cannot be demounted until another job step requests the device on which it is mounted.

In the following examples of DD statements, an IBM 2311 Disk Storage Drive is indicated as the mountable device. Alternative parameters are stacked.

Statement 1:

```
//DDn   DD  UNIT=(2311,,DEFER),{DISP=OLD       },
                               {DISP=(,KEEP)   }
                               {DISP=(NEW,KEEP)},
//              VOLUME=(PRIVATE,,SER=(xxxxxx))
```

Attributes of the DD Statement: Specific, private, nonsharable.

This DD statement makes a specific request for a private, nonsharable volume or volumes to be mounted on a single 2311 device.

A utility program causes a mount message to be issued for a specific volume when the volume is required for processing by the program. Multiple volumes can be processed, one at a time.

The user should supply the operator with the volume or volumes that he wishes to have him mount during the course of the utility job step. These volumes should be clearly marked so that the operator knows which volume to mount when the utility program causes a mount message to be issued for a specific volume.

This DD statement ensures that the volume integrity of a mountable volume is maintained; that is, the specified volume or volumes can be mounted or demounted through the execution of the utility program only.

Notes: 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.

This DD statement is valid at all PCP, MFT, and MVT installations.

Statement 2:

```
//DDn   DD  UNIT=(2311,,DEFER),VOLUME=PRIVATE,{DISP=(,KEEP)   }
                                              {DISP=(NEW,KEEP)}
```

Attributes of the DD Statement: Nonspecific, private, nonsharable.

The operational results of this statement are identical to those of statement 1.

Caution: 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. When uncertain, the user should make a specific request, as in statement 1.

Statement 3:

```
//DDn   DD  UNIT=2311,VOLUME=(PRIVATE,,SER=(xxxxxx)),DISP=OLD
```

Attributes of the DD Statement: Specific, private, sharable.

This DD statement makes a specific request for one private, sharable volume to be mounted on a 2311 device. The DD statement does not ensure that volume integrity is maintained; that is, it should be used with extreme caution in an MFT or MVT environment. (If the statement is used in an MFT or MVT environment, there is the possibility that a concurrently running job step might make a specific request for the volume, use the volume, and demount it.)

Statement 4:

```
//DDn   DD  UNIT=(2311,,DEFER),VOLUME=SER=xxxxxx,{DISP=OLD       }
                                                 {DISP=(,KEEP)   }
                                                 {DISP=(NEW,KEEP)}
```

Attributes of the DD Statement: Specific, public, nonsharable.

This DD statement makes a specific request for one public, nonsharable volume to be mounted on a 2311 device. 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 some other 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.

This DD statement is valid at all PCP, MFT, and MVT installations.

Statement 5:

```
//DDn   DD  UNIT=2311,VOLUME=SER=xxxxxx,DISP=OLD
```

Attributes of the DD Statement: Specific, public, sharable.

This DD statement makes a specific request for one public, sharable volume to be mounted on a 2311 device. 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 some other job step requires the device on which the volume is mounted. (This DD statement is also used to define permanently resident devices.)

This DD statement does not ensure that the volume integrity of a mountable volume is maintained; that is, it should be used with extreme caution in an MFT or MVT environment. (If the DD statement is used to define a mountable device in an MFT or MVT environment, there is the possibility that a concurrently running job step might use the device.)

Appendix E: Generation Data Groups

A generation data group is a group of cataloged data sets that are chronologically or functionally related. Each data set within a generation data group is called a generation data set or, more simply, a generation.

The advantages of grouping related data sets are:

- All of the data sets in the group are referred to by a common name.
- The operating system keeps track of the relationship of each data set to the other data sets in the group.
- Outdated or obsolete generations can be automatically deleted by the operating system, if desired.

Data sets are grouped by cataloging them in a generation data group index. This index differs from a standard catalog index in that it is constructed so that the operating system can maintain a set of lower-level entries in the index. In this manner, the operating system can always have a record of the relative age of the generations that are represented in the lower-level index entries.

A generation data group can contain sequential, indexed sequential, partitioned, and direct data sets. A generation can reside on any volume that is compatible with its organization. For example, a partitioned generation can reside on any direct access volume but cannot reside on a sequential volume, such as magnetic tape.

Within a generation data group, the generations can have like or unlike DCB attributes and data set organizations. If the attributes and organizations of all generations in a group are identical, the generations can be retrieved together as a single data set.

PREPARING TO CATALOG A GENERATION DATA GROUP

Before you can catalog a generation or a group of generations, you must build a generation data group index in the SYSCTLG (catalog) data set. In addition, you should decide, at this time, how you are going to supply DCB attributes for the first generation that you create and catalog.

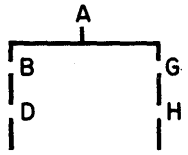
Building a Generation Data Group Index

Use the BLDG function of the IEHPROGM system utility program to build your generation data group index. The BLDG function builds the index, providing lower-level entries for as many generations (up to 255) as you would like to have in your generation data group. The BLDG function also serves the purpose of telling the operating system how to handle older or obsolete generations when the index is full. For example, when the index is full you may wish to empty it, scratch all of the existing generations, and begin the cataloging of a new series of generations. As an alternative, you may wish to scratch only the oldest generation, thus making room for a new generation.

Examples showing how to build a generation data group index are included in "Cataloging a Generation."

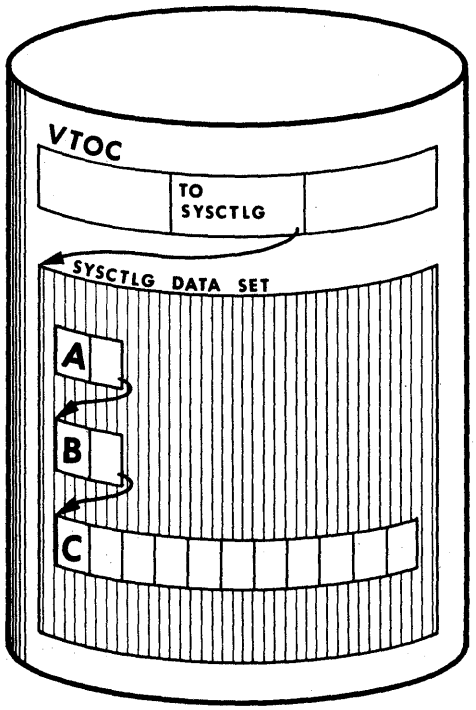
Generation Data Groups Figure 1 shows a generation data group index. When the index was built, provision was made for the subsequent cataloging of ten generations.

Note: You cannot build a generation data group index using index qualifiers identical to those in an existing, nongeneration index structure. For example, if the following index structure exists in the catalog:



you cannot build a generation data group index whose first three qualifiers are either A.B.D or A.G.H . You must either use a different name for your generation data group index, or delete the existing index structure.

System Residence Volume



Generation Data Groups Figure 1. A Generation Data Group Index

After the index is built, you can catalog a generation by its generation group name and either an absolute generation and version number or a relative generation number.

Absolute Generation and Version Numbers: An absolute generation and version number is used to identify a specific generation of a generation data group. The generation and version numbers are in the form GxxxxVyy, where xxxx is an unsigned 4-digit decimal generation number and yy is an unsigned 2-digit decimal version number. For example,

A.B.C.G0000V00 is generation zero, version zero of the generation data group A.B.C .

A.B.C.G0001V00 is generation one, version zero of generation data group A.B.C .

A.B.C.G0009V01 is generation nine, version one of generation data group A.B.C .

When you catalog a generation, using relative or absolute numbers, a generation and version number is placed as a low-level entry in the generation data group index. (The volume serial number of the volume containing the generation is also included.) In order to catalog a version number other than V00, you must use an absolute generation and version number when you catalog the generation.

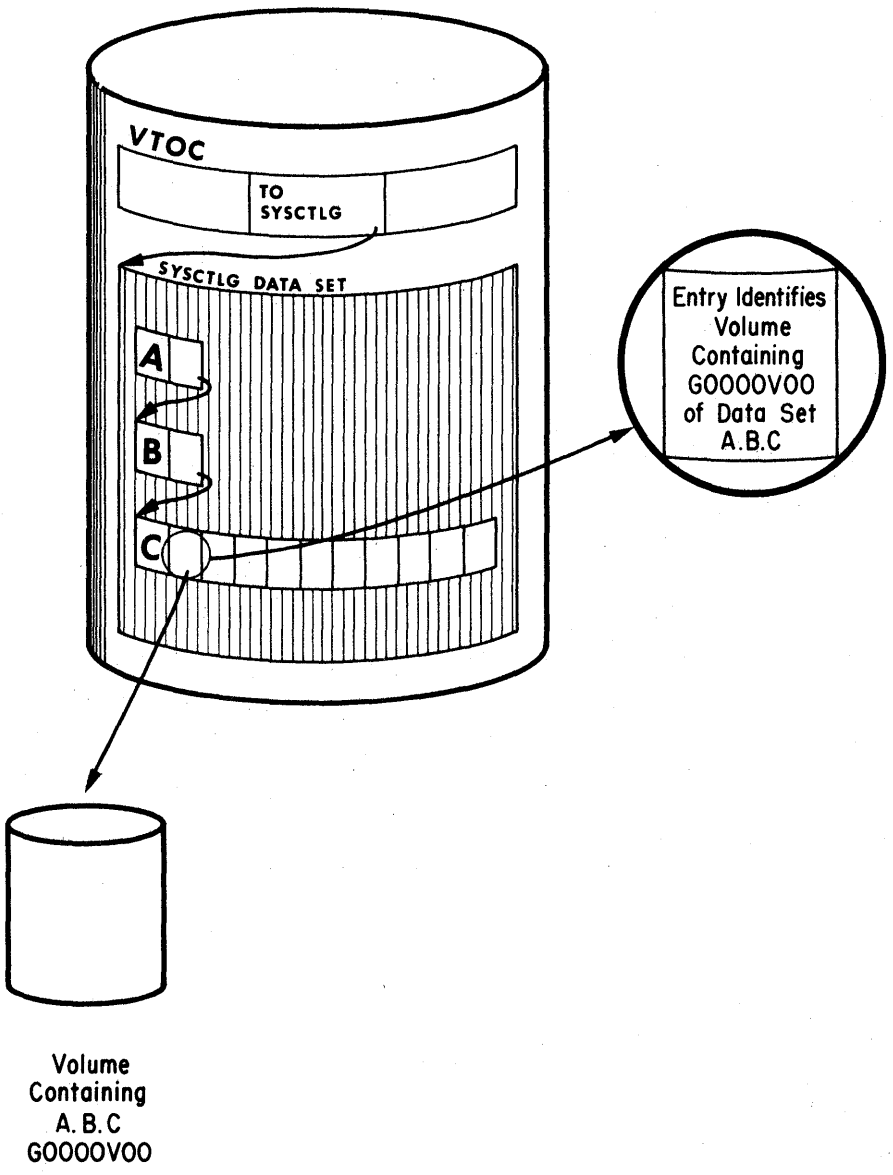
Generation Data Groups Figure 2 shows a generation data group index after generation A.B.C.G0000V00 is cataloged.

Generation Data Groups Figures 3 and 4 show how the index looks after additional generations are cataloged. Note that the low-level index entries are shifted. The operating system shifts them to allow you to use relative generation numbers when cataloging or retrieving a generation.

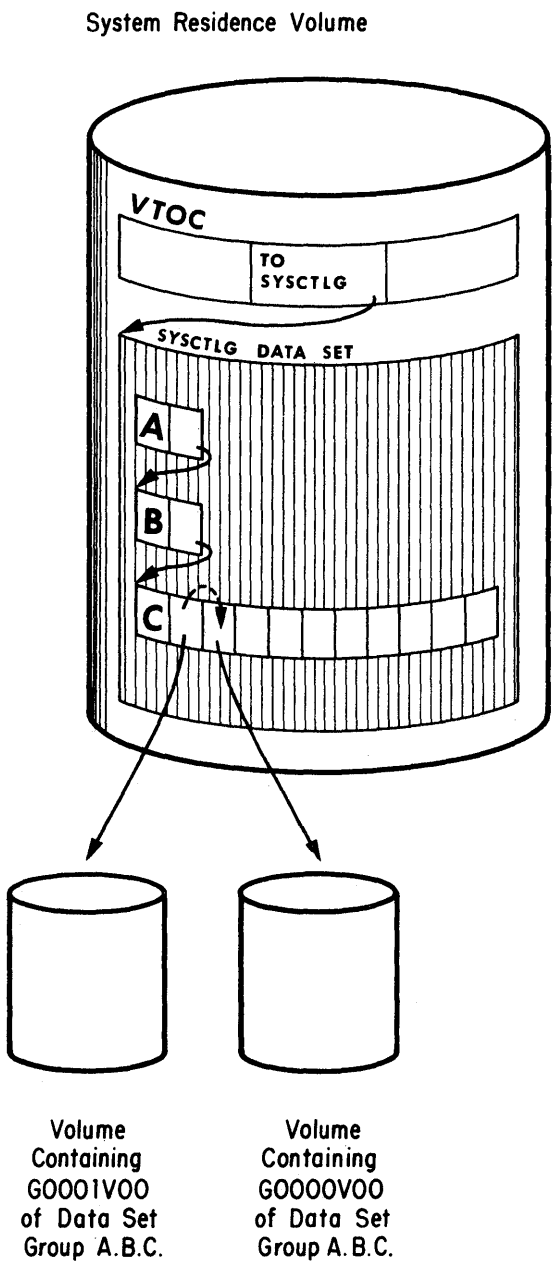
Note: A new version of a specific generation can be cataloged automatically by specifying the old generation number along with a new version number. For example, if generation A.B.C.G0005V00 is cataloged in the index and you now create and catalog A.B.C.G0005V01, the new entry is cataloged in the index location previously occupied by A.B.C.G0005V00. (This process removes the old entry from the catalog.)

Relative Generation Numbers: As an alternative to using absolute generation and version numbers when cataloging or referring to a generation, you can use a relative generation number. To specify a relative number, use the generation data group name followed by a negative integer, a positive integer, or a zero, enclosed in parentheses. For example:

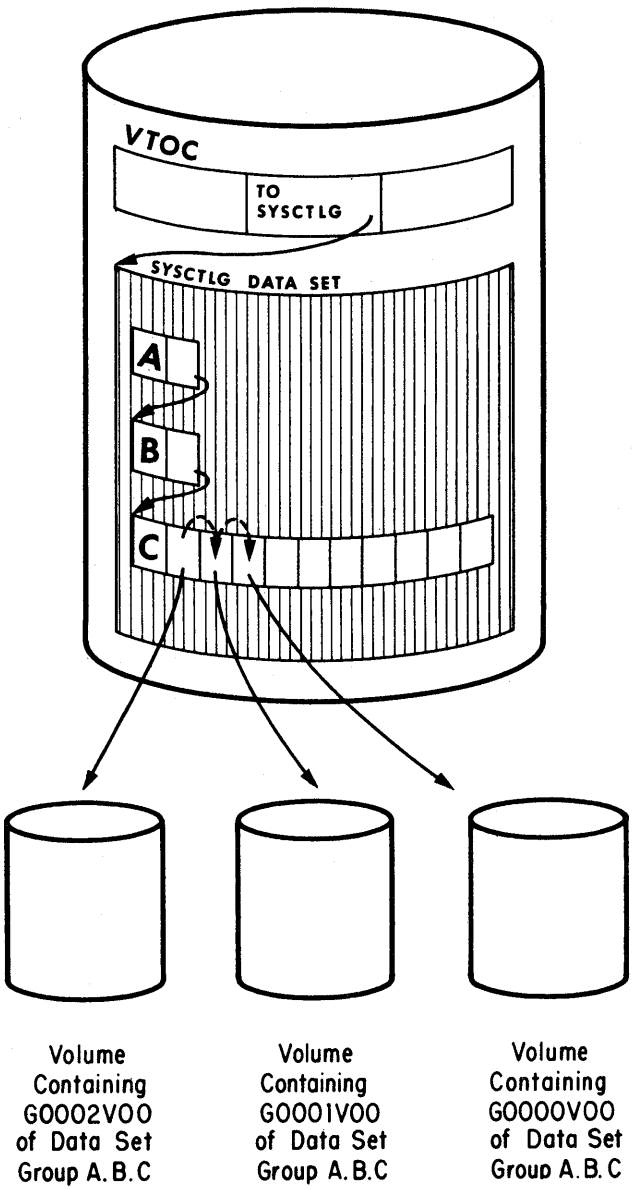
A.B.C(-1)
A.B.C(+1)
A.B.C(0)



Generation Data Groups Figure 2. A Generation Data Group Index -- One Entry



Generation Data Groups Figure 3. A Generation Data Group Index -- Two Entries



Generation Data Groups Figure 4. A Generation Data Group Index -- Three Entries

The value of the specified integer tells the operating system what generation number to assign to a new generation, or it tells the system the location (in the generation data group index) of an entry representing a previously cataloged generation.

When you use a relative generation number to catalog a generation, the operating system assigns an absolute generation number and a version number of V00 to represent that generation. The assigned generation number depends on the number last assigned and the value of the relative generation number that you are now specifying. For example, if in a previous job, generation A.B.C.G0005V00 was the last generation cataloged, and you now specify

A.B.C(+1),

The generation now cataloged is assigned the number G0006V00.

When you use a relative generation number to refer to a generation that was cataloged in a previous job, the relative number has the following meaning:

A.B.C.(0) refers to the latest existing cataloged entry.
A.B.C(-1) refers to the next-to-the-latest entry.
etc.

When you use a relative generation number to refer to a generation that was cataloged in a previous job step of your job, the relative number has the following meaning:

A.B.C(+1) refers to the latest existing catalog entry.
A.B.C(0) refers to the next-to-the-latest entry.
A.B.C(-1) refers to the third latest entry.
etc.

Note: A job step that terminates abnormally may be deferred for a later step restart. If the step cataloged a generation data set, you must change all relative generation numbers in the succeeding steps before resubmitting the job. For example, if the succeeding steps contained the relative generation numbers:

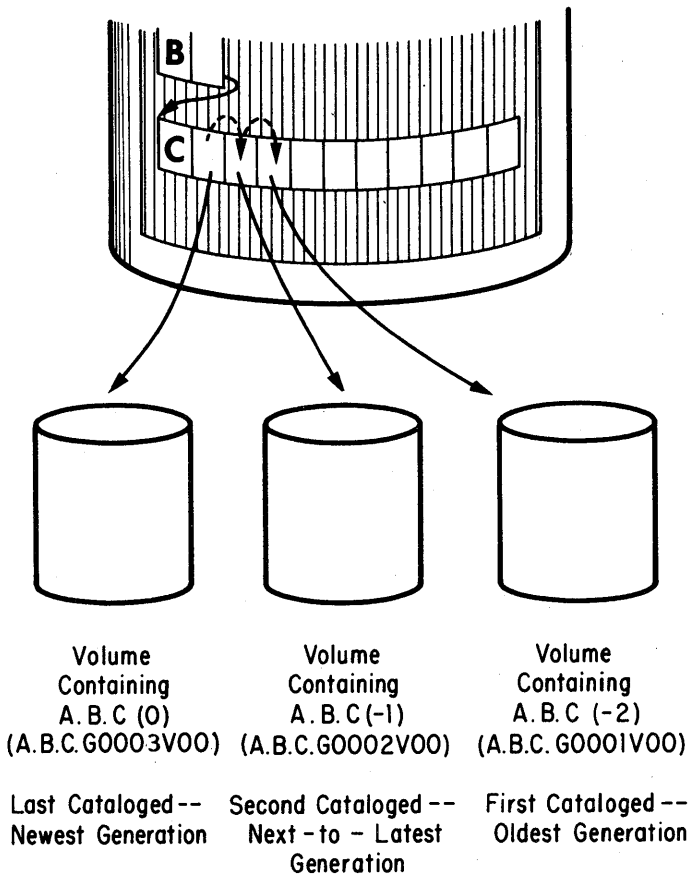
A.B.C(+1), referring to the entry cataloged in the terminated step,
A.B.C(0), referring to the next-to-latest entry,
A.B.C(-1), referring to the third latest entry,
etc.

You must change them as follows before the step can be restarted:

A.B.C(0)
A.B.C(-1)
A.B.C(-2)
etc.

For further information on deferred step restarts and checkpoint restarts, refer to the publication IBM System/360 Operating System: Concepts and Facilities, GC28-6535.

Generation Data Groups Figure 5 shows how an index looks after three generations -- A.B.C(+1),A.B.C(+1),and A.B.C(+2) -- have been cataloged. The first generation is assigned the generation number G0001V00; the second, G0002V00; the third, G0003V00.



Generation Data Groups Figure 5. Relative Positioning -- Three Entries in the Catalog

Providing DCB Attributes

If you are using absolute generation and version numbers, DCB attributes for the generation can be supplied directly in the DCB parameter of the DD statement defining the generation to be created and cataloged.

However, if you are planning to use relative generation numbers when you catalog your generations, you must plan to specify DCB attributes in an alternative manner. DCB attributes can be supplied either (1) by creating a model DSCB on the volume on which the index resides (the volume containing the SYSCTLG data set), or (2) by referring to a cataloged data set for the use of its attributes. Attributes can be supplied before you catalog your generation, when you catalog it, or at both times, as follows:

1. Create a model DSCB on the volume on which your index resides. You can provide initial DCB attributes when you create your model; however, you need not provide any attributes at this time. Initial or overridding attributes can be supplied when you create and catalog your generation.[1]

---

[1]Only one model DSCB is necessary for any number of generations. If you plan to use only one model, do not supply any DCB attributes when you create the model. When you subsequently create and catalog a generation, include necessary DCB attributes in the DD statement referring to the generation. Note that in this manner, any number of generation data groups can refer to the same model.

To create a model DSCB, include the following DD statement in the job step that builds the index or in any other job step that precedes the step in which you create and catalog your generation:

```
//name DD DSNAME=datagrpname,DISP=(,KEEP),SPACE=(TRK,(0)),
//        UNIT=xxxx,VOLUME=SER=xxxxxx,
//        DCB=(applicable subparameters)
```

where
datagrpname is the common name by which each generation is identified, and xxxxxx is the serial number of the volume containing the catalog.

Note: If no DCB subparameters are desired initially, you need not code the DCB parameter.

2. You do not need to create a model DSCB if you can refer to a cataloged data set whose attributes are identical to those you desire or to an existing model DSCB for which you can supply overriding attributes*. A cataloged data set referred to in this manner must reside on the same volume as your index.

   To refer to a cataloged data set for the use of its attributes, specify DCB=(dsname) on the DD statement that creates and catalogs your generation. To refer to an existing model, specify DCB=(modeldscbname, your attribute) on the DD statement that creates and catalogs your generation.

CATALOGING A GENERATION

You can catalog a generation through the use of normal job control language procedures or through the use of the IEHPROGM system utility program.

Using JCL Procedures to Catalog a Generation

Assuming that a generation data group index has been built and that provisions have been made for supplying DCB attributes, a generation is created and cataloged in the same manner as any other type of data set; that is, all of the normal data set manipulations apply to a generation.

When you use relative numbers in your job control language procedures, you must include the CATLG subparameter in the DD statement defining the new generation. When you use absolute generation and version numbers, you need not catalog the new generation immediately.

Using the IEHPROGM Program to Catalog a Generation

You can use the CATLG function of the IEHPROGM system utility program to catalog a generation. Again, the prerequisite for cataloging a generation is the existence of a generation data group index in the SYSCTLG (catalog) data set.

Note: You must always use an absolute generation and version number if you catalog or uncatalog a generation with the IEHPROGM program. (The IEHMOVE and IEHLIST system utility programs also require that absolute generation and version numbers be used.)

Creating an ISAM Data Set as Part of a Generation Data Group

To create an ISAM data set as part of a generation data group, you must:

1. Create the ISAM data set separately from the generation data group.
2. Use the IEHPROGM utility to put the ISAM data set into the generation group.

Use the RENAME function to rename the data set. Then use the CATLG function to catalog the data set. For instance, if MASTER is the name of the generation data group, and GggggVvv is the absolute generation name, you would code the following:

```
RENAME DSNAME=ISAM,VOL=2314=SCRTCH,NEWNAME=MASTER.GggggVvv

CATLG DSNAME=MASTER.GggggVvv,VOL=2314=SCRTCH
```

RETRIEVING A GENERATION

A generation is retrieved through the use of job control language procedures. Any operation that can be applied to a nongeneration data set can be applied to a generation. For example, a generation can be updated and then reentered in the catalog, or it can be copied, printed, punched, or used in the creation of new generation or nongeneration data sets.

You can retrieve a generation by using either relative generation numbers or absolute generation and version numbers.

MULTIPROGRAMMING CONSIDERATIONS

Since in a multiprogramming environment, two or more jobs can compete for the same resource (which includes a generation), generation data groups should be updated with caution.

1. No two jobs running concurrently should refer to the same generation data group. As a partial safeguard against this situation, you should use absolute generation and version numbers when cataloging or retrieving a generation in a multiprogramming environment. Otherwise, if you use relative numbers, a concurrently running job may update the generation data group index -- perhaps cataloging a new generation which you will then retrieve in place of the one you wanted.
2. Even when using absolute generation and version numbers, a concurrently running job might catalog a new version of a generation or perhaps delete the generation you wished to retrieve. For this reason, some degree of control should be maintained over the execution of job steps referring to generation data groups.

Generation Data Groups Examples

The following examples show some of the ways in which generations can be created and cataloged or retrieved and used as source data in the creation of new generation or nongeneration data sets.

Generation Example 1

In this example, STEPA, an IEHPROGM job step, creates a model DSCB and builds a generation data group index. STEPB, an IEBGENER job step, creates and catalogs a sequential generation from card input.

STEPA

- The BLDDSCB DD Statement: creates a model DSCB on the system residence volume.

- The SYSIN DD Statement: indicates that a utility control statement (BLDG) is included next in the input stream.

- The BLDG Utility Control Statement: specifies the generation group name A.B.C and makes provision for ten lower-level entries in the index. When the index subsequently becomes full, it is to be emptied, and all of the generations are to be deleted.

Note: The 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, new generations will continue to be assigned generation numbers according to the last generation number assigned prior to the empty operation. To restart the numbering operation (i.e., to reset to G0000V00 or G0001V00), delete the generation data group index and build a new one.

STEPB

- The SYSUT2 DD Statement: defines an output sequential generation. The generation is assigned the absolute generation and version number G0001V00 in the index.

- The SYSUT1 DD Statement: defines the input card data set.

```
//BLDINDX  JOB
//STEPA    EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//BLDDSCB  DD   DSNAME=A.B.C,DISP=(,KEEP),SPACE=(TRK,(0)),UNIT=2311,
//             VOLUME=SER=111111,DCB=(LRECL=80,RECFM=FB,BLKSIZE=800)
//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=2311,DISP=(,CATLG),SPACE=(TRK,20),
//             VOLUME=SER=231100
//SYSUT1   DD   DATA
           .
           .
           input card data
           .
           .
/*
```

Generation Example 1. Building a Generation Data Group Index and Creating the First Generation From Card Data

Generation Example 2

In this example, a second generation is created and cataloged in the index built in Example 1. DCB attributes are included to override those attributes that were specified when the model DSCB was created.

- The SYSUT2 DD Statement: 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.

  Note: 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.

- The SYSUT1 DD Statement: defines the input card data set.

```
//          JOB
//          EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DUMMY
//SYSUT2   DD   DSNAME=A.B.C(+1),UNIT=2311,DISP=(,CATLG),SPACE=(TRK,20),
//              VOLUME=SER=231101,DCB=(LRECL=80,RECFM=FB,BLKSIZE=1600)
//SYSUT1   DD   DATA
             .
             .
             input card data
             .
             .
/*
```

Generation Example 2. Creating a Second Generation From Card Data -- Providing Overriding Attributes

Generation Example 3

In this example:

1. A generation data group index for generation data group A.B.C is built.
2. Three existing noncataloged, nongeneration data sets are renamed.
3. The renamed data sets are cataloged as generations in the generation data group index.

- The DD1 DD Statement: defines the system residence volume, on which the SYSCTLG (catalog) data set resides.
- The BLDG Utility Control Statement: 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 Utility Control Statements: rename three nongeneration data sets residing on a 2311 disk volume.
- The CATLG Utility Control Statement: catalog the renamed data sets in the generation data group index.

Note: Because the DCB parameters were supplied when the nongeneration data sets were created, no DCB parameters are now specified; hence, no model DSCB is required.

```
//BLDINDEX JOB
//         EXEC PGM=IEHPROGM
//SYSPRINT DD   SYSOUT=A
//DD1      DD   UNIT=2311,VOLUME=SER=111111,DISP=OLD
//DD2      DD   UNIT=(2311,,DEFER),DISP=OLD,
//              VOLUME=(PRIVATE,,SER=(231100))
//SYSIN    DD   *
       BLDG     INDEX=A.B.C,ENTRIES=10
       RENAME   DSNAME=DATASET1,VOL=2311=231100,NEWNAME=A.B.C.G0001V00
       RENAME   DSNAME=DATASET2,VOL=2311=231100,NEWNAME=A.B.C.G0002V00
       RENAME   DSNAME=DATASET3,VOL=2311=231100,NEWNAME=A.B.C.G0003V00
       CATLG    DSNAME=A.B.C.G0001V00,VOL=2311=231100
       CATLG    DSNAME=A.B.C.G0002V00,VOL=2311=231100
       CATLG    DSNAME=A.B.C.G0003V00,VOL=2311=231100
/*
```

Generation Example 3. Renaming Nongeneration Data Sets and Cataloging Them as Generations

Generation Example 4

In this example, a nongeneration version of a generation data set is to be made. The generation is represented as the next-to-the-latest entry in the generation data group index.

The name of the resultant data set is changed from A.B.C.GxxxxVxx to TESTSET. This example assumes that the generation to be copied is partitioned.

- The SYSUT1 DD Statement: defines the generation from which a copy is to be made.
- The SYSUT2 DD Statement: defines a resultant partitioned data set (TESTSET) on a 2311 output volume. The DCB attributes in this statement are identical to those assigned to the generation. (Reblocking is permissable; however, the SYSUT2 block size specification must be a multiple of the original block size.)

```
//COPY     JOB
//         EXEC PGM=IEBCOPY
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=A.B.C(-1),DISP=(OLD,CATLG)
//SYSUT2   DD   DSNAME=TESTSET,UNIT=2311,DISP=(,KEEP),
//              VOLUME=SER=231100,SPACE=(TRK,(20,10,5)),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=80)
//SYSIN    DD   DUMMY
/*
```

Generation Example 4. Creating a Nongeneration Copy of a Partitioned Generation

Generation Example 5

In this example, a partitioned generation, consisting of three members, is to be used as source data in the creation of a new generation. The IEBUPDTE data set utility program 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.

- The SYSUT1 DD Statement: defines the latest generation, which is used as source data.
- The SYSUT2 DD Statement: defines the new generation, which is created from the source generation and from an additional member included as input card data.
- The REPRO Utility Control Statements: reproduce the named source members in the output generation.
- The ADD Utility Control Statement: specifies that the data cards following the input stream be included as MEM4.
- The NUMBER Utility Control Statement: indicates that the new member is to have sequence numbers assigned in columns 73-80. The first record is assigned sequence number 10. The sequence number of each successive record is incremented by 5.
- The ENDUP Utility Control Statement: 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.

```
//          JOB
//          EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSNAME=A.B.C(0),DISP=(OLD,CATLG)
//SYSUT2   DD   DSNAME=A.B.C(+1)DISP=(,CATLG),UNIT=2311,
//              VOLUME=SER=231100,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,IGNORE=EOF
./      NUMBER   NEW1=10,INCR=5
             .
             .
             data cards comprising MEM4
             .
             .
./      ENDUP
/*
```

Generation Example 5. Modifying the Latest Generation and Cataloging it as a New Generation

Appendix F: Utility Program Handling of User Labels

Six utility programs can process user labels. They are IEBGENER, IEBCOMPR, IEBPTPCH, IEHMOVE, IEBCTRIN, and IEBUPDTE. Following is a general discussion of the manner in which these programs process user labels. Exceptions are noted both here and in the discussions of the IEHMOVE, IEBGENER and IEBUPDTE programs. In general, user label support allows the utility program user to:

- Process user labels as data set descriptors.
- Process user labels as data.
- Exit to a user's routine for totaling the processed records prior to each WRITE command (IEBGENER and IEBUPDTE only).

For either of the first two options, the user must specify standard user 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.

Processing User Labels as Data Set Descriptors

Under this option one of the user's label processing routines receives control for each user label of the specified type. The user's routine can then include, exclude, or modify the user label. The user specifies this option by providing an EXITS statement with one or more of the following keyword parameters:

INHDR=routinename
(name of user routine to process input header labels)

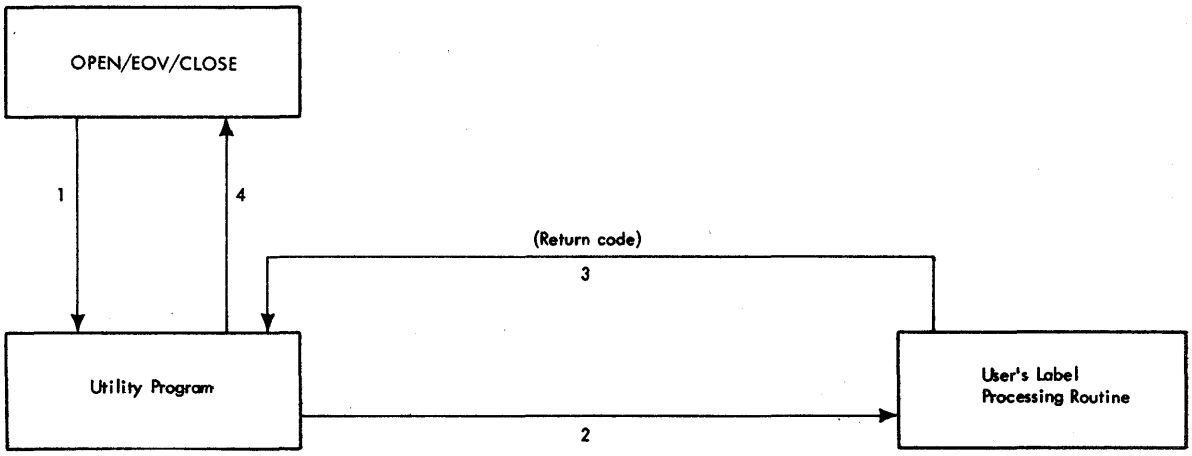
INTLR=routinename
(name of user routine to process input trailer labels)

OUTHDR=routinename
OUTHDR2=routinename
OUTHDR3=routinename
(name of user routine to process output header labels)

OUTTLR=routinename
OUTTLR2=routinename
OUTTLR3=routinename
(name of user routine to process output trailer labels)

These 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.

User Labels Figure 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), it passes control to the utility program, which then, if an exit has been specified for this type of label, passes control to the user routine. The user routine processes the label as desired and returns control, along with a return code, to the utility program. The utility program then returns control to OPEN, EOV, or CLOSE.



User Labels Figure 1. System Action at OPEN, EOV, or CLOSE Time

This cycle is repeated up to 8 times, depending upon the number of user labels in the group and the return codes supplied by the user's routine.

The user's label processing routine receives control with the following information in the general registers:

Register	Contents
0	no meaningful information
1	address of parameter list described below
2-13	utility registers
14	return address in utility program
15	entry point address in user routine

The user's label processing routine must save all registers and restore them before returning control to the utility program. The user's routine must not use the save area to which register 13 points.

User Labels Figure 2 shows the format of the parameter list which the user routine receives from the utility program.

Flag Byte	Address Constant (3 bytes)
	Address of the label buffer
Flag 1	Address of the DCB being processed
Flag 2	Address of the status information (if an uncorrectable I/O error occurs)
	Address of totaling area

User Labels Figure 2. Format of Parameter List Passed to User's Label Processing Routine

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. (Except when the utility program has been requested to generate labels. In that case, the buffer contains the newly generated label.) The label processing routine constructs a label in the label buffer.

If standard user labels (SUL) have been specified on the DD statement for a data set, but the data set has no user labels, the system will still take 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.

Bit 0 of Flag 1 will be set to its normal 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 I/O error occurs while reading or writing a user label, the appropriate label processing routine is entered with bit 0 of flag 2 set on. The low order three bytes of this parameter contain the address of the standard status information as supplied for SYNAD routines. (The SYNAD routine is not entered.)

The code returned by the user's label processing routine determines system response as follows:

Routine Type	Return Code	System Response
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 less 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.

The user's routine must return one of these codes in register 15 unless:

1. The buffer address was set to zero before entry to the label processing routine. In this case, the system will resume normal processing regardless of return code.

2. The user's label processing routine was entered after an uncorrectable output error occurred. In this case the system will always attempt to resume normal processing.

Note: Slightly different return codes are used for the UPDATE=INPLACE option of the IEBUPDTE program. See the discussion of the IEBUPDTE program.

Processing User Labels as Data

Under this option 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 this option the user should include a utility control statement with the following format:

Name	Operation	Operand
[name]	LABELS	DATA={YES / NO / ALL / ONLY / INPUT}

YES
: The utility program will process as data any user labels that are not rejected by a user's label processing routine. The utility program will stop processing labels as data in compliance with standard return codes. This is the default option. Even in the absence of user label processing routines or a LABELS card, user labels will be printed, punched, compared or copied (provided, of course, that standard user labels (SUL) were specified on the DD statement).

NO
: The utility program will not process user labels as data.

ALL
: The utility program will process all user labels as data regardless of any return code. A return code of 16 requires utility termination after all labels in the current group have been processed. For example, if a user routine returns a code of 16 while the utility is processing a group of user header labels, the utility will process the remainder of the group and then terminate the job step.

ONLY
: The utility program will process as data only user header labels. All user header labels will be processed as data regardless of any return code. The job will terminate upon return from the OPEN routine.

INPUT
: This option is valid only for IEBGENER. The user must supply the user labels for the output data set as 80-byte input records in the data portion of SYSIN. These user labels must be identified by RECORD LABELS=n statements, where n indicates the number of input records following this statement that should be treated as user labels.

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, as indicated above. In addition, a user's output label processing routine has the opportunity to overrule the action of a LABELS statement since 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 in any desired manner.

Exiting To a User's Totaling Routine

This option passes an output record (see following note) to the user's routine just prior to writing the record on the data set. 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) will be pointed to by the parameter list passed to the appropriate user label routine. If, during the processing of the utility, an incorrectable I/O error occurs, the user totaling routine is entered with bit zero of flag two set ON. The low order three bytes contain the address of the standard status information as supplied for SYNAD routines.

The code returned by the user's totaling routine determines system response as follows:

Return Code	System Response
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	Terminate.

Note: An output record is defined as a physical record (block), except when IEBGENER is used to process and reformat a data set containing spanned records.

IEBUPDTE and IEHMOVE

The user cannot update labels by means of the IEBUPDTE program. This function must be performed by user label processing routines. IEBUPDTE will, however, allow the user to create labels on the output data set from data supplied in the input stream. See the discussion of the LABEL statement in IEBUPDTE.

IEHMOVE does not allow exits to user routines and will not recognize options concerning the processing of user labels as data. IEHMOVE will always move or copy user labels directly onto a new data set. See the discussion of the IEHMOVE program in this book.

Volume Switch Labels

Volume switch labels of a multivolume data set cannot be processed by the IEHMOVE, IEBGENER, or IEBUPDTE program. Volume switch labels are therefore lost when these utilities create output data sets. If you want to ensure that volume switch labels will be retained, process your multivolume data sets one volume at a time.

Appendix G: Utility Program Messages

The description of most messages includes a programmer response. A more detailed response is included in the publication IBM System/360 Operating System: Messages and Codes, GC28-6631. Refer to this publication before responding to any message or calling IBM.

This appendix contains messages that are issued by the system, data set, and independent utility programs. Special considerations are to be noted for data set utility and independent utility messages:

- Data set utility messages indicating job termination can be interpreted several ways:
  1. If the utility program was invoked, a return code is passed to the calling program with the option to terminate.
  2. If the utility program represents one step of a multistep job, the step is terminated.
  3. In all other cases, the job is terminated.
- Independent utility messages are of two types: error and diagnostic. Error messages describe error conditions associated with the utility programs, while diagnostic messages describe and locate faulty conditions associated with the hardware.

Independent Utility Messages

Error Messages for DASDI and DUMP/RESTORE

Error messages are listed in alphameric order with explanations and, when applicable, a recommended response.

IBC101W INVALID CARD CODE. CORRECT ERROR. DEPRESS INTERRUPT KEY.

Explanation: An invalid card code appears in the above card.

IBC102A CONTROL STATEMENT ERROR. JOB TERMINATED.

Explanation: A utility control statement contains an incorrect keyword, parameter, or name field.

IBC103A STATEMENT SEQUENCE ERROR. JOB TERMINATED.

Explanation: The utility statements are not in the proper sequence, or unnecessary utility statements are present.

IBC104W SVC INTERRUPT. JOB TERMINATED.

Explanation: An SVC was initiated without a response having been defined.

IBC105A DEFINE INPUT DEVICE.

Response: Enter the following message from the console typewriter: INPUT=dddd cuu, where dddd is the device type and cuu is the channel and unit address of the input device.

IBC106A THE VOLID IN CONTROL STATEMENT DOES NOT AGREE WITH ID IN VOL LABEL WHICH FOLLOWS. VOLID=xxx.

Explanation: The VOLID parameter in the utility control statement did not match the volume serial number found on the receiving volume (xxx).

Response: Correct statement or mount correct volume and restart program.

IBC107W TRACK ZERO BAD. JOB TERMINATED.

Explanation: The device cannot be initialized as a systems residence volume due to a defective surface on cylinder 00, track 00.

IBC108A HA OR R0 FIELD BAD. JOB TERMINATED.

Explanation: The device cannot be initialized due to a bad surface area in the home address or track descriptor record areas.

IBC108I HA OR R0 FIELD BAD

Explanation: The home address or record zero was defective and has been moved down the track (on 2314 Disk and 2321 Data Cell only). The defective track and the alternate track assigned are listed.

IBC109I TRACK CHK INDICATES TRACK IS GOOD.

Explanation: The track in question is good and no alternate was assigned.

IBC110I BAD TRACK ccccchhhh.

Explanation: A defective track was found at the specified location (cccc is the cylinder number, hhhh is the head number).

IBC111I ALTERNATE {ccccchhhh / NONE}

Explanation: An alternate track at the specified location (ccccchhhh) is assigned to replace the defective track (cccc is the cylinder number and hhhh is the head number of the alternate track). If no location is specified with this message, either the defective track is in the alternate track area or the applicable device is a drum device.

IBC112W ALT TRACKS DEPLETED. JOB TERMINATED.

Explanation: The number of alternate tracks assigned has exceeded the maximum number for this device.

IBC113W IMPROPER VTOC BEGIN ADDRESS. JOB TERMINATED.

Explanation: The starting address for the VTOC can not be track 0 for any direct access device or track 1 for the 2302 or 2311 devices if IPL text is written by the program.

User Response: Change the STRTADR parameter on the VTOCD statement to use another track.

IBC151W MACHINE CHECK. JOB TERMINATED.

Explanation: A machine malfunction has caused a machine interrupt resulting in termination of the job.

IBC152W PROGRAM INTERRUPT. JOB TERMINATED.

Explanation: A program interrupt has occurred resulting in termination of the job.

IBC153A TYPEWRITER FAILED TO READ LAST MESSAGE. DEPRESS INTERRUPT KEY.

Explanation: The console typewriter failed to read the input message.

Response: Depress the console interrupt key and attempt to enter the input message again.

IBC154A READY READER cuu. DEPRESS INTERRUPT KEY.

Explanation: The reader has a card jam, a transport jam, or is out of cards.

Response: Correct the faulty condition and depress the console interrupt key to continue the program.

IBC155A READY PRINTER cuu. DEPRESS INTERRUPT KEY.

Explanation: The printer is not ready due to a forms check, an open interlock, or a depressed stop key.

Response: Correct the faulty condition and depress the console interrupt key to continue the program.

IBC156A READY TAPE cuu. DEPRESS INTERRUPT KEY.

Explanation: The tape drive on channel c, unit uu is not ready.

Response: Correct the faulty condition and depress the console interrupt key.

IBC157A READY DASD cuu. DEPRESS INTERRUPT KEY.

Explanation: The direct access device on channel c, unit uu is not ready.

Response: Correct the faulty condition and depress the console interrupt key.

IBC158A WRONG TAPE ON cuu. MOUNT PROPER TAPE. INTERRUPT.

Explanation: The tape on the specified device does not pertain to this job.

Response: Mount the correct tape and depress the interrupt key.

IBC159A READER CHECK. CORRECT ERROR. DEPRESS INTERRUPT KEY.

Explanation: A reader check has occurred.

Response: Correct the faulty condition and clear the reader check. Continue the program by depressing the console interrupt key.

IBC160A PRINT CHECK. CORRECT ERROR. DEPRESS INTERRUPT KEY.

Explanation: A print check has occurred.

Response: Correct the faulty condition and clear the print check. Depress the console interrupt key to continue the program.

IBC161A END OF TAPE. MOUNT TAPE ON cuu. DEPRESS INTERRUPT KEY.

Explanation: End of present tape reel.

Response: Mount another tape volume on the active tape device, i.e., the TODEV device for DUMP operations or the FROMDEV device for RESTORE operations.

IBC162A MOUNT ANOTHER PACK ON UNIT cuu. DEPRESS INTERRUPT KEY.

Explanation: End of the present disk pack.

Response: Mount another disk pack on the active disk drive, i.e., the TODEV device for DUMP operations or the FROMDEV device for RESTORE operations.

IBC163A END OF JOB.

Explanation: A normal end-of-job condition has occurred.

IBC164A VOLUME LABEL COULD NOT BE READ.

Explanation: An error was encountered while reading the standard volume label.

System Action: Volume cannot be identified; job terminates.

IBC165A ATTEMPT TO RESTORE TO WRONG DEVICE

Explanation: The DUMP/RESTORE program attempted to restore data to a device type other than the type from which it was dumped.

System Action: Job terminates.

IBC166A NOT A RESTORE VOL. ON cuu. MOUNT PROPER VOLUME. DEPRESS INTERRUPT KEY.

Explanation: A volume other than a restore volume is mounted on the named device.

Response: Mount the correct volume and depress the interrupt key.

IBC167A SEEKED BALLAST CELL. MOUNT PROPER CELL. DEPRESS INTERUPT KEY.

Explanation: A ballast cell is mounted in the bin requested in a utility control statement.

Response: Mount the proper cell and continue the program by depressing the interrupt key.

IBC168I TRACK 0 HAS AN ALTERNATE ASSIGNED. VOLUME HAS BECOME NON-IPL-ABLE

Explanation: Track 0 has been flagged as a defective track. (This volume is usable as a work volume, but not as a system residence volume.)

Diagnostic Messages for Independent Utilities

Diagnostic messages appear in the following format:

number (16-byte text) cuu xx ssss yyyyyyyyyyyy cccchhhh

where c is the channel of the device in error, uu is the unit, xx is the command code, ssss are the status bytes from the channel status word, yyyyyyyyyyyy are the sense bytes, and cccchhhh is the track address of the direct access device being used when the failure occurred.

The following message texts are listed in alphameric order. All of the messages except IBC202A describe conditions that cause termination of the job.

IBC201W COMMAND REJECT.

Explanation: The specified channel has rejected an incorrect channel command word (CCW) list.

IBC202A INTERV. REQUIRED.

Explanation: The specified device is not ready.

Response: The specified device requires operator intervention to make it ready.

IBC203W BUS. OUT CHECK.

Explanation: A bus out check has occurred on the specified channel.

IBC204W EQUIPMENT CHECK.

Explanation: An equipment failure has occurred.

IBC205W DATA CHECK.

Explanation: A solid data check has occurred on the specified device.

IBC206W OVERRUN.

Explanation: An overrun check has occurred on the specified channel.

IBC207W FLAGGED TRACK.

Explanation: A track condition check has occurred on the specified device.

IBC208W DATA CONV. CHECK.

Explanation: A data converter check has occurred on the specified device.

IBC209W END OF CYLINDER.

Explanation: An unusual end of cylinder condition has occurred on the specified device.

IBC210W INVALID ADDRESS.

Explanation: An invalid address has been issued to the specified device.

IBC211W NOT AVAILABLE.

Explanation: The specified device is not attached to the system.

IBC212W READ DATA CHECK.

Explanation: A permanent read data check has been detected on the specified tape unit.

IBC213W COUNT FIELD CHECK.

Explanation: A data check has occurred in the count field of the specified direct access device.

IBC214W TRACK OVERRUN.

Explanation: A track overrun condition has occurred.

IBC215W FILE PROTECTED.

Explanation: The specified device is file protected.

IBC216W DASD-END OF FILE.

Explanation: An unusual end of file has occurred on the specified direct access storage device.

IBC217W NO RECORD FOUND.

Explanation: A no record found condition has occurred on the specified direct access device.

IBC218W INVALID ERROR.

Explanation: An invalid error return has occurred.

IBC219W WRONG ERROR.

Explanation: The error return is valid but is not associated with the specified device.

IBC220W CHAN. CTRL ERROR.

Explanation: A channel control check has occurred on the specified channel.

IBC221W INTERFACE ERROR.

Explanation: An interface control check has occurred on the specified channel.

IBC222W CHAN. DATA CHECK.

Explanation: A channel data check has occurred on the specified channel.

IBC223W DASD OVERFLOW.

Explanation: An overflow incomplete condition has occurred on the specified direct access device.

IBC224W PROGRAM CHECK.

Explanation: A program check has occurred due to an incorrect channel command word (CCW).

IBC225W PROTECTION CHECK.

Explanation: A protection check has occurred on the specified device.

IBC226W UNIT EXCEPTION.

Explanation: A unit exception has occurred on the specified unit.

IBC227W INCORRECT LENGTH.

Explanation: A wrong length record condition has occurred on the specified unit.

IBC228W CHAINING CHECK.

Explanation: A chaining check has occurred on the specified channel.

IBC229W COMMAND SEQ. ERR.

Explanation: An invalid sequence of channel command words (CCWs) was issued.

IBC230W SEEK CHECK ERROR.

Explanation: An invalid SEEK address was issued, or a unit malfunction caused a SEEK check.

IBC231W WRITE DATA CHECK.

Explanation: A permanent write data check has occurred on the specified tape unit.

IBC232W TAPE -- LOAD POINT.

Explanation: A tape at load point condition has occurred on the specified tape unit.

IBC233W NOISE RECORD.

Explanation: A noise record was found on the specified tape unit.

IBC234W MISSING ADR-MARK.

Explanation: A missing address marker has occurred on the specified device.

IBC235W BLANK TRACK.

Explanation: A blank track has been encountered on the specified data cell.

IBC236W 3 BLANK CYLINDERS.

Explanation: Three blank cylinders have been encountered during the analysis of a strip.

IBC237W 3 BLANK STRIPS.

Explanation: Three blank strips have been encountered within one subcell.

IBC238W 3 BLANK SUBCELLS.

Explanation: Three blank subcells have been encountered within a cell.

IBC239W 3 BLANK TRACKS.

Explanation: Three blank tracks have been encountered within one cylinder.

IBC249W I/O ERROR, JOB TERMINATED.

Explanation: This message follows all messages that describe input/output error conditions.

Error Messages for RECOVER/REPLACE

IBC300I TRACK HAD BAD {R0|HA}

Explanation: An error was encountered while reading either the home address (HA) or the track descriptor record (R0).

IBC301I ADDRESS MARKER MISSING AFTER xxx

Explanation: The RECOVER routine found that an address marker was missing after the specified record.

IBC302I UNEXPECTED EOF INTERRUPT. EOP

Explanation: An unexpected end of file was generated.

System Action: End-of-pass (EOP). The RECOVER program continues to the next statement.

IBC303I TRACK HAS HA AND R0 ONLY. EOP

Explanation: Track has only home address and track descriptor record, and was not flagged as a bad track.

System Action: The RECOVER routine continues to the next control card. There is no need to replace data on this track.

IBC304I DATA TRACK bbbbccccchhhh ON ALT. bbbbccccchhhh

Explanation: The specified data track has been assigned this alternate.

IBC305I HA IS BAD or xxx HAS BAD {KEY|DATA|KEY AND DATA} or

xxx HAS BAD {COUNT|ADDRESS MARK} LIST OF BAD RECORD FOLLOWS. ASSUME RECORD SIZE REST OF TRACK

Explanation: An error was encountered while reading the home address or the key and/or data fields of the specified record (xxx); or the specified record has a bad count or a missing address mark.

System Action: In the last case, the remaining portion of track is listed (in hexadecimal) on the message output device.

IBC306I ALT. TRACK bbbbccccchhhh ORIGINAL bbbbccccchhhh

Explanation: The REPLACE routine has assigned the specified alternate track to receive the data from the original defective track.

System Action: The data recovered from the bad track is merged with the replacement data and written on the alternate. The original track is flagged as defective.

IBC307I VTOC ON BAD TRACK, POSSIBLY CANNOT ASSIGN ALTERNATE

Explanation: The bad track contains the VTOC DSCB which has the alternate track information.

System Action: The REPLACE program assigns an alternate track if the VTOC DSCB record containing the alternate track information is not defective. If the record is defective, the alternate track cannot be assigned.

IBC400A JOB TERMINATED

Explanation: Appends REPLACE routine error messages that are accompanied by termination of the job.

IBC401A DOES NOT MEET DATA CARD ID OR FORMAT REQUIREMENTS

Explanation: Data card is not in correct format.

System Action: Job terminates.

IBC402A I/D PARAMETER ON DATA CARD DOES NOT MATCH INSERT RECORD PARAMETER

System Action: Job terminates.

User Response: Either (1) supply the correct record number or (2) put the data records in the proper order, reload the program, and execute the REPLACE routine.

IBC403A KEY AND/OR DATA FIELD LENGTH DOES NOT EQUAL COUNT FIELD REQUIREMENTS

Explanation: Either too much or not enough data was supplied in the data record.

System Action: Job terminates.

User Response: Supply the correct number of replacement data bytes (including key), reload the program, and execute the REPLACE routine.

IBC404A DATA CARD HAS INVALID CHARACTER

Explanation: An invalid hexadecimal character was found in the data record.

System Action: Job terminates.

IBC407A PREVIOUS RECOVER PROGRAM WAS ABORTED. RERUN RECOVER

Explanation: The RECOVER routine did not go to completion.

System Action: Job terminates.

IBC408A RECORD NUMBER SUPPLIED DOES NOT EQUAL CORRESPONDING BAD RECORD NUMBER

Explanation: Replacement records are not in proper order.

System Action: Job terminates.

User Response: Reorder replacement records, reload the program, and execute the REPLACE routine.

IBC409A NEW COUNT DOES NOT EQUAL GOOD ORIGINAL COUNT

Explanation: The count field for this record does not match the count field supplied on the INSERT statement.

System Action: Job terminates.

IBC411A {TRACK|VOLID} NUMBER SUPPLIED DOES NOT MATCH NUMBER ON TAPE

Explanation: The track number or volume serial number on the REPLACE statement does not match the corresponding number on the recover tape.

System Action: Job terminates.

IBC412A RECOVER TAPE WAS NOT MADE BY THIS LEVEL OF RECOVER/REPLACE. RERUN RECOVER

Explanation: The REPLACE routine attempted to use output of an earlier version of the RECOVER routine.

System Action: Job terminates.

User Response: Use an up-to-date version of the RECOVER routine to produce a correct recover tape.

IBC413A TAPE ON cuu NOT A RECOVER TAPE

Explanation: A recover tape is not mounted on the specified channel and unit, as stated in the REPLACE statement.

System Action: Job terminates.

Data Set Utility Messages

The IEBEDIT Program

IEB001I xxxxxxxx NOT OPENED

Explanation: The named data set(s) (SYSUT1, SYSUT2, and/or SYSIN) could not be opened. Either the DD statement(s) defining the data set(s) was not included in the input stream or a DCB parameter was found invalid.

System Action: The job step is terminated. (The return code is 8.)

IEB008I INVALID NAME FIELD

Explanation: An incorrectly coded name (for example, a 9-character name) was found in an EDIT statement.

System Action: processing continues with the next EDIT statement, if any. (The return code is 4.)

IEB009I INVALID STATEMENT SYNTAX

Explanation: An EDIT statement is coded incorrectly.

System Action: Processing continues with the next EDIT statement, if any. (The return code is 4.)

IEB010I INVALID OPERATION CODE

Explanation: An operator other than the four-character operator EDIT is specified in a utility control statement.

System Action: Processing continues with the next EDIT statement, if any. (The return code is 4.)

IEB011I INVALID OPERAND

Explanation: An operand other than a valid operand is included in an EDIT statement. The following operands are examples of invalid operands:

TYPE=POSTION -- wrong spelling.
TYPE=(POSITION,INCLUDE) -- incompatible subparameters.

System Action: Processing continues with the next EDIT statement, if any. (The return code is 4.)

IEB014I INVALID DELIMITER

Explanation: An invalid delimiter is included in an EDIT statement. The following EDIT statement contains an invalid delimiter:

```
EDIT START=JOBB.TYPE=INCLUDE, C
     STEPNAME=STEPC
            --There should be a
              comma, rather than a
              period, between JOBB
              and Type.
```

System Action: Processing continues with the next EDIT statement, if any. (The return code is 4.)

IEB019I REQUIRED OPERAND OMITTED

Explanation: This message is reserved for future use.

IEB020I INVALID CONTINUATION CARD

Explanation: A continuation card begins in a column other than column 16.

System Action: Processing continues with the next EDIT statement, if any. (The return code is 4.)

IEB021I INVALID CHARACTER

Explanation: An EDIT statement contains an invalid character.

System Action: Processing continues with the next EDIT statement, if any. (The return code is 4.)

IEB022I JOB NAME NOT FOUND BEFORE END OF FILE

Explanation: Either no JOB statement was found in the input data set or the specified job could not be found.

System Action: The job step is terminated. (The return code is 4.)

IEB023I stepname STEP COULD NOT BE FOUND

Explanation: The indicated stepname could not be found in the input data set.

System Action: Processing continues with the next EDIT statement, if any. (The return code is 4.)

IEB024I {0 / 4 / 8} WAS HIGHEST SEVERITY CODE

Explanation: This message, which is issued at the completion of the IEBEDIT job step, indicates the highest return code generated during the execution of the program.

IEB027I I/O ERRORxxxxx, jobname, stepname, unit address, device type, ddname, operation attempted, error description, last seek address or block count, access method.

Explanation: An input-output error occurred while processing the named data set (SYSUT1, SYSUT2, or SYSIN). Error analysis information such as jobname, stepname, unit address, device type, etc., is included.

System Action: The job step is terminated. (The return code is 8.)

IEB030I xxxxxxxx BLKSIZE INVALID

Explanation: The block size of the named data set (SYSUT1 or SYSIN) is other than a multiple of 80 bytes.

System Action: The job step is terminated. (The return code is 8.)

IEB032I SYSUT2 BLKSIZE INVALID - SYSUT1 ASSUMED

Explanation: A block size other than a multiple of 80 bytes is specified for the output (SYSUT2) data set. The SYSUT1 block size attributes are assigned to the SYSUT2 data set.

System Action: Processing continues. (The return code is 4.)

IEB033I STATEMENT NOT PROCESSED EOF ON SYSUT1

Explanation: A specified EDIT statement was not processed. An end of file was encountered on the input volume.

System Action: The job step is terminated. (The return code is 4.)

IEB034I STEPNAME REQUIRED WITH TYPE={INCLUDE / EXCLUDE}

Explanation: No stepname was specified with a TYPE=INCLUDE or TYPE=EXCLUDE operation.

System Action: Processing continues with the next EDIT statement, if any. (The return code is 4.)

The IEBCOPY Program

IEB100I I/O ERROR READING MEMBER membername

Explanation: An I/O error was encountered reading the specified member. Message IEB139I gives detailed information regarding the location of the error record (always issued previous to this message.)

System Action: The next COPY control statement is sought, unless a data check in the key or data portion only occurred. In this case the error is ignored and data is copied as it came into main storage. (The return code is 4.)

User Response: Depending on the type of error, re-run the COPY operation with the data set in error allocated (1) at a different physical location on the volume, (2) on a different device, (3) on a different channel. If the error is on an input data set, it may be necessary to re-create the data set.

IEB101I I/O ERROR WRITING MEMBER DATA AT TTR=ttr
[-DURING READ BACK CHECK]

Explanation: An I/O error occurred while copying member data to the output data set. The TTR of the record in error relative to the beginning of the data set is given. [ ] part of message is only given if the error occurred during read back check.

System Action: If the error was encountered during read back check and involved a data check in key or data only, the error is ignored. Otherwise the next COPY control statement is sought. (The return code is 4.)

User Response: Depending on the type of error, re-run the COPY operation with the data set in error allocated (1) at a different physical location on the volume, (2) on a different device, (3) on a different channel.

IEB102I MEMBER membername NOT COPIED DUE TO I/O ERROR

Explanation: An I/O error on the SYSUT3 work file has made processing of the specified member impossible. If ******** replaces the membername in the above message, the error was found reading from SYSUT3 and the output directory will have to be investigated to determine which member was not copied (possibly via use of IEHLIST.)

System Action: Processing continues with the next member to be copied. (The return code is 4.)

User Response: Depending on the type of error, re-run the COPY operation with the data set in error allocated (1) at a different physical location on the volume, (2) on a different device, (3) on a different channel.

IEB103I MEMBERS membername THROUGH END OF DATA SET ARE NOT ACCESSIBLE DUE TO I/O ERROR

Explanation: Due to an I/O error while updating the output data set's directory, members starting from the named member through the end of the data set (in alphanumeric order) have become inaccessible. If ******** replaces the membername in the above message, the error occured during a readback check and the output directory will have to be investigated to determine which directory entries are invalid.

System Action: The next COPY operation is sought. (The return code is 4.)

User Response: Depending on the type of error, re-run the COPY operation with the data set in error allocated (1) at a different physical location on the volume, (2) on a different device, (3) on a different channel.

IEB104I INVALID COMMAND OR KEYWORD

Explanation: A command or keyword on the control statement just listed is misspelled or invalid for the IEBCOPY program.

System Action: The COPY operation is terminated. The next COPY control statement is sought. (The return code is 4.)

User Response: Correct the error and re-submit the job.

IEB105I PARAMETER INVALID

Explanation: A parameter on the control statement just listed is too long or contains an invalid character.

System Action: The COPY operation is terminated. The next COPY control statement is sought. (The return code is 4.)

User Response: Correct the error and re-submit the job.

IEB106I UNEQUAL PARENTHESES

Explanation: The statement just printed has an unbalanced number of parentheses.

System Action: The COPY operation is terminated. The next COPY control statement is sought. (The return code is 4.)

User Response: Correct the error and re-submit the job.

IEB107I INVALID CONTINUATION

Explanation: The control statement just listed is invalid. Parameters may have ended with a comma (which infers continuation) but the continuation column (72) was blank. An attempt may have been made to continue a statement from within a RENAME/REPLACE specification within nested parentheses and this is invalid.

System Action: The COPY operation is terminated. The next COPY control statement is sought. (The return code is 4.)

User Response: Correct the error and re-submit the job.

IEB108I MEMBER WITHOUT SELECT OR EXCLUDE

Explanation: A statement contained MEMBER= but was not associated with a SELECT or EXCLUDE command.

System Action: The COPY operation is terminated. The next COPY control statement is sought. (The return code is 4.)

User Response: Correct the error and re-submit the job.

IEB109I NO MIXING OF SELECT AND EXCLUDE MODES IN SAME COPY STEP

Explanation: A SELECT statement immediately follows an EXCLUDE statement without an INDD= statement between, or the converse - an EXCLUDE statement immediately follows a SELECT statement.

System Action: The COPY operation is terminated. The next COPY control statement is sought. (The return code is 4.)

User Response: Correct the error and re-submit the job.

IEB110I INVALID REPLACE SPECIFIED

Explanation: Parameters were not embedded within parentheses correctly or parentheses were missing from valid RENAME/REPLACE specifications.

System Action: The COPY operation is terminated. The next COPY control statement is sought. (The return code is 4.)

User Response: Correct the error and re-submit the job.

IEB111I NULL PARAMETERS

Explanation: A control statement was completely blank or blanks followed the equal sign immediately after a keyword.

System Action: The COPY operation is terminated. The next COPY control statement is sought. (The return code is 4.)

User Response: Correct the error and re-submit the job.

IEB112I CANNOT RENAME/REPLACE ON EXCLUDE

Explanation: The control statement just listed has a parameter embedded within parentheses to show RENAME/REPLACE of this member. This is invalid with an exclusive copy.

System Action: The COPY operation is terminated. The next COPY control statement is sought. (The return code is 4.)

User Response: Correct the error and re-submit the job.

IEB113I OUTDD OR INDD NOT SPECIFIED

Explanation: The commands are incomplete. An INDD=keyword must be associated with a COPY statement that has defined the output data set (OUTDD=). A SELECT or EXCLUDE statement may have been read without an INDD= preceding it.

System Action: The COPY operation is terminated. The next COPY control statement is sought. (The return code is 4.)

User Response: Correct the error and re-submit the job.

IEB114I OUTDD/LIST NOT ON COPY CARD

Explanation: The OUTDD= or LIST= keywords were scanned but were not physically or logically associated with the COPY statement.

System Action: The COPY operation is terminated. The next COPY control statement is sought. (The return code is 4.)

User Response: Correct the error and re-submit the job.

IEB115I END OF FILE ON SYSIN

Explanation: On the first read or during a "flush" end of file was given by the SYSIN device.

System Action: Control is returned to the caller; this is the end of the last COPY operation.

User Response: None.

IEB116I MIXING CONTROL STATEMENTS FROM OLD AND NEW VERSION OF IEBCOPY

Explanation: Both types of statements were contained within the same copy step or multiple COPY operations are attempted using old version IEBCOPY control statements.

System Action: If a complete set of valid statements occurred together, one COPY operation will have been done. If the statements are intermixed, no COPY will be done. The job will be terminated. (The return code is 4.)

User Response: Correct the error and re-submit the job.

IEB117I TABLES EXCEED ALLOCATED CORE

Explanation: The amount of main storage available for creation of the INDD table and SELECT/EXCLUDE Table has been exceeded.

System Action: The COPY operation is terminated. The next COPY control statement is sought. (The return code is 4.)

User Response: Multiple COPY, OUTDD= and INDD= statements can be used so the INDD table which is built for each copy step will not be so large. The number of member names in SELECT/EXCLUDE statements per copy step can also be decreased, and the number of copy steps increased.

IEB118I CONTROL STATEMENT ERROR

Explanation: The statement just listed has an invalid command, keyword or parameter. There may be multiple INDD= keywords on the same statement or old and new versions of IEBCOPY keywords are mixed.

System Action: The job step is terminated if using old IEBCOPY type of statements. Otherwise, the COPY operation is terminated and the next COPY control statement is sought. (The return code is 4.)

User Response: Correct the error and re-submit the job.

IEB119I STATEMENT SEQUENCE ERROR

Explanation: There is an error in the old version IEBCOPY control statement sequence or multiple COPY statements immediately following each other where the first COPY statement was incomplete or out of place.

System Action: If using old version IEBCOPY mode, the job step is terminated; otherwise, the next COPY control statement is sought. (The return code is 4.)

User Response: Correct the error and re-submit the job.

IEB120I ddname VALIDATION ERROR

Explanation: The name of the DD statement on which the error occured is identified in the area designated by ddname. This message is always given by the validation routine when there is an error during validation or opening of any data set. The message immediately following this message will explain the nature of the error.

System Action: None.

User Response: None.

IEB121I OPEN ERROR

Explanation: The data set defined in the preceding message could not be opened.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: Check for invalid DD statement parameters.

IEB122I DSCB COULD NOT BE OBTAINED

Explanation: There was an error return given from the OBTAIN macro used to read the DSCB for the data set defined in the preceding message.

System Action: The next copy control statement is sought. (The return code is 4.)

User Response: Check to see that a DSCB for the data set in question is available.

IEB123I DATA SET NOT PARTITIONED

Explanation: The data set identified in the preceding message does not have partitioned organization. If the data set that is not partitioned is an input or an output data set, it cannot be processed by the IEBCOPY program.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: Make sure that input and output data sets are partitioned data sets.

IEB124I INVALID LRECL

Explanation: The logical record length of the data set defined is not valid. It may be zero or the input data set LRECL may not be equal to the output data set LRECL.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: Check for unequal LRECL parameters for the input or output data sets.

IEB125I INVALID BLOCKSIZE

Explanation: The blocksize of the data set defined is not valid. The blocksize may be zero or larger than track size going to a non-track overflow data set.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: Check for an invalid blocking factor (i.e., for FB RECFM, blocksize might not be an integer multiple of LRECL).

IEB126I ddname REFERENCES AN UNMOVABLE DATA SET

Explanation: The input data set (ddname) is flagged as unmovable so it will not be compressed in place because it may contain location dependent data.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: None.

IEB127I RECFM INCOMPATIBLE

Explanation: The record format of the input data set defined is incompatible with that of the output data set (i.e., cannot copy from fixed record format to variable record format or vice versa).

System Action: The next COPY control statement is sought (The return code is 4.)

User Response: None.

IEB128I CANNOT REBLOCK TRACK OVERFLOW DATA SETS

Explanation: The input and/or the output data set have track overflow specification so reblock/deblock will not be done.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: None.

IEB129I CANNOT REBLOCK KEYED DATA SETS

Explanation: The input and/or the output data set has keyed records so reblock/deblock will not be done.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: None.

IEB130I KEY LENGTHS UNEQUAL

Explanation: The key length of the input and output data sets are not equal.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: None.

IEB131I CANNOT COMPRESS KEYED DATA SET

Explanation: A compress in place COPY operation was requested but the data set contains keyed records. IEBCOPY will not compress keyed data sets.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: None.

IEB132I INVALID RE/DE-BLOCKING

Explanation: The data set previously defined is incompatible with the output data set. For example, a variable format record may contain an LRECL that is greater than the output blocksize.

System Action: The COPY operation is terminated. The next COPY control statement is sought. (The return code is 4.)

User Response: Re-specify the output blocksize to allow this member to be properly copied, and re-submit the job.

IEB133I MINIMUM REQUESTED CORE NOT AVAILABLE

Explanation: A variable conditional GETMAIN was issued, and the return code indicates that the minimum amount of core requested was not obtainable.

System Action: The job is terminated. (The return code is 8.)

User Response: If on a PCP system, it may be necessary to deblock the blocked SYSIN/SYSPRINT data set(s). If on an MFT or MVT system, allocate a larger partition or region to the IEBCOPY program. This error may also occur if blocked SYSIN/SYSPRINT is specified and the required access method modules use core such that the minimum requested by GETMAIN cannot be made available. In this case, de-block the blocked SYSIN/SYSPRINT data set(s).

IEB134I CANNOT COMPRESS WITH SELECT OR EXCLUDE

Explanation: An input data set's DDNAME was specified which was identical to the current output data set's DDNAME, but a SELECT or EXCLUDE control statement was also specified. This is an "implied" COMPRESS, and a mixed-mode copy step is not allowed.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: If the COMPRESS is desired, do not follow the INDD statement or group which contains the duplicate DDNAME with a SELECT or EXCLUDE control statement. If the COMPRESS is not desired, remove the duplicate DDNAME from the appropriate INDD statement.

IEB135I MINIMUM I/O BUFFER NOT ALLOCATABLE

Explanation: There is not available enough unallocated core to contain two minimum size I/O buffers without overlaying required tables.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: Make more core available to the Utility Program. If a relatively large number of membernames are specified on the current SELECT or EXCLUDE control statement(s), it may be necessary to fragment this into smaller groups of membernames and more copy steps. If using an MFT or MVT system, allocate a larger partition or region to the IEBCOPY program.

IEB136I CANNOT ALLOCATE TWO TRACKS OF I/O BUFFERS FOR COMPRESS

Explanation: There is not available enough unallocated core to contain two times the device-dependent track-blocksize as specified by the results of a DEVTYPE macro. COMPRESS operations must have this much I/O buffer space for full track I/O and synchronization.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: Make more core available to the Utility Program. If several input data set DDNAME's are specified on the current INDD control statement or group, remove the DDNAME causing the COMPRESS and put it into a separate copy operation. Also it may be necessary to take actions similar to those described in message IEB133I.

IEB137I CANNOT SPECIFY DUPLICATE MEMBERNAMES FOR SELECT/EXCLUDE/RENAME - NAME = membername

Explanation: The user has specified duplicate membernames in either his EXCLUDE statement(s) or his SELECT statement(s). If in the latter, the user may have specified duplicate renamed or un-renamed "oldnames", duplicate "newnames", or a combination of these. The membername specified is the one which was duplicated.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: If duplicate names must be specified, put each duplicate in a separate copy step. It is advisable not to specify duplicate membernames at all.

IEB138I CANNOT PROCESS ALL OLD/NEW-NAMES SPECIFIED

Explanation: The core required for processing the number of oldnames/newname pairs specified is not available.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: Decrease the number of renamed members specified within any one SELECT control statement, and spread the SELECT control statements over more copy steps.

IEB139I SYNADAF message text - { DURING READ / DURING WRITE / DURING READBACK CHECK / READING FROM SYSIN / WRITING TO SYSPRINT }

Explanation: An I/O error has occurred, the SYNADAF macro issued, and this message text is generated by the SYNADAF macro.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: Depending on the type of error, re-run the COPY operation with the data set in error allocated (1) at a different physical location on the volume, (2) on a different device, (3) on a different channel. If the error is on an input data set, it may be necessary to re-create the data set.

IEB140I ddname REFERENCES A NULL INPUT DATA SET

Explanation: The ddname printed is used to reference an "empty" input data set; there are no membernames contained in the directory of this data set.

System Action: The next input data set or control statement is sought.

User Response: None.

IEB141I CANNOT RE/DE-BLOCK WITH NOTE-LIST/USER TTRN IN MEMBER membername

Explanation: The directory entry for the named member indicates the presence of a Note List and/or User TTRN's. However, the user's data set specifications indicate the requirement to re/de-block members as they are copied. These two facts are incompatible in this Utility.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: If this member is to be copied, it cannot be re/de-blocked. Either re-specify those factors which cause re/de-blocking (i.e., BLKSIZE RECFM, LRECL parameters of the appropriate DCB's referenced in JCL), or re-build the directory entry and alter the member data as needed to eliminate the Note-List/User TTRN indicators.

IEB142I CANNOT CONTINUE TO BUILD CTLTAB

Explanation: The Utility program requires more core to build required control table to process the current input data set.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: More core is required to contain the control table; if using MFT or MVT, allocate a larger partition to the Utility program. If using PCP, or if a larger partition is not available, it will be necessary to use SELECT control statements for the members to be copied, and to assure that there are at least two of these control statements, each having approximately the same number of membernames specified, and each in a separate copy step.

IEB143I ALL SELECTED MEMBERS COPIED - DID NOT USE ALL SPECIFIED INDD'S

Explanation: All specified (selected) members have been successfully copied, and the directory entries referencing these members are properly set up. It was not necessary to use all specified input data sets in order to "find" and process all selected members.

System Action: The next control statement is sought.

User Response: None.

IEB144I THERE ARE xxx UNUSED TRACKS IN OUTPUT DATA SET REFERENCED BY ddname

Explanation: This message is issued upon completion of copying all required members to the output data set whose ddname is printed. The message is given following the completion of the COPY operation. If an error has occurred, the number of tracks given in this message may be incorrect.

System Action: The next control statement is sought.

User Response: None.

IEB145I CANNOT COMPRESS TRACK OVERFLOW DATA SET

Explanation: The Utility program will not allow a compress-in-place operation to be done if the Track Overflow bit has been set in the DCB which references the "output" data set.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: None.

IEB146I CANNOT COMPRESS WITH RE/DE-BLOCKING

Explanation: The Utility program will not allow a compress-in-place operation to be done if the user has not specified the same data set characteristics in both the input and output DD statements which reference the data set to be compressed.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: Specify the same data set characteristics (i.e., BLKSIZE, RECFM LRECL) for both the input and output DD statements to be used while compressing. This is best done by referencing the same ddname in the relevant INDD and OUTDD control statements.

IEB147I END OF JOB - {0 / 4 / 8} WAS HIGHEST SEVERITY CODE

Explanation: This message, which is issued at the completion of the IEBCOPY job step, indicates the highest return code generated during the execution of the program.

System Action: None.

User Response: None.

IEB148I NO SPACE IN OUTPUT DIRECTORY FOR DIRECTORY ENTRIES FROM INPUT DATA SET ddname

Explanation: While building an updated output directory (to reflect members copied from the input data set referenced by "ddname" in the above message), the utility program has determined that the amount of directory space allocated to the output data set is insufficient.

System Action: If message number IEB168I does not immediately follow this message, the output data set directory (a) reflects those members copied as of the immediately preceding input data set, if any, or (b) is left as it originally was if this input data

set is the first one from which members were to have been copied. If the message IEB168I does follow, the output directory is truncated.

User Response: Execute the IEHLIST program to determine just which members are usable and referenced by the truncated output directory.

IEB149I THERE ARE xxx UNUSED DIRECTORY BLOCKS IN OUTPUT DIRECTORY

Explanation: This message is issued upon completion of copying all required members to the output data set, at the end of the current COPY operation. If an error has occurred, the number of blocks given in this message may be incorrect.

System Action: The next control statement is sought.

User Response: None.

IEB150I CANNOT SORT CTLTAB BY INPUT TTR

Explanation: There was not enough main storage available to sort the input TTRs.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: Decrease the number of members being copied in the copy step that failed. PCP: in the case of a selective copy, the members specified as selected will have to be distributed over a greater number of Select control statements and COPY steps. In the case of an exclusive copy, the Exclude can be expressed as a Select, and the members selected should once again be distributed as described above. In the case of a full copy, all the members should be specified using Select control statements. Once again the selected members should be distributed as described above. MVT/MFT: Assign the utility program to a larger partition. If this cannot be done, follow the suggestions made above for PCP system users.

IEB151I ERROR FORCES JOB TO TERMINATE

Explanation: This message is issued as the result of a previous error (as indicated by one or more preceeding error messages) which has caused further processing to be terminated.

System Action: The job is terminated.

User Response: Correct the error(s) indicated by preceeding error message(s) and re-submit that portion of the job which is not successfully completed.

IEB152I membername COMPRESSED-WAS ALREADY IN PLACE AND NOT MOVED

Explanation: The member named in this message did not need to be physically moved during the compress-in-place operation.

System Action: None.

User Response: None.

IEB153I ALL MEMBERS COMPRESSED-ALL WERE ORIGINALLY COMPRESSED

Explanation: The data set which was to have been compressed in place was not in need of being compressed, since there were no embedded "gaps" between any of the members of the data set. No members from this data set were physically moved.

System Action: None.

User Response: None.

IEB154I membername HAS BEEN SUCCESSFULLY COPIED

Explanation: The member named in this message has been successfully copied from the input data set to the output data set. If the job step completes successfully, this copied member can be accessed and used.

System Action: None.

User Response: None.

IEB155I membername HAS BEEN SUCCESSFULLY COPIED AND IS A 'NEWNAME'

Explanation: The member named in this message is a renamed member which has been successfully copied from the input data set to the output data set. The "oldname" of this member can be determined by checking the Utility control statement(s) printed at the beginning of the copy step in which this message occurred. If the job step completes successfully, this copied member can be accessed (using the new membername specified), and used.

System Action: None.

User Response: None.

IEB156I NOT A DIRECT ACCESS DATA SET

Explanation: The data set defined in the previous message is not on a direct access device. IEBCOPY will not copy non-direct access data sets.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: Correct the error and re-submit the job.

IEB157I DD STATEMENT NOT FOUND

Explanation: The DD statement for the data set defined in the previous message could not be found.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: Insert a DD statement for the data set and re-run.

IEB158I PARM EQUAL COMPRESS NOT VALID

Explanation: PARM=COMPRESS was specified on the EXEC statement but the user has specified new version IEBCOPY statements which do not use PARM=COMPRESS to designate compress mode.

System Action: Processing continues, but the compress in place will not be done unless ddnames referenced in subsequent COPY operations cause it.

User Response: None.

IEB159I NO MEMBERS COPIED FROM INPUT DATA SET REFERENCED BY ddname

Explanation: The input data set whose ddname appears in this message was not used for one of the following reasons:

1. A selective copy was specified but none of the members to be copied were on this data set.
2. All of the members which were to have been copied from this input data set had names which were duplicates of membernames on the output data set, and the Replace option was not specifed.
3. An I/O error (the message for which would have been printed prior to this time) has precluded use of members from this input data set.

System Action: Normally, the next input data set will be processed. If an I/O error has occurred, the action indicated by the previous I/O error message(s) will be taken.

User Response: None if this condition was desired; otherwise, take appropriate action, depending upon the condition indicated in the above explanation.

IEB160I CONCATENATED DATA SETS

Explanation: The DD name given in the previous message is the first in a group of concatenated data sets. IEBCOPY will not process concatenated data sets.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: If more than one input data set is to be used in the copy step, a separate DD card is required for each. The ddnames must then also be specified within the INDD= keyword on a COPY or INDD utility control card.

IEB161I COMPRESS TO BE DONE USING INDD NAMED ddname

Explanation: A request for a compress in place operation has been detected. The input and output data sets are the same data set.

System Action: A compress in place operation is attempted.

User Response: None.

IEB162I INPUT DATA SET FROM INDD NAMED ddname NOT SAME AS OUTDD-CANNOT COMPRESS

Explanation: PARM=COMPRESS is specified but the input and output data sets are not the same data set.

System Action: PARM=COMPRESS is ignored.

User Response: If a COMPRESS is wanted, correct DD cards or IEBCOPY control cards and re-run.

IEB163I NO MEMBER NAMES FOR PARTIAL COPY, WILL NOT COPY

Explanation: The old version of IEBCOPY statement specified TYPCOPY=I but was not followed by any MEMBER=statements.

System Action: The job step is terminated. (The return code is 4.)

User Response: Correct the error and re-submit the job.

IEB164I TOTAL COPY ASSUMED

Explanation: The old version of IEBCOPY statement specified TYPCOPY=E but was not followed by any MEMBER=statements.

System Action: A full copy is done. (The return code is 4.)

User Response: Correct the error and re-submit the job.

IEB165I membername 'FOUND' BUT NOT COPIED, DUE TO I/O ERROR READING INPUT DIRECTORY

Explanation: A Selective copy operation was being attempted, and the member named in this message had been encountered on the current input data set prior to the occurrence of the described I/O error.

System Action: None.

User Response: None.

IEB166I NO MEMBERS COPIED TO DATA SET REFERENCED BY ddname

Explanation: Due to a validation error described in a previous message, no copying was done to the output data set referenced by the specified ddname.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: None.

IEB167I FOLLOWING MEMBER(S) COPIED FROM INPUT DATA SET REFERENCED BY ddname -

Explanation: The ddname given in this message references the input data set from which member(s) whose names will be listed were copied. This message is to assist the user in tracing which data sets were used and how they were used.

System Action: None.

User Response: None.

IEB168I **WARNING** DUE TO ERROR, POSSIBLE LOSS OF ACCESS TO MEMBER DATA AND/OR INCOMPLETE DIRECTORY

Explanation: If preceded by message IEB148I, the output directory has been truncated. Otherwise the output directory may be incomplete.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: Depending on the type of error, re-run the COPY operation with the data set in error allocated (1) at a different physical location on the volume, (2) on a different device, (3) on a different channel. If the error is on an input data set, it may be necessary to re-create the data set. Another utility (such as IEHLIST) should be used to determine the final status of the output directory.

IEB169I **WARNING** DUE TO I/O ERROR ON SYSUT4, OUTPUT DIRECTORY MAY BE INCOMPLETE

Explanation: Due to an I/O error on SYSUT4, the output directory may not be complete.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: Depending on the type of error, re-run the COPY operation with the data set in error allocated (1) at a different physical location on the volume, (2) on a different device, (3) on a different channel. The output data set's directory should be investigated to see if all information is valid (possibly via use of IEHLIST.)

IEB170I **WARNING** DUE TO SYSUT3 I/O ERROR, COMPRESS-IN-PLACE NOT DONE AND COPY OPERATION TERMINATED

Explanation: An I/O error has occurred while using the "spill" data set. None of the members will have been physically moved, so the data set remains as it was prior to processing.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: Depending on the type of error, re-run the COPY operation with the data set in error allocated (1) at a different physical location on the volume, (2) on a different device, (3) on a different channel.

IEB171I ** WARNING ** DIRECTORY MAY NOT REFLECT VALID LOCATION OF MEMBER DATA

Explanation: An I/O error during a compress-in-place operation may have affected the validity of the data sets directory.

System Action: The next COPY control statement is sought. (The return code is 4.)

User Response: The data set in question should be re-created or dumped and checked for valid information. (Possibly via IEHLIST and/or IEHDASDR.)

IEB172I ddname COULD NOT BE OPENED

Explanation: The specified data set could not be opened. This is normally the SYSPRINT data set.

System Action: This data set will not be used. I/O error messages and an end-of-job message will be issued to the console typewriter via alternate methods. The error is ignored. (The return code is 4.)

User Response: It will be necessary to use another utility (such as IEHLIST) to verify the ending status of all COPY operations performed.

IEB173I ddname - INVALID BLOCKSIZE

Explanation: An invalid blocksize associated with the specified data set was detected. This is normally the SYSPRINT data set.

System Action: This data set will not be used. I/O error messages and an end-of-job message will be issued to the console typewriter via alternate methods. The error is ignored. (The return code is 4.)

User Response: It will be necessary to use another utility (such as IEHLIST) to verify the ending status of all COPY operations performed.

IEB174I ** WARNING ** INPUT RECORD IS A SHORT LENGTH RECORD - DDNAME = inddname - OUTPUT TTRN = tt tt rr nn

Explanation: An unexpected short length record (shorter than BLKSIZE) has been found on the input data set described by inddname. It was copied to the output data set (at tt tt rr nn) exactly as it was read from the input data set.

System Action: The error is ignored.

User Response: If the error cannot be ignored by the user, the input data set will have to be re-created.

IEB175I ** WARNING ** INPUT RECORD IS GREATER THAN OUTPUT BLKSIZE- DDNAME = inddname - OUTPUT TTRN = tt tt rr nn

Explanation: An input record whose length is greater than the output block size has been processed. The record was copied to the output data set (at tt tt rr nn) exactly as it was on input (no truncation).

System Action: The error is ignored.

User Response: If necessary, re-execute the COPY operation again, specifying a larger block-size on the output data set via JCL.

IEB176I MEMBER membername IN DATASET REFERENCED BY ddname HAS MORE THAN ONE NOTELIST POINTER

Explanation: The directory entry for the referenced member has more than one note list (user TTRN with N having a value greater than zero). This is an invalid directory entry and the member cannot be correctly processed.

System Action: The next COPY operation is sought. (The return code is 4.)

User Response: Re-create the member in error.

IEB177I membername WAS SELECTED BUT NOT FOUND IN ANY INPUT DATA SET

Explanation: The member named in this message was specified on a SELECT statement for the previous copy step/operation but did not exist on any of the specified input data sets.

System Action: None

User Response: None

The IEBCOMPR Program

IEB201I INVALID CONTROL STATEMENT.

Explanation: The construction of the above control statement is invalid.

System Action: The job is terminated. (The return code is 12.)

IEB202I INVALID DIRECTORY BLOCKSIZE

Explanation: The length of the partitioned data set directory entry is less than 14 or greater than 256 bytes.

System Action: The job step is terminated. (The return code is 12.)

IEB205I USER DATA FIELDS UNEQUAL.

Explanation: The user data fields of the SYSUT1 and SYSUT2 data sets are unequal.

System Action: The fields are listed and the comparison continues. (The return code is 8.)

IEB210I TRUE NAMES MISSING FROM BOTH SETS.

Explanation: All the names in one directory must have counterpart entries in the other. This condition was not met.

System Action: The job is terminated. (The return code is 12.)

IEB211I KEY LENGTHS ARE NOT EQUAL.

Explanation: The SYSUT1 and SYSUT2 keys are of different lengths.

System Action: The job is terminated. (The return code is 12.)

IEB212I INVALID RECORD FORMAT.

Explanation: The record formats are not standard.

System Action: The job is terminated. (The return code is 12.)

IEB213I REPETITIOUS CARD INVALID

Explanation: Either a COMPARE or LABEL command has appeared twice.

System Action: The job is terminated at the end of the control card scan. (The return code is 12.)

IEB214I FIXED RECORD LENGTHS UNEQUAL.

Explanation: SYSUT1 contains records of a different length than those in SYSUT2.

System Action: The job is terminated. (The return code is 12.)

IEB215I RECORD FORMATS DIFFERENT.

Explanation: The record characteristics of the SYSUT1 and SYSUT2 data sets differ.

System Action: The job is terminated. (The return code is 12.)

IEB216I ILLEGAL CONTROL CARD SEQUENCE

Explanation: The COMPARE command did not precede all other utility statements, or two COMPARE cards appeared in the input stream.

System Action: The job is terminated at the end of the control card scan. (The return code is 12.)

IEB217I INVALID LRECL FOR V/VS RECORD

Explanation: The field of The ll V/VS record is less than 5 or greater than 32,756.

System Action: The job step is terminated. (The return code is 12.)

IEB218I PERMANENT INPUT ERROR - FIND MACRO

Explanation: A permanent input error was found by the FIND macro during a partitioned data set directory search.

System Action: The job step is terminated. (The return code is 12.)

IEB219I INVALID BLKSIZE FOR V/VS RECORD

Explanation: the LL field of the V/VS record is less than 9 or greater than 32,760.

System Action: The job step is terminated. (The return code is 12.)

IEB221I RECORDS ARE NOT EQUAL.

Explanation: Two corresponding records do not contain the same data.

System Action: The records are listed and the comparison continues. (The return code is 8.)

IEB222I KEYS ARE NOT EQUAL.

Explanation: Two corresponding keys do not contain the same data.

System Action: The records are listed and the comparison continues. (The return code is 8.)

IEB223I EXTRA RECORD ON SYSUT2.

Explanation: The SYSUT2 data set contains more records than the SYSUT1 data set.

System Action: The records are printed. (The return code is 8.)

IEB224I EXTRA RECORD ON SYSUT1.

Explanation: The SYSUT1 data set contains more records than the SYSUT2 data set.

System Action: The extra records are printed. (The return code is 8.)

IEB225I JOB TERMINATED AFTER EXIT.

Explanation: The return code from an exit routine indicates termination.

System Action: The job is terminated. (The return code is 12 or 16. It is determined by exit routine.)

IEB226I WARNING - INVALID NAME

Explanation: The statement label either contains an invalid character or is longer than eight characters.

System Action: Processing will continue normally.

IEB227I TEN CONSECUTIVE ERRORS.

Explanation: An error routine is not specified and ten successive unequal comparisons have occurred.

System Action: If the input data sets are sequential, the job is terminated. If the input data sets are partitioned, processing will continue with the next member. If the current member is the last, the job is terminated. (The return code is 12 for sequential data sets and 8 for partitioned data sets.)

IEB229I DDNAME xxx CANNOT BE OPENED

Explanation: The named DD statement does not exist.

System Action: The job is terminated. (The return code is 12.)

User Response: Either correct the ddname if it is misspelled in the DD statement or the ddlist, or insert a new DD statement with this name.

IEB230I SYSIN BLOCKSIZE ERROR

Explanation: The SYSIN DD statement specifies a block size that is not a multiple of the specified logical record length.

System Action: The job is terminated. (The return code is 12.)

IEB231I EXTRA USER INPUT HEADER LABELS ON SYSUT1

Explanation: The SYSUT1 data set contains more user input header labels than the SYSUT2 data set.

System Action: The extra labels are printed. (The return code is 8.)

IEB232I EXTRA USER INPUT HEADER LABELS ON SYSUT2

Explanation: The SYSUT2 data set contains more user input header labels than the SYSUT1 data set.

System Action: The extra labels are printed. (The return code is 8.)

IEB233I EXTRA USER INPUT TRAILER LABELS ON SYSUT1

Explanation: The SYSUT1 data set contains more user input trailer labels than the SYSUT2 data set.

System Action: The extra labels are printed. (The return code is 8.)

IEB234I EXTRA USER INPUT TRAILER LABELS ON SYSUT2

Explanation: The SYSUT2 data set contains more user input trailer labels than the SYSUT1 data set.

System Action: The extra labels are printed. (The return code is 8.)

IEB235I SYSUT1 CONTAINS NO USER INPUT HEADER LABELS

Explanation: User requested INHDR exit and/or label comparison, yet there is no input header label on SYSUT1.

System Action: Consequently, message IEB232I will be issued and the system action pertaining to the message will be taken.

IEB236I SYSUT2 CONTAINS NO USER INPUT HEADER LABELS

Explanation: User requested INHDR exit and/or label comparison, yet there is no input header label on SYSUT2.

System Action: Consequently message IEB231I will be issued and the system action pertaining to the message will be taken.

IEB237I BOTH INPUT DATA SETS CONTAIN NO USER HEADER LABELS

Explanation: User requested INHDR exit and/or label comparison, yet there are no input header labels on SYSUT1 and SYSUT2.

System Action: The processing continues. (The return code is 8.)

IEB238I SYSUT1 CONTAINS NO USER INPUT TRAILER LABELS

Explanation: User requested INTLR exit and/or label comparison, yet there are no input trailer labels on SYSUT1.

System Action: Consequently message IEB234I will be issued and the system action pertaining to the message will be taken.

IEB239I SYSUT2 CONTAINS NO USER INPUT TRAILER LABELS

Explanation: User requested INTLR exit and/or label comparison, yet there are no input trailer labels on SYSUT2.

System Action: Consequently message IEB233I will be issued and the system action pertaining to the message will be taken.

IEB240I BOTH INPUT DATA SETS CONTAIN NO USER TRAILER LABELS

Explanation: User requested INTLR exit and/or label comparison, yet there are no input trailer labels on SYSUT1 and SYSUT2.

System Action: Processing continues. (The return code is 8.)

IEB241I INPUT HEADER LABELS ARE NOT EQUAL

Explanation: Corresponding input header labels are not equal.

System Action: Print unequal label from SYSUT1 then SYSUT2. (The return code is 8.)

IEB242I INPUT TRAILER LABELS ARE NOT EQUAL

Explanation: Corresponding input trailer labels are not equal.

System Action: Print unequal label from SYSUT1 then SYSUT2. (The return code is 8.)

IEB243I I/O ERROR WHILE READING USER INPUT HEADER LABEL ON SYSUT1

Explanation: Uncorrectable I/O error occurred while reading user input header labels on SYSUT1.

System Action: Job is terminated. (The return code is 12.)

IEB244I I/O ERROR WHILE READING USER INPUT HEADER LABEL ON SYSUT2

Explanation: Uncorrectable I/O error occurred while reading user input header label on SYSUT2.

System Action: Job is terminated. (The return code is 12.)

IEB245I I/O ERROR WHILE READING USER INPUT TRAILER LABEL ON SYSUT1

Explanation: Uncorrectable I/O error occurred while reading user input trailer label on SYSUT1.

System Action: Job is terminated. (The return code is 12.)

IEB246I I/O ERROR WHILE READING USER INPUT TRAILER LABEL ON SYSUT2

Explanation: Uncorrectable I/O error occurred while reading user input trailer label on SYSUT2.

System Action: Job is terminated. (The return code is 12.)

IEB247I (no.) INPUT (header/trailer) LABELS FROM BOTH DATA SETS ARE COMPARED

Explanation: Inform the user how many user input header trailer labels are compared due to his request.

System Action: Proceed normally if return code from user exit routine is not 16. Otherwise, message IEB225I will be issued and its system action will be taken.

IEB248I (no.) EXITS TO (user label processing routine name) IS MADE FOR (SYSUT1/SYSUT2) RETURN CODE FROM USER ROUTINE IS (no.)

Explanation: User returned a return code other than 4 to indicate no more labels will be processed.

System Action: Proceed normally if return code from user exit routine is not 16. Otherwise message IEB225I will be issued and its system action will be taken.

IEB249I NO RECORDS ARE COMPARED, DATA=ONLY

Explanation: User specified DATA=ONLY, hence only user header labels are processed.

System Action: Job is terminated. (The return code is 0.)

IEB250I USER LABEL IS NOT SUPPORTED BY PARTITIONED DATA SET

Explanation: User specify INHDR/INTLR exit while comparing PO data sets.

System Action: Job is terminated. (The return code is 12.)

IEB251I INCOMPATIBLE MAXIMUM LOGICAL RECORD LENGTH

Explanation: One input data set contains logical record greater than 32K, yet the other one does not.

System Action: Job is terminated. (The return code is 12.)

IEB252I KEYED DATA SETS. ONE CONTAINS SPANNED RECORD, THE OTHER ONE DOES NOT

Explanation: Both input data sets contain keyed record. But one data set has variable spanned record, the other one does not.

System Action: Job is terminated. (The return code is 12.)

IEB253I RECORDS ARE COMPARED AT PHYSICAL BLOCK LEVEL

Explanation: When both data sets contain keyed spanned records or logical record greater than 32K, comparison is made at block level.

System Action: Proceed normally.

IEB254I CORRESPONDING BLOCK LENGTHS ARE NOT EQUAL

Explanation: Corresponding big LL are not equal when comparison is made at block level.

System Action: The unequal blocks are printed. (The return code is 8.)

IEB255I CORRESPONDING BLOCK LENGTHS ARE NOT EQUAL

Explanation: Variable/variable spanned records are of different length.

System Action: The unequal records are printed. (The return code is 8.)

IEB256I IEBCOMPR DOES NOT COMPARE PARTITIONED DATA SETS WITH VS RECFM

Explanation: User wishes to compare partitioned data sets containing variable spanned records, but the IEBCOMPR program will not support this function.

System Action: Job terminated. (Return code is 12.)

IEB257I JOB TERMINATED AFTER EXIT FOR USER VOLUME SWITCH LABEL PROCESS

Explanation: User wishes to terminate processing after examining his volume switch input header/trailer labels in his labels exit routine.

System Action: Job is terminated (return code is 16).

IEB258I USER LABELS NOT COMPARED, UNABLE TO TAKE EXIT FOR ONE DATA SET

Explanation: User wishes to process his input header/trailer labels as data, but utility is unable to take user input header/trailer label exit for one of the input data sets; probably SUL has been omitted from the SYSUT1 or SYSUT2 DD statement.

System Action: The job is terminated. (The return code is 12.)

User Response: Ensure that both SYSUT1 and SYSUT2 DD statements specify SUL in LABEL parameter.

IEB259I INVALID KEYWORD IN OR BEFORE COLUMN xx

Explanation: A keyword has been misspelled or is invalid for a specific command.

System Action: Job is terminated after the control card scan. (The return code is 12.)

IEB260I MISSING COMMAND IN OR BEFORE COLUMN xx

Explanation: A command is absent from a card; or a card which has a continuation indicator contains an error. The continuation indicator is not recognized and the scan routine looks for a command on the following card.

System Action: The job is terminated at the end of the control card scan. (The return

code is 12.)

IEB261I INVALID PARAMETER IN OR BEFORE COLUMN xx

Explanation: The error is due to one or more of the following conditions:

a. A parameter is longer than eight characters.
b. A parameter is not valid for a preceding keyword.
c. A parameter is not directly preceded by an equal sign.
d. A parameter is misspelled.

System Action: The job is terminated at the end of the control card scan. (The return code is 12.)

IEB262I MISSING KEYWORD IN OR BEFORE COLUMN xx

Explanation: A blank character directly preceded the equal sign; or the command requires a keyword and none was found.

System Action: The job is terminated at the end of the control card scan. (The return code is 12.)

IEB263I MISSING PARAMETER IN OR BEFORE COLUMN xx

Explanation: An expected parameter was not found.

System Action: The job is terminated after the control card scan. (The return code is 12.)

IEB264I FIRST CONTROL CARD IS NOT COMPARE

Explanation: The command on the first control card was not COMPARE.

System Action: The job is terminated at the end of the control card scan. (The return code is 12.)

IEB265I INVALID COMMAND IN OR BEFORE COLUMN xx

Explanaiton: A command was misspelled, or there was no blank delimiter immediately before or after it.

System Action: The job is terminated at the end of the control card scan. (The return code is 12.)

IEB266I CONTINUATION CARD BEGINS IN WRONG COLUMN

Explanation: The continuation card does not begin in columns 4-16.

System Action: The job is terminated at the end of the control card scan. (The return code is 12.)

IEB267I I/O ERROR, jobname, stepname, unit address, type, ddname, operation attempted, error description, last seek address or block count, access method.

Explanation: An I/O error occurred while processing a data set. The generated message includes error analysis information provided by the SYNADAF data management macro issued in the SYNAD routine.

System Action: The job step is terminated. (The return code is 12.)

The IEBGENER Program

IEB302I INVALID PARAMETER LIST

Explanation: The parameter list supplied by the user is invalid.

System Action: The job step is terminated. (The return code is 12.)

IEB303I INVALID CONTROL STATEMENT.

Explanation: The construction of the above control statement is invalid.

System Action: The job is terminated. (The return code is 12.)

IEB304I CONTROL STATEMENT INPUT ERROR.

Explanation: A permanent error was detected while reading the SYSIN data set.

System Action: The job is terminated. (The return code is 12.)

IEB305I JOB TERMINATED AFTER LABEL EXIT.

Explanation: The return code from a label exit routine indicates termination.

System Action: The job is terminated. (The return code is 16.)

IEB306I JOB TERMINATED AFTER KEY EXIT.

Explanation: The return code from a KEY exit routine indicates termination.

System Action: The job is terminated. (The return code is 12 or 16, determined by the exit routine.)

IEB307I JOB TERMINATED AFTER DATA EXIT.

Explanation: The return code from a DATA exit routine indicates termination.

System Action: The job is terminated. (The return code is 12 or 16, determined by the exit routine.)

IEB308I PERMANENT INPUT ERROR.

Explanation: A permanent error was detected while reading the SYSUT1 data set.

System Action: The job is terminated. (The return code is 12.)

IEB309I PERMANENT OUTPUT ERROR.

Explanation: A permanent error was detected while writing the SYSUT2 data set.

System Action: The job is terminated. (The return code is 12.)

IEB310I STOW ERROR IN OUTPUT DATA SET

Explanation: A permanent error was detected while writing the directory of the SYSUT2 data set. This error could result if the SYSUT2 data set is not partitioned.

System Action: The job is terminated. (The return code is 12.)

IEB311I CONFLICTING DCB PARAMETERS

Explanation: The DCB parameters in the SYSUT2 DD statement are not compatible with those specified in the SYSUT1 DD statement (e.g., the block size specified in the SYSUT2 DD statement is not a multiple of the block size specified in the SYSUT1 DD statement, and no editing is specified).

System Action: The job step is terminated. (The return code is 12.)

IEB312I JOB TERMINATED AFTER ERROR EXIT.

Explanation: The return code from an ERROR exit routine indicated termination.

System Action: The job is terminated. (The return code is 12.)

IEB315I SPACE NOT AVAILABLE.

Explanation: The required main storage space is not available.

System Action: The job is terminated. (The return code is 12.)

IEB316I DDNAME xxx CANNOT BE OPENED

Explanation: The named DD statement does not exist.

System Action: The job is terminated. (The return code is 12.)

User Response: Either correct the ddname if it is misspelled in the DD statement or the ddlist, or insert a new DD statement with this name.

IEB317I JOB TERMINATED, NO INPUT BLKSIZE.

Explanation: The BLKSIZE parameter is omitted from the SYSUT1 DD statement.

System Action: The job is terminated. (The return code is 12.)

IEB318I JOB TERMINATED, NO OUTPUT BLKSIZE

Explanation: The BLKSIZE parameter is omitted from the SYSUT2 DD statement.

System Action: The job is terminated. (The return code is 12.)

IEB319I INVALID SYSIN BLOCKSIZE

Explanation: The SYSIN DD statement specifies a block size that is not a multiple of the specified logical record length.

System Action: The job is terminated. (The return code is 12.)

IEB320I OUTPUT DATA SET WITH KEYS IN VS/VBS PROCESSING

Explanation: The user specified keys for a VS/VBS output data set while in a 'non-straight copy' process.

System Action: Job terminated (the return code is 12).

User Response: Specify control statement in accordance with data set characteristics.

IEB321I INPUT DATA SET WITH KEYS IN VS/VBS PROCESSING

Explanation: There were keys in the input data set in one of the following situations:

- The input data set is VS or VBS, and the output data set has different attributes from the input data set.
- The input data set is not VS or VBS, but the output data set is VS or VBS.

System Action: The job is terminated. (The return code is 12.)

IEB322I JOB TERMINATED AFTER OPENING OUTPUT DATA SET UPON USER REQUEST

Explanation: The input header user label routine requires termination of the job after the output data set is opened.

System Action: Job terminated (return code is 16).

IEB323I JOB TERMINATED AFTER HEADER LABEL PROCESSING

Explanation: The user had specified the control statement: LABELS DATA=ONLY.

System Action: Only user header labels are processed. Job terminated (return code is 8).

IEB324I n TIMES TO routinename EXITROUTINE

Explanation: This message gives the number of times that a certain user label exit routine had gotten control.

System action: Processing continued (return code is 0).

IEB325I LAST RETURN CODE WAS nn

Explanation: This message usually follows IEB324I and gives the last return code that was returned by the routine mentioned in the preceding IEB324I message.

System Action: Processing continued (return code is 0).

IEB326I name LABEL GAVE I/O ERROR

Explanation: This message is given when the system indicates that there was an I/O ERROR while reading or writing a label. The

data set is not opened if the error occurs on a header label.

System Action: Job terminated (return code is 12).

IEB327I SPECIFIED KEY EXITS ARE NOT TAKEN

Explanation: The user specified key exits on a job requiring processing of a VS/VBS data set with reformatting.

System Action: Processing continued (return code 4).

IEB328I LRECL EXCEEDS 32K; STRAIGHT COPY NOT SPECIFIED

Explanation: In the utility control statement, a process different from 'straight copy' was specified, while the RECFM in either input DCB or output DCB was VS/VBS and LRECL in either DCB or both was specified as > 32756.

System Action: Job terminated (the return code is 12).

User Response: Make utility control statements and DCB information for input and output compatible.

IEB329I RECFM=VS/VBS IS NOT ALLOWED FOR PDS

Explanation: The user specified that the output data set of IEBGENER should be partitioned and should contain VS records.

System Action: Job terminated (return code is 12).

User Response: Make control statements compatible.

IEB330I TOTALING EXIT REQUESTS TERMINATION

Explanation: The user's totaling routine returned a return code of 16 to the utility.

System Action: Processing is terminated.

IEB331I PROCESSING ENDS UPON REQUEST OF TOTALING EXIT

Explanation: The user's totaling routine returned a return code of eight to the utility.

System Action: Processing is terminated, but normal "end of data" processing will be done as far as the output data set is concerned (user label processing).

IEB332I TOTALING EXIT DEACTIVATED UPON ITS OWN REQUEST

Explanation: The user's totaling routine returned a return code of zero to the utility.

System Action: Processing continues, but no further totaling exit(s) will be taken.

IEB333I RECORD LABELS=n STATEMENTS ARE REQUIRED

Explanation: The user included the statement LABELS DATA=INPUT but did not include the required statement RECORD LABELS=n.

System Action: The job is terminated. (The return code is 12.)

IEB334I NO EDITING OR CONVERSION WILL BE DONE

Explanation: This message will be given if both data sets are VS or VBS, have the same blocksize, and have the same logical record length, to remind the user that any specified editing requirements will be ignored.

System Action: Processing continues. (The return code is 0.)

IEB336I INVALID COMMAND IN COLUMN xx

Explanation: The error is due to one or more of the following conditions:

a. GENERATE is not the first control card.
b. GENERATE appears twice.
c. Any command is misspelled or has too many letters.
d. A command is not GENERATE, EXITS, MEMBER, RECORD, or LABELS.
e. LABELS appears twice.

f. There are more input LABELS than specified by RECORD LABELS=n.

System Action: The job is terminated. (The return code is 12.)

IEB337I INVALID KEYWORD IN COLUMN xx

Explanation: A keyword is either misspelled, has too many letters, or is not valid for a specific command.

System Action: The job is terminated. (The return code is 12.)

IEB338I INVALID PARAMETER IN COLUMN xx

Explanation: The error is due to one or more of the following conditions:

a. A member name contains more than eight characters.
b. The IDENT keyword for the RECORD statement is followed by more than three parameters.
c. The FIELD keyword for the RECORD statement is followed by more than four parameters.
d. Conversion parameters in the FIELD keyword for the RECORD statement are not HE, PZ or ZD.
e. Parameters for the DATA keyword on the LABELS statement are not ALL, ONLY, YES, NO, or INPUT.
f. The LABELS keyword for the RECORD statement is not followed by a number from one to eight.

System Action: The job is terminated. (The return code is 12.)

IEB339I COMMAND MISSING PRECEDING COLUMN xx

Explanation: A command is absent from a card; or a card which has a continuation indicator contains an error. The continuation inidcator is not recognized, and the scan routine looks for a command on the following card.

System Action: The job is terminated. (The return code is 12.)

IEB340I KEYWORD MISSING PRECEDING COLUMN xx

Explanation: MEMBER is not followed by NAME; or LABELS is not followed by DATA.

System Action: The job is terminated. (The return code is 12.)

IEB341I PARAMETER MISSING PRECEDING COLUMN xx

Explanation: A keyword is not followed by a parameter; or the IDENT keyword of the RECORD statement is not followed by all three parameters.

System Action: The job is terminated. (the return code is 12.)

IEB342I INVALID SPACE ALLOCATION

Explanation: Keywords for the GENERATE statement (i.e., MAXNAME=n,MAXGPS=n, MAXFLDS=n, MAXLITS=n) have been omitted or their parameters are too small. LABELS DATA=INPUT was not specified before RECORD LABELS=n.

System Action: The job is terminated. (The return code is 12.)

IEB343I ALLOWED NO. OF CARDS EXCEEDED

Explanation: Three or more LABELS cards are encountered.

System Action: The job is terminated. (The return code is 12.)

IEB344I WARNING: INVALID STATEMENT LABEL

Explanation: The statement label is greater than eight characters or contains invalid characters.

System Action: Control card analysis will continue normally.

IEB345I CONTINUATION NOT BEFORE COLUMN 17

Explanation: The expected continuation was not found before column 17.

System Action: The job is terminated. (The return code is 12.)

IEB346I MISSING PARENTHESES

Explanation: Parentheses are not closed, or an error was encountered in a parameter list before the closing parentheses.

System Action: The job is terminated. (The return code is 12.)

IEB347I DUPLICATE KEYWORD

Explanation: An EXITS keyword is given twice.

System Action: The job is terminated. (The return code is 12.)

IEB348I PRECEDING MEMBER REQUIRES 'IDENT'

Explanation: Two MEMBER NAME=(name ,alias ...) cards are encountered without a RECORD IDENT=ident parameters being associated with the first one.

System Action: The job is terminated. (The return code is 12.)

IEB349I INCONSISTENT PARAMETERS IN FIELD OR IDENT

Explanation: The first and second parameters on an IDENT or FIELD keyword are not consistent.

System Action: The job is terminated. (The return code is 12.)

IEB350I LITERAL LENGTH EXCEEDS 40

Explanation: The literal in a Field A keyword for a RECORD statement exceeds 40 characters.

System Action: The job is terminated. (The return code is 12.)

IEB351I I/O ERROR jobname, stepname, unit address, device type, ddname, operation attempted, error description, last seek address or block count, access method.

Explanation: An I/O error occurred while processing a data set. The generated message includes error analysis information provided by the SYNADAF data management macro issued in the SYNAD routine.

System Action: The job step is terminated. (The return code is 12.)

The IEBPTPCH Program

IEB401I PRINT/PUNCH STATEMENT NOT FIRST.

Explanation: A PRINT or PUNCH statement is not the first utility control statement.

System Action: The job is terminated. (The return code is 12.)

IEB402I INVALID OPERATION

Explanation: The operation in the above utility statement is invalid.

System Action: The job is terminated. (The return code is 12.)

IEB403I MORE THAN TWO TITLE STATEMENTS.

Explanation: More than two TITLE statements are included.

System Action: The job is terminated. (The return code is 12.)

IEB404I KEYWORD INVALID OR OMITTED.

Explanation: A required keyword is either incorrectly written or not included in the above statement.

System Action: The job is terminated. (The return code is 12.)

IEB405I PARAMETER INVALID OR OMITTED.

Explanation: A required parameter is either incorrect, inconsistent, or not included in the above statement.

System Action: The job is terminated. (The return code is 12.)

IEB406I JOB TERMINATED AFTER USER EXIT.

Explanation: The job was terminated after control was returned from an exit routine.

System Action: The job is terminated. (The return code is 12 or 16, determined by exit routine.)

IEB407I JOB TERMINATED DUE TO I/O ERROR.

Explanation: A permanent error was encountered.

System Action: The job is terminated. (The return code is 12.)

User Response: Check for an incorrect BLKSIZE specification in the DCB parameter of the SYSIN DD statement.

IEB408I MEMBER xxx CANNOT BE FOUND

Explanation: The specified member is not contained in the SYSUT1 data set.

System Action: The member was not printed/punched. (The return code is 8.)

IEB409I INVALID CONTROL STATEMENT

Explanation: The construction of the above control statement is invalid.

System Action: The job is terminated. (The return code is 12.)

IEB410I INCORRECT RECORD STATEMENT

Explanation: The above RECORD statement is written incorrectly.

System Action: The job is terminated. (The return code is 12.)

IEB411I DDNAME xxx CANNOT BE OPENED

Explanation: The named DD statement does not exist.

System Action: The job is terminated. (The return code is 12.)

User Response: Either correct the ddname if it is misspelled in the DD statement or the ddlist, or insert a new DD statement with this name.

IEB412I DCB BLOCKSIZE NOT AVAILABLE

Explanation: No block size is specified for the data set.

System Action: The job is terminated. (The return code is 12.)

User Response: Ensure that all necessary parameters are included in the data control block.

IEB414I { MAXFLDS / MAXGPS / MAXLITS / MAXNAME } PARAMETER IS TOO SMALL

Explanation: The number of FIELD keywords, IDENT keywords, literals, or name keywords in MEMBER or RECORD statements is greater than the number specified in the MAXFLDS, MAXGPS, MAXLITS, or MAXNAME parameter, respectively.

System Action: The job is terminated. (The return code is 12.)

User Response: Specify a greater value for the named parameter.

IEB415I VS/VBS DATA PROCESSED IN BLOCKS

Explanation: The specified LRECL for the VS/VBS input data area exceeds 32756 bytes.

System action: Processing is continued on physical basis, (blocks rather than logical records are printed or punched).

IEB416I PREFORM, VS LRECL LARGER THAN 32K

Explanation: The specified LRECL for the VS/VBS input data set exceeds 32756 bytes and in the utility control statement PRINT/PUNCH, PREFORM was specified.

System Action: The job is terminated (the return code is 12).

User Response: Delete PREFORM parameter from the PRINT/PUNCH control statement, or reformat the data set.

IEB417I DATA SET EMPTY

Explanation: The data set to be printed or punched contains no data.

System Action: The print or punch operation is terminated. (The return code is 12.)

IEB418I VS/VBS NOT ALLOWED IN PDS

Explanation: Data set organization conflicts with record format. If RECFM=VS/VBS, then TYPORG must be PS.

System Action: The job is terminated (the return code is 12).

User Response: Ensure that DCB information and the TYPORG parameter in the utility statement match.

IEB419I USER RETURN CODE xx INVALID

Explanation: A required return code other than 0, 4 or 16 was returned by user.

System Action: The return code is ignored. Processing continues according to prior conditions.

User Response: None.

IEB420I SYSIN IS EMPTY

Explanation: The SYSIN data set does not contain any IEBPTPCH control statements.

System Action: The job step is terminated. (The return code is 12.)

User Response: Include either a PRINT or PUNCH statement and rerun the operation.

IEB421I I/O ERROR jobname, stepname, unit address, device type, ddname, operation attempted, error description, last seek address or block count, access method

Explanation: A permanent input/output error has occurred while processing on the named device. The generated message includes error analysis information provided by the SYNADAF data management macro issued in the SYNAD routine.

System Action: The return code is 12. Normal processing is terminated.

User Response: Correct the error condition described in the message.

IEB431I INVALID KEYWORD IN COLUMN xx

Explanation: A keyword is either written incorrectly or is not applicable to the command for which it is specified.

System Action: The job is terminated at the end of the control card scan. (The return code is 12.)

IEB432I INVALID PARAMETER IN COLUMN xx

Explanation: A parameter is either written incorrectly or is not applicable to the keyword for which it is specified.

System Action: The job is terminated at the end of the control card scan. (The return code is 12.)

IEB433I MISSING KEYWORD BEFORE COLUMN xx

Explanation: A required keyword has been omitted, or is preceded or followed by an invalid delimiter.

System Action: The job is terminated at the end of the control card scan. (The return code is 12.)

IEB434I MISSING PARAMETER BEFORE COLUMN xx

Explanation: A required parameter was omitted or it was preceded or followed by an invalid delimiter.

System Action: The job is terminated at the end of the control card scan. (The return code is 12.)

IEB435I MISSING COMMAND PRECEDING COLUMN xx

Explanation: A required command was omitted, or there was an error on the preceding card if the card in error is a continuation.

System Action: The job is terminated at the end of the control card scan. (The return code is 12.)

IEB436I INVALID COMMAND

Explanation: A command is written incorrectly or is invalid because of conditions set upon that command by keywords or parameters on previous statements.

System Action: The job is terminated at the end of the control card scan. (The return code is 12.)

IEB437I INVALID LITERAL

Explanation: There is an error in the literal field of the control card.

System Action: The job is terminated at the end of the control card scan. (The return code is 12.)

IEB438I INVALID NAME

Explanation: The statement name is either too long or has an invalid character.

System Action: Processing continues normally.

IEB439I CONTINUATION NOT STARTED IN 4-16

Explanation: Data on a continuation card does not begin in columns 4-16.

System Action: The job is terminated at the end of the control card scan. (The return code is 12.)

IEB440I MISSING PARENTHESIS

Explanation: A parenthesis was omitted or there was an error within the parentheses.

System Action: The job is terminated at the end of the control card scan. (The return code is 12.)

IEB441I MEMBER INVALID: TYPORG NOT PO

Explanation: A MEMBER statement is invalid if the organization is physical sequential (PS). TYPORG=PO must be specified on the PRINT or PUNCH utility control card.

System Action: The job is terminated at the end of the control card scan. (The return code is 12.)

The IEBUPDAT Program

IEB501I INVALID EXIT NAME. JOB TERMINATED.

Explanation: An exit routine name in the EXEC statement is invalid.

System Action: The job is terminated. (The return code is 12.)

IEB502I EXIT RETURN CODE INDICATES TERMINATION.

Explanation: The return code from an exit routine is 16.

System Action: The job is terminated.

IEB503I I/O ERROR ON SYSUT1. JOB TERMINATED.

Explanation: A permanent error was encountered while the SYSUT1 data set was being read.

System Action: The job is terminated. (The return code is 12.)

IEB504I I/O ERROR ON SYSIN. JOB TERMINATED.

Explanation: A permanent error was encountered while the SYSIN data set was being read.

System Action: The job is terminated. (The return code is 12.)

IEB505I I/O ERROR ON SYSUT2. JOB TERMINATED.

Explanation: A permanent error was encountered while the SYSUT2 data set was being written.

System Action: The job is terminated. (The return code is 12.)

IEB506I NM BLOCKSIZE IS ASSUMED 80

Explanation: No block size, or a blocksize which is not a multiple of 80, is specified in the SYSUT2 DD statement. No blocksize is available in an existing data set control block.

System Action: Processing continues. A block size of 80 bytes is assumed. (The return code is 8.)

IEB509I CURRENT TRANSACTION REJECTED.

Explanation: The transaction represented by the printed control statement and logical record statements is rejected because the control statement is written incorrectly or appears in the wrong position with respect to other control statements.

System Action: Processing continues with the next member of the library. (The return code is 4.)

IEB510I NO RECORDS WITHIN DELETE RANGE.

Explanation: No records were found within the range specified in the DELET statement.

System Action: Processing continues with the next member of the library. (The return code is 4.)

IEB511I NO RECORDS WITHIN NUMBER RANGE.

Explanation: No records were found within the range specified in the NUMBR statement.

System Action: Processing continues with the next member of the library. (The return code is 4.)

IEB512I DIRECTORY WRITE ERROR.

Explanation: A permanent error was detected while writing the directory of the SYSUT2 data set. This error could result if the SYSUT2 data set is not partitioned.

System Action: The job is terminated. (The return code is 16.)

IEB513I OUTPUT DIRECTORY FILLED.

Explanation: The directory of the SYSUT2 data set does not contain sufficient space for all the member entries.

System Action: The job is terminated. (The return code is 12.)

IEB514I MEMBER HAS NO RECORDS.

Explanation: The member identified in the printed header statement contains no records.

System Action: Processing continues with the next member of the library. (The return code is 4.)

IEB515I IMPROPER INVOCATION PARAMETER.

Explanation: Either the program or the EXEC statement calling IEBUPDAT has incorrectly passed parameters.

System Action: The request is terminated. (The return code is 12.)

IEB516I MEMBER NAME SEQUENCE ERROR.

Explanation: Member names, specified on header statements, are not in binary collating sequence.

System Action: Processing continues with the next member of the library. (The return code is 4.)

IEB517I DDNAME xxx CANNOT BE OPENED

Explanation: The named DD statement does not exist.

System Action: The job is terminated. (The return code is 12.)

User Response: Either correct the ddname if it is misspelled in the DD statement or the ddlist, or insert a new DD statement with this name.

The IEBISAM Program

IEB600I UTILITY PROGRAM IEBISAM HAS SUCCESSFULLY COMPLETED THE REQUESTED OPERATION COMPLETION CODE=00

Explanation: This is an informational message indicating that the program has successfully completed an operation.

System Action: Program operation has completed. (The return code is 0.)

IEB601I DCB FIELD VALUES INCONSISTENT COMPLETION CODE=08

Explanation: One or more of the following DCB subparameters contain erroneous values: RECFM, LRECL, BLKSIZE, RKP, and KEYLEN.

System Action: The requested operation is not performed; the program is terminated. (The return code is 08.)

User Response: Correct the value or values in error and rerun the job step.

IEB602I jobname, stepname, unit address, device type, ddname, operation attempted, error description, last seek address or block count, access method.
COMPLETION CODE=08

Explanation: An input/output error occurred while processing the named data set. Error analysis information such as job name, stepname, unit address, device type, etc., is included.

System Action: The job step is terminated. (The return code is 08.)

IEB603I DUPLICATE RECORD (LOAD operation only) COMPLETION CODE=08

Explanation: A record key is identical to a record key previously placed in the indexed sequential data set.

System Action: The program is terminated. Reconstruction of the indexed sequential data set is incomplete. (The return code is 08.)

User Response: Determine the cause of the error, UNLOAD the original indexed sequential data set, and respecify the LOAD operation.

IEB604I NUMBER OF CHARACTERS TO BE TRANSMITTED EXCEEDS LIMIT (LOAD operation only) COMPLETION CODE=08

Explanation: The number of characters in a fixed-length record exceeds the value specified in LRECL or in LRECL + KEYLEN (for fixed-length, unblocked records with a relative key position of 0).

System Action: The program is terminated. The requested operation is not performed. (The return code is 08.)

User Response: Determine the cause of the error, UNLOAD the original indexed sequential data set, and respecify the LOAD operation.

IEB605I CLOSE REQUESTED BY USER AFTER A USER EXIT
COMPLETION CODE=04

Explanation: A user routine passed a return code of either 04 or 12 to the IEBISAM utility program.

System Action: The job step is terminated. (The return code is 04.)

IEB606I ILLEGAL RETURN CODE RECEIVED FROM A USER EXIT
COMPLETION CODE=12

Explanation: A user routine passed a return code other than 00, 04, 08, or 12 to the IEBISAM utility program.

System Action: The job step is terminated. (The return code is 12.)

IEB607I SYSUT2 or SYSUT1 DD CARD MISSING
COMPLETION CODE=16

Explanation: No SYSUT1 statement or SYSUT2 statement was included in the job step.

System Action: The job step is terminated. (The return code is 16.)

User Response: Include the missing SYSUT1 or SYSUT2 DD statement and rerun the job step.

IEB608I INVALID OPTION IN THE PARM FIELD OF THE EXECUTE CARD
COMPLETION CODE=16

Explanation: The PARM parameter of the EXEC statement is coded incorrectly.

System Action: The progran is terminated. (The return code is 16.)

IEB609I INPUT SEQUENCE ERROR

Explanation: Either a record has been lost or a noise record was encountered when loading an unloaded indexed sequential data set.

System Action: The job step is terminated. (The return code is 8.)

The IEBDG Program

IEB700I DATA GENERATION HAS BEEN [SUCCESSFULLY] COMPLETED.
COMPLETION CODE IS {ZERO / FOUR / EIGHT / TWELVE / SIXTEEN}

Explanation: When the completion code is zero, data generation has completed successfully. When the completion code is other than zero, the word 'successfully' will be omitted from the message. A return code of four indicates that the job step was terminated at the user's request. A return code of eight indicates that an error occurred while processing a utility control statement. A return code of twelve indicates that an error occurred while processing an input or output data set. A return code of sixteen indicates that incorrect parameters were found in a DCB during OPEN.

System Action: The program is ended.

IEB702I OPERATION WAS NOT DSD,FD,CREATE,REPEAT,OR END.
CORRECT AND RERUN.

Explanation: A utility control statement specifying an invalid operation was encountered.

System Action: Syntax checking of the remainder of the control statements in the set continues, but no additional data is generated. Processing continues with the next DSD statement encountered, if any. (The return code is 8.)

User Response: The output data set may have been only partially completed. Use the IEHPROGM system utility program to scratch the data set, if necessary.

IEB703I INVALID KEYWORD VALUE.
DELIMITER, DESCRIPTOR OR TYPE IS IMPROPER OR DUPLICATED. AN FD NAME HAS OCCURRED PREVIOUSLY.

Explanation: A utility control statement keyword value was found to be invalid or missing. For example:

- A double quote coded within a picture caused an invalid length.
- A starting character of * when AL or AN format is specified.
- A character other than 0-9 or A-F when a hexadecimal digit is specified.
- Any non-numeric number when a decimal number is specified; for example, 123A56.
- A Keyword was misspelled.
- An FD card contains a previously used name.

Message IEB727I shows the point at which the invalidity was uncovered.

System Action: Syntax checking of the remainder of the control statements in the set continues, but no additional data is generated,. Processing continues with the next DSD statement encountered, if any. (The return code is 8.)

User Response: The output data set may have been only partially completed. Use the IEHPROGM system utility program to scratch the data set, if necessary.

IEB704I INPUT DDNAME ON CREATE OR FD CARD IS NOT SPECIFIED ON DSD CARD.

Explanation: The ddname specified on an FD card was not referred to on the DSD statement beginning this set of utility control statements. The IEBDG program was unable to OPEN the data set.

System Action: Syntax checking of the remainder of the control statements in the set continues, but no additional data is generated. Processing continues with the next DSD statement encountered, if any. (The return code is 8.)

User Response: The output data set may have been only partially completed. Use the IEHPROGM system utility program to scratch the data set, if necessary. Include the missing ddname (indicated by message IEB727I) in the DSD statement and resubmit the job.

IEB705I INVALID KEYWORD, POSSIBLE IMBEDDED COMMA.

Explanation: A utility control statement keyword was found invalid. Message IEB727I shows the point at which the invalidity was uncovered.

System Action: Syntax checking of the remainder of the control statements in the set continues, but no additional data is generated. Processing continues with the next DSD statement encountered, if any. (The return code is 8.)

User Response: The output data set may have been only partially completed. Use the IEHPROGM system utility program to scratch the data set, if necessary.

IEB706I NUMBER SPECIFIED IS LARGER THAN 32,767 OR EXCEEDS MACHINE CAPACITY (2,147,483,647).

Explanation: An FD length parameter may have been specified incorrectly.

System Action: No conversion is performed. Syntax checking of the remainder of the control statements in this set continues, but no additional data is generated. Processing continues with the next DSD statement encountered, if any. (The return code is 8.)

User Response: The output data set may have been only partially completed. Use the IEHPROGM system utility program to scratch the data set, if necessary.

IEB707I FD NAME ON CREATE CARD IS NOT PREVIOUSLY DEFINED BY AN FD CARD

OR IS NOT ASSOCIATED WITH CORRECT DCB

Explanation: INPUT=ddname parameters on the CREATE and FD cards are not the same.

System Action: Syntax checking of the remainder of the control statements in this set continues, but no additional data is generated. Processing continues with the next DSD statement encountered, if any. (The return code is 8.)

User Response: The output data set may have been only partially completed. Use the IEHPROGM system utility program to scratch the data set, if necessary. Ensure that the FD statement precedes the CREATE statement referring to it and resubmit the job.

IEB708I PICTURE LENGTH TOO LARGE FOR CONVERSION

Explanation: A decimal picture was to be converted to packed decimal or to a binary equivalent; however, the number of specified digits in the picture exceeds 16.

System Action: No conversion is performed. Syntax checking of the remainder of the utility control statements in this set continues, but no additional data is generated. Processing continues normally with the next DSD statement encountered, if any. (The return code is 8.)

User response: The output data set may have been only partially completed. Use the IEHPROGM system utility program to scratch the data set, if necessary.

IEB709I USER EXIT ROUTINE RETURNED AN INVALID RETURN CODE

Explanation: A return code other than 0, 4, 12, or 16 was found by the IEBDG program after control was passed from an exit routine to the IEBDG program.

System Action: Syntax checking of the remainder of the utility control statements in this set continues, but no additional data is generated. Processing continues normally with the next DSD statement encountered, if any. (The return code is 8.)

User Response: The output data set may have been only partially completed. Use the IEHPROGM system utility program to scratch the data set, if necessary.

IEB710I UNABLE TO GET ENOUGH SPACE TO PROCESS REMAINING CONTROL CARDS

Explanation: A GETMAIN operation was unable to get additional required space.

System Action: Syntax checking of the remainder of the utility control statements in this set continues, but no additional data is generated. Processing continues normally with the next DSD statement encountered, if any. (The return code is 8.)

User Response: The output data set may have been only partially completed. Use the IEHPROGM system utility program to scratch the data set, if necessary. In those applications in which a REGION parameter is specified, check to ensure that the specified value is sufficient and resubmit the job.

IEB711I KEYWORD VALUE NOT FOLLOWED BY A BLANK OR COMMA

Explanation: A keyword value (indicated by message IEB727I) is not followed by a valid delimiter.

System Action: Syntax checking of the remainder of the utility control statements in this set continues, but no additional data is generated. Processing continues normally with the next DSD statement encountered, if any. (The return code is 8.)

User Response: The output data set may have been only partially completed. Use the IEHPROGM system utility program to scratch the data set, if necessary.

IEB712I CONTROL CARD NAME OR KEYWORD VALUE EXCEEDS 8 CHARACTERS

Explanation: The length of a keyword value or control card name exceeds 8 characters.

System Action: Syntax checking of the remainder of the utility control statements in this set continues, but no additional data is generated. Processing continues normally with the next DSD statement encountered, if any. (The return code is 8.)

User Response: The output data set may have been only partially completed. Use the IEHPROGM system utility program to scratch the data set, if necessary.

IEB713I FLAGGED KEYWORD IS NOT COMPATIBLE WITH A PRECEDING KEYWORD

Explanation: A keyword (indicated by message IEB727I) is not compatible with a keyword preceding it in the control statement; for example, PICTURE and FORMAT are mutually exclusive.

System Action: Syntax checking of the remainder of the utility control statements in this set continues, but no additional data is generated. Processing continues normally with the next DSD statement encountered, if any. (The return code is 8.)

User Response: The output data set may have been only partially completed. Use the IEHPROGM system utility program to scratch the data set, if necessary.

IEB714I REPEAT CARD ERROR OR REQUIRED NUMBER OF CREATE CARDS NOT PRESENT

Explanation: Either (1) two or more REPEAT statements refer to the same CREATE statement or to the same group of CREATE statements, or (2) a CREATE keyword in a REPEAT statement specifies a number greater than the number of following CREATE statements.

System Action: Syntax checking of the remainder of the utility control statements in this set continues, but no additional data is generated. Processing continues normally with the next DSD statement encountered, if any. (The return code is 8.)

User Response: The output data set may have been only partially completed. Use the IEHPROGM system utility program to scratch the data set, if necessary.

IEB715I NAME AND/OR LENGTH OR QUANTITY SPECIFICATION(S) OMITTED FROM FD AND/OR REPEAT CARD

Explanation: Either the FD field name, length specification, or quantity is missing from an FD and/or REPEAT statement.

System Action: Syntax checking of the remainder of the utility control statements in this set continues, but no additional data is generated. Processing continues normally with the next DSD statement encountered, if any. (The return code is 8.)

User Response: The output data set may have been only partially completed. Use the IEHPROGM system utility program to scratch the data set, if necessary.

IEB716I PICTURE STRING OR FD FIELD OVERFLOWS OUTPUT RECORD OR INPUT FIELD SELECTED OVERRUNS INPUT WORKAREA

Explanation: At some time in the construction of an output record by a CREATE statement, a specified picture string or FD field extends past the end of the defined record.

System Action: Syntax checking of the remainder of the utility control statements in this set continues, but no additional data is generated. Processing continues normally with the next DSD statement encountered, if any. (The return code is 8.)

User Response: The output data set may have been only partially completed. Use the IEHPROGM system utility program to scratch the data set, if necessary. Check DCB Parameters. Compare LRECL against length of defined record.

IEB717I INPUT RECORD SIZE EXCEEDS SPECIFIED OUTPUT RECORD SIZE

Explanation: The record length specified in a DD statement defining an output data set is not sufficient to contain corresponding input records.

System Action: Syntax checking of the remainder of the utility control statements in this set continues, but no additional data is generated. Processing continues normally with the next DSD statement encountered, if any. (The return code is 8.)

User Response: The output data set may have been only partially completed. Use the IEHPROGM system utility program to scratch the data set, if necessary.

IEB718I DSD CONTROL CARD MUST BE FIRST CARD OF SET

Explanation: Control card is out of order or missing.

System Action: Further syntax checking is performed, but no additional data is generated.

User Response: Place DSD card in correct order in input stream.

IEB720I BLANK DOES NOT FOLLOW OPERATION OR CONTROL CARD NAME

Explanation: A control card name or operation is not delimited by a blank.

System Action: Syntax checking of the remainder of the utility control statements in this set continues, but no additional data is generated. Processing continues normally with the next DSD statement encountered, if any. (The return code is 8.)

User Response: The output data set may have been only partially completed. Use the IEHPROGM system utility program to scratch the data set, if necessary.

IEB721I KEYWORD, KEYWORD VALUE OR DELIMITER IS MISSING OR EXTENDS INTO COLUMN 72

Explanation: A keyword, keyword value or delimiter is missing or extends into column 72.

System Action: Syntax checking of the remainder of the utility control statements in this set continues, but no additional data is generated. Processing continues normally with the next DSD statement encountered, if any. (The return code is 8.)

User Response: The output data set may have been only partially completed. Use the IEHPROGM system utility program to scratch the data set, if necessary.

IEB723I PICTURE PARAMETER IS NOT FOLLOWED BY A BLANK OR COMMA

Explanation: The picture length subparameter is not followed by a blank or comma, or picture value field is not followed by a blank or comma.

System Action: Syntax checking of the remainder of the utility control statements in this set continues, but no additional data is generated.

User response: The output data set may have been only partially completed. Use the IEHPROGM system utility program to scratch the data set if necessary.

IEB724I UNABLE TO OPEN DATA SET. LOOK FOR CONFLICTING VALUES OR MISSING DD CARD

Explanation: An input or output data set referred to by a DSD statement could not be opened.

System Action: No syntax checking or data generation is performed for this set of utility control statement. Processing continues normally with the next DSD statement encountered, if any. (The return code is 8.)

User Response: Check for a missing DD statement. If the applicable DD statement is included, check for missing or invalid BLKSIZE, LRECL, or RECFM subparameters on that statement.

IEB725I A DUPLICATE DSD CARD HAS BEEN FOUND. CHECK FOR MISSING END CARD

Explanation: An END card is missing or out of order.

System Action: Syntax checking of utility control statements continues, but no additional data is generated. Normal processing resumes with the next DSD statement encountered, if any.

User Response: Put END card in correct order.

IEB726I EXEC STATEMENT PARM PARAMETER IS CODED INCORRECTLY

Explanation: The EXEC statement PARM parameter contains an invalid character or characters or is otherwise coded incorrectly.

System Action: The line count for the message data set is set to a default value of 58. (The return code is 0.)

User Response: Check to ensure that the specified LINECNT value is a 4-digit decimal number.

IEB727I ERROR

Explanation: This message is printed to pinpoint the location of syntax errors, incompatible keywords, and other control statement coding errors. In most cases the "E" of ERROR falls directly below the point at which the error was detected in the applicable control statement.

System Action: None; that is, the system action and return code is generated with the accompanying error message, indicating the type of error.

IEB728I INPUT STREAM FLUSHED FROM THIS POINT.
LRECL, BLKSIZE, OR RECFM INCORRECT IN INPUT OR OUTPUT DCB

Explanation: An input or output data set could not be opened.

System Action: No syntax checking or data generation is performed for this set of utility control statements. Processing continues normally with the next DSD statement encountered, if any. (The return code is 8.)

User Response: Check for missing or invalid BLKSIZE, LRECL, or RECFM specifications on applicable DD statements.

IEB729I PERMANENT I/O ERROR, jobname, stepname, unit address, device type, ddname, operation attempted, error description, last seek address or blockcount, access method

Explanation: An input/output error occurred while processing a data set. Error analysis information, such as ddname, is included in the message.

System Action: The job step is terminated. (The return code is 12.)

User Response: Analyze the displayed error information and correct the data set, DD card, or DCB parameters if necessary. If the error persists, call a field engineer.

The IEBUPDTE Program

IEB801I {OM / NM} BLOCKSIZE ASSUMED 80.

Explanation: Necessary DCB parameters were omitted from the indicated DD Statement; OM (old master) indicates the SYSUT1 DD statement and NM (new master) indicates the SYSUT2 DD statement. The program assumes that the applicable data set contains 80-byte (fixed-length) unblocked records.

System Action: No immediate action is taken. However, if the data set contains records other than as indicated in the explanation, additional messages will be generated during execution and the job step terminated.

User Response: If the record format specifications are compatible with the above assumptions, ignore the message. Otherwise, specify the applicable parameters and rerun the job step.

IEB802I I/O ERROR jobname, stepname, unit address, device type, ddname, operation attemped, error description, track address or relative block number, access method.

Explanation: A permanent I/O error occurred while processing on the named device. If possible,

the generate message includes error analysis information such as jobname, stepname, unit address, device type, etc., provided by the SYNADAF data management macro issued in the SYNAD routine.

System Action: The jobstep is terminated. (The return code is 12.)

User Response: Check for missing or incorrect DCB parameters on the appropriate dd statement.

IEB803I PERMANENT INPUT ERROR - FIND MACRO

Explanation: A permanent input error was found by the FIND macro during a partitioned directory search.

System Action: The job step is terminated. (The return code is 12.)

IEB804I PERMANENT INPUT ERROR - BLDL MACRO

Explanation: A permanent input/output error was detected by the BLDL macro when attempting to search a partitioned data set directory.

System Action: The job step is terminated. (The return code is 12.)

IEB805I CONTROL STATEMENT ERROR

Explanation: The last utility control statement listed in the printout contains an incorrect name or keyword.

System Action: If both the old master data set and the updated master data set are partitioned, the program continues processing with the next function statement, if any. Otherwise, the job step is terminated. (The return code is 4.)

IEB806I STATEMENT SEQUENCE ERROR

Explanation: Either the utility control statements are out of sequence, or an unnecessary statement has been encountered.

System Action: If both the old master data set and the updated master data set are partitioned, the program continues processing with the next function statement, if any. Otherwise, the job step is terminated. (The return code is 4.)

IEB807I INVALID OPERATION

Explanation: A conflict between specifications has been uncovered. (For example, a DELETE statement is encountered during an UPDATE=INPLACE operation, or a CHANGE statement is encountered but PARM=NEW is specified on the EXEC statement, or data statements are out of sequence.)

System Action: If both the old master data set and the updated master data set are partitioned, the program continues processing with the next function statement if any. Otherwise, the job step is terminated. (The return code is 4.)

IEB808I TERMINATED THIS MEMBER. IEBUPDTE WILL TRY NEXT MEMBER.

Explanation: A control statement error, a statement sequence error, or an invalid operation has terminated an update operation.

System Action: Processing continues with the next function statement, if any.

IEB810I DELETE RANGE INVALID.

Explanation: A DELETE statement specifies a SEQ1 or SEQ2 value that does not match a sequence number in an existing logical record.

System Action: Processing continues with the next function statement, if any. (The return code is 4.)

IEB811I NUMBER RANGE INVALID.

Explanation: A NUMBER statement specifies a SEQ1 value that does not match a sequence number in an existing logical record.

System Action: Processing continues with the next function statement, if any. (The return code is 4.)

IEB812I DIRECTORY WRITE ERROR.

Explanation: A permanent input/output error was uncovered while writing the directory of the SYSUT2 data set.

System Action: The job step is terminated. (The return code is 12.)

IEB813I OUTPUT DIRECTORY FULL.

Explanation: Insufficient space was allocated for directory entries in the SYSUT2 data set. The member was not placed in the data set.

System Action: The job step is terminated. (The return code is 12.)

User Response: Re-create the SYSUT2 data set, allocating sufficient space for the additional directory entries. Then rerun the IEBUPDTE program to include the member or members that were omitted.

IEB814I DDNAME xxx CANNOT BE OPENED.

Explanation: The named data set cannot be opened.

System Action: The job step is terminated. (The return code is 12.)

User Response: Check the DD statements for proper data set definition.

IEB815I CANNOT PROCESS MORE THAN ONE PS DATA SET PER PASS.

Explanation: A control statement has specified the processing of a second input sequential data set in the same job step.

System Action: The first data set is processed and the program is terminated. (The return code is 12.)

User Response: Update the additional sequential data set in a separate job step.

IEB816I MEMBER NAME (xxx) FOUND
or alternately
IEB816I MEMBER NAME (xxx) FOUND IN NM DIRECTORY. TTR IS NOW ALTERED.

Explanation: The member name xxx already exists.

System Action: If the member is found to exist in the new master directory, the optional message is issued. If the member is found to exist, but in some other directory, the basic message is issued.

IEB817I MEMBER NAME (xxx) NOT FOUND IN NM DIRECTORY. STOWED WITH TTR.

Explanation: The member name xxx does not exist in the directory of the new master (NM) data set.

System Action: An entry (TTR) is made for the member in the directory. Processing continues.

IEB818I HIGHEST CONDITION CODE WAS xxx.

Explanation: The listed code is the highest generated in the job step.

IEB819I END OF JOB IEBUPDTE.

Explanation: The IEBUPDTE program has completed processing.

IEB820I CANNOT PROCESS MORE THAN ONE UPDATE INPLACE PER PASS.

Explanation: Two or more UPDATE=INPLACE operations were specified in the same job step.

System Action: The job step is terminated. (The return code is 12.)

IEB821I INVALID EXIT NAME. JOB ENDED.

Explanation: The name of a user exit routine is invalid.

System Action: The job step is terminated. (The return code is 12.)

User Response: Check the applicable function statement for an incorrect specification.

IEB822I EXIT RETURN CODE ENDED JOB.

Explanation: A user routine passed a return code of 16 to the IEBUPDTE program.

System Action: The job step is terminated. (The return code is 16.)

IEB823I xxx HAS NO RECORDS.

Explanation: The indicated SYSUT1 or SYSIN data set contains no records.

System Action: If the data set is the old master data set (SYSUT1), the program continues processing with the next member, if any. (The return code is 4.) If the data set is the control data set (SYSIN), the job step is terminated. (The return code is 12.)

IEB825I ALIAS IGNORED - SEQUENTIAL DATA SET.

Explanation: An ALIAS statement specified an alias name for an output sequential data set.

System Action: The statement is ignored.

IEB826I MEMBER NAME FOUND IN OM DIRECTORY AS AN ALIAS - CHANGED TO TRUE NAME IN NM DIRECTORY.

Explanation: The specified member name is an alias name in the old master (OM) directory. It is entered as a member name in the new master (NM) directory.

IEB827I INVALID INPUT PARAMETER.

Explanation: Either (1) the EXEC statement contains an incorrect parameter, or (2) an incorrect parameter was passed to the IEBUPDTE program.

System Action: The job step is terminated. (The return code is 12.)

IEB828I PAGE NUMBER PARAMETER INVALID.

Explanation: An invalid starting page number (for the message data set) was passed to the IEBUPDTE program.

System Action: A page number of 1 is assigned to the first page of the printout. (The return code is 4.)

IEB829I DDNAME PARAMETER IS INVALID.

Explanation: An incorrect DDNAME parameter was passed to the IEBUPDTE program.

System Action: The job step is terminated. (The return code is 12.)

IEB830I OLD AND NEW MASTER LRECL UNEQUAL.

Explanation: The logical record lengths of the old and new master data sets are unequal.

System Action: The job step is terminated. (The return code is 12.)

User Response: Rerun the job step. Specify the correct logical record length (LRECL) in the DCB parameter of the SYSUT2 DD statement.

IEB831I OLD AND NEW MASTER DSORGS INCOMPATIBLE

Explanation: (1) The data set organizations as implied or specified on the SYSUT1 and SYSUT2 DD statements are inconsistent with one another, or (2) the data set organization as implied or specified on the utility control statements is inconsistent with the organization implied or specified on the SYSUT1 and/or SYSUT2 DD statements.

System Action: The job step is terminated. (The return code is 12.)

User Response: In case #1, check the SYSUT1 and SYSUT2 DD statements to ensure that the space allocation is consistent with the data set organization. Ensure that the DSORG subparameter, if included, is coded correctly. In case #2, check the utility control statements to ensure that the specified keywords are consistent with the data set organization(s) specified or implied on the SYSUT1 and SYSUT2 DD statements.

IEB832I (Routine Name) IS PROCESSING USER {INPUT / OUTPUT} {HEADER / TRAILER LABELS}

Explanation: Will be printed immediately preceding each entry to OPEN, CLOSE or EOV. The user's routine name will be inserted in the message. A maximum of six similar messages can be printed.

This message will act as a header message for all subsequent user label messages pertaining to that data set, both normal and error, since it will always be the first message and will have the routine name in common with subsequent user label messages:

System Action: Processing continues. (The return code is 0.)

IEB833I (No.) ENTRANCES TO (routine name)

Explanation: Will be printed after user label processing is complete. The number of entrances to the user's routine and the routine name will be inserted into the message. This message in conjunction with the return code message (IEB834I) or the I/O Error message (IEB837I) will allow the user to determine the status of his labels.

System Action: Processing continues. (The return code is 0.)

IEB834I LAST RETURN CODE FROM (Routine Name) WAS (xxx)

Will be printed after the preceding message (IEB833I) and will contain the name of the user's routine and the last return code issued by that routine. However, if an error occurred during user label processing this message may be replaced by the I/O error message IEB802I.

System Action: Processing continues. (The return code is 0.)

IEB835I {TOTALING / USER LABELS / DATA SETS} SUPPORTED ONLY ON PS

Explanation: This message is printed prior to OPEN or CLOSE when the user has requested either user label processing or totaling exits for a data set whose organization is not physical sequential.

System Action: The job is terminated. (The return code is 4.)

IEB836I TRAILER LABEL PROCESSING NOT SUPPORTED FOR UPDATE=INPLACE

Explanation: The user specified user trailer label exits with UPDATE=INPLACE.

System Action: The job is terminated. (The return code is 4.)

IEB837I I/O ERROR WHILE PROCESSING USER LABEL

Explanation: An I/O error occurred during user label processing. The results of the label processing are unpredictable.

System Action: The job is terminated. (The return code is 12.)

User Response: Check for missing or incorrect DCB parameters for the data set.

IEB839I (Routine Name) IS TAKING TOTALING EXITS

Explanation: The user has requested totaling exits to be taken prior to writing every record.

System Action: Processing continues. The return code is zero.

IEB840I (Routine Name) REQUESTED TERMINATION OF TOTALING EXITS

Explanation: A return code other than four was passed to IEBUPDTE by a user totaling routine.

System Action: If the return code is four, totaling exits are discontinued, but processing continues. If the return code is eight or 16, processing is discontinued immediately.

IEB841I INVALID RETURN CODE FROM (Routine Name), TOTALING EXITS DISCONTINUED

Explanation: The return code passed to IEBUPDTE from the user totaling routine during a totaling exit was not valid (0, 4, 8, 16). Totaling exits are discontinued and processing continues normally.

System Action: Processing continues. (The return code is 0.)

IEB842I TOTALING EXITS NOT SUPPORTED FOR UPDATE=INPLACE.

Explanation: The user specified totaling with UPDATE=INPLACE.

System Action: The job is terminated. The return code is 4.

IEB843I INVALID CORE SIZE

Explanation: The core requested in the TOTAL keyword contains either a non-numeric character or is numerically less than 2 bytes or greater than 32K bytes.

System Action: Processing is terminated. The return code is 4.

IEB844I NO USER{HEADER|TRAILER}LABELS EXIST ON INPUT DATA SET

Explanation: The user specified SUL on the DD statement for the input data set, but there are no labels on this data set.

System Action: Processing continues. (The return code is 0.)

IEB845I NO USER{HEADER|TRAILER}LABELS CREATED ON OUTPUT DATA SET

Explanation: The user specified SUL on the DD statement for SYSUT2 but no labels were copied from the SYSUT1 data set and no labels were generated via a LABEL statement.

System Action: Processing continues. (The return code is 0.)

IEB846I ALIAS IGNORED FOR UPDATE=INPLACE

Explanation: Alias statements for partitioned data set members cannot be processed using the UPDATE=INPLACE function. All alias statements are ignored.

System Action: Processing continues. (The return code is 12.)

The IEBTCRIN Program

IEB901A M ddd,xxxxxx,jjj,sss

Explanation: M indicates that an MTDI or MTST cartridge file is to be mounted on device ddd. The volume was required by job jjj, or, if applicable, step sss of job jjj. If a volume serial number was provided in the SYSUT1 DD card, this serial number will replace xxxxxx. If a volume serial is not provided TCRINP will be used.

User Response: Mount the requested cartridges on the specified device and press the START button to ready the device. If the volume cannot be mounted, issue a CANCEL command to terminate the job named jjj.

IEB902I INVALID NAME FIELD

Explanation: The name field of the above statement contains either:

1) a non-alphabetic character in column 1, or

2) more than eight characters.

System Action: Diagnosis of this control statement is terminated. Any additional control statements are scanned for syntax errors after which the job step is terminated. (The return code is 12.)

User Response: Correct the name field and resubmit the job.

IEB903I INVALID OPERATION

Explanation: An operator other than the operators TCRGEN and EXITS was specified.

System Action: Diagnosis of this control statement is terminated. Any additional control statements are scanned for syntax errors after which the job step is terminated. (The return code is 12.)

User Response: Correct the operation field and resubmit the job.

IEB904I INVALID KEYWORD

Explanation: An invalid keyword appears in the operand of the above statement. For example, the TYPE keyword was misspelled as TAPE.

System Action: Processing continues with the next keyword, if any. Any additional control statements are scanned for syntax errors after which the job step is terminated. (The return code is 12.)

User Response: Correct the control statement and resubmit the job.

IEB905I INVALID PARAMETER VALUE

Explanation: An invalid parameter value appears in the operand of the above statement. For example:

1) The MAXLN parameter value contains more than 5 digits.

2) The REPLACE parameter is not of the form X'xx' where x is either A-F or 0-9.

3) The OUTHDR2 user routine name is more than eight characters.

4) The VOKCHK parameter was misspelled as VOKCHECK.

System Action: Processing continues with the next keyword, if any. Any additional control statements are scanned for syntax errors after which the job step is terminated. (The return code is 12.)

User Response: Correct the control statement and resubmit the job.

IEB906I DUPLICATE OPERATION FIELD

Explanation: A control statement with the same operation field as the control statement listed above has been processed previously.

System Action: Diagnosis of this control statement is terminated. Any additional control statements are scanned for syntax errors after which the job step is terminated. (The return code is 12.)

User Response: Resubmit the job using only the correct control statements.

IEB907I INCONSISTENT REPLACE PARAMETER

Explanation: The TCRGEN statement contains a REPLACE parameter which is inconsistent with specified or implied TYPE, TRANS, and/or EDIT options.

System Action: The job step is terminated (The return code is 12.)

User Response: Correct the control statement and resubmit the job.

IEB908I CONFLICTING OPTIONS SPECIFIED

Explanation: Either 1) the control statement contains two or more keyword parameters which cannot appear together, or 2) the control statement contains two or more appearances of the same keyword.

System Action: The job step is terminated. (The return code is 12.)

User Response: Correct the control statement(s) and resubmit the job.

IEB909I EXPECTED CONTINUATION NOT RECEIVED

Explanation: In a control statement, continuation of the operand was specified by a comma at the end of the operand and the following card image was not a continuation; that is, columns 1-3 were not blank or columns 4-16 were blank.

or

In a control statement, continuation of a comment was specified by a nonblank character in column 72 and the following card image was not a continuation; that is, columns 1-3 were not blank.

System Action: Diagnosis of this control statement is terminated. Any additional control statements are scanned for syntax errors after which the job step is terminated. (The return code is 12.)

User Response: Correct the control statement and resubmit the job.

IEB910I NO SYSUT1 DD CARD - JOB TERMINATED

Explanation: A job control statement was not supplied for the SYSUT1 data set. Execution is impossible.

System Action: The job step is terminated. (The return code is 12.)

User Response: Add the SYSUT1 DD card and resubmit the job.

IEB911I NO SYSIN DD CARD - ALL DEFAULT OPTIONS ASSUMED

Explanation: A job control statement was not provided for SYSIN.

System Action: The utility is executed using all default options. (The return code is 4.)

User Response: None - if the use of default options was desired. If default options were not expected, add the SYSIN DD card and one or more control statements and resubmit the job.

IEB912I NO SYSPRINT DD CARD - DUMMY ASSUMED

Explanation: A job control statement was not supplied for the SYSPRINT data set.

System Aciton: The data set will be handled as if DUMMY had been specified; that is, no messages will be output to SYSPRINT. (The return code is 4.)

User Response: None - if no SYSPRINT output is desired. If output was desired, add the SYSPRINT DD card and resubmit the job.

IEB913I NO xxxxxxxx DD CARD - DUMMY ASSUMED

Explanation: A job control statement was not supplied for the xxxxxxxx data set. (xxxxxxxx may be either SYSUT2 or SYSUT3.)

System Aciton: The data set will be handled as if DUMMY had been specified; that is, no data will be output. The records that are passed to the user exit will be constructed using the default value (VB) of the RECFM DCB parameter. In some cases this may produce undesirable results. (The return code is 4.)

User Response: None - if no output was desired. If output was desired, add the xxxxxxxx DD card and resubmit the job.

IEB914I DCB SUBPARAMETER(S) MISSING IN xxxxxxxx DD CARD - DEFAULTS ASSUMED

Explanation: The LRECL, BLKSIZE and/or RECFM subparameters were not specified.

System Action: Defaults are assigned, and processing continues. (The return code is 4.)

User Response: None - if default parameters were expected. If default parameters were not expected, add the DCB subparameters to the xxxxxxxx DD card and resubmit the job.

IEB915I DDNAME xxxxxxxx CANNOT BE OPENED

Explanation: The data set specified cannot be opened due to an undetermined error.

System Action: The job step is terminated. (The return code is 16.)

User Response: None.

IEB916I INCONSISTENT xxxxxxxx DCB PARAMETERS

Explanation: Two or more DCB parameters for the indicated data set are inconsistent.

System Action: The job step is terminated. (The return code is 16.)

User Response: Correct the control statement and resubmit the job.

IEB917I LOAD MODULE SPECIFIED FOR xxxxxxxx NOT FOUND

Explanation: Either a user exit routine specified in the EXITS statement or a user translate table specified in the TCRGEN statement could not be located in the job library or link library.

System Action: The job step is terminated. (The return code is 16.)

User Response: Check the control statements for keypunch errors. If none are present, verify that the module is present in the link library or job library. If the module is present in the job library, check that the JOBLIB card is present.

IEB918I JOB TERMINATED AFTER xxxxxxxx EXIT

Explanation: A user-supplied exit routine requested termination upon return to the utility.

System Action: The job step is terminated. (The return code is 16.)

User Response: None.

IEB919I INSUFFICIENT STORAGE AVAILABLE

Explanation: Storage requested by a utility GETMAIN macro instruction could not be allocated. More main storage was requested than was available.

System Action: The job step is terminated. (The return code is 16.)

User Response: If additional storage is available, increase the value specified in the REGION parameter of the JOB statement or the EXEC statement. If additional storage is not available, reduce the value specified for BUFL in the SYSUT1 DD statement.

IEB920I ddd,device type,ddname,operation attempted,error description, asterisks,access method

Explanation: A permanent I/O error occurred on the ddname data set mounted on unit record device ddd.

System Action: The job step is terminated. (The return code is 16.)

User Response: If the error persists, have the computing system checked.

IEB921I ddd,device type,ddname,operation attempted,error description, relative block number,access method

Explanation: A permanent I/O error occurred on the ddname data set mounted on tape device ddd.

System Action: The job step is terminated. (The return code is 16.)

User Response: If the error persists, have the computing system checked.

IEB922I ddd,device type,ddname,operation attempted,error description,actual track address and block number,access method

Explanation: A permanent I/O error occurred on the ddname data set mounted on direct access device ddd.

System Action: The job step is terminated. (The return code is 16.)

User Response: If the error persists, have the computing system checked.

System Utility Messages

The IEHLIST Program

IEH101I NO CATALOG ON SPECIFIED VOLUME.

Explanation: No catalog exists on the volume identified in the LISTCTLG statement.

System Action: The request is ignored. (The return code is 8.)

IEH102I THIS VOLUME DOES NOT CONTAIN DATA SET xxx.

Explanation: The data set identified in the LISTVTOC or LISTPDS statement is not contained in the specified volume's table of contents.

System Action: The request is ignored. (The return code is 8.)

IEH103I INVALID CONTROL STATEMENT -- xxx.

Explanation: A utility statement is invalid. (The entire statement is written.)

System Action: The request is ignored. (The return code is 8.)

IEH104I THE PDS ORGANIZATION DOES NOT APPLY FOR DATA SET xxx.

Explanation: The data set identified in the LISTPDS statement is not partitioned.

System Action: The request is ignored. (The return code is 8.)

IEH105I ILLEGAL NODE POINT SPECIFIED, OR INCONSISTENT CATALOG STRUCTURE FOUND -- REQUEST TERMINATED.

Explanation: The node point identified in the LISTCTLG statement is invalid, or an incorrect catalog structure exists.

System Action: The request is ignored. (The return code is 8.)

IEH106I UNAVAILABLE DEVICE TYPE OR VOLUME I.D. SPECIFIED.

Explanation: The VOL parameter of the utility statement is invalid, or the volume cannot be mounted.

System Action: The request is ignored. (The return code is 8.)

IEH107I JOB TERMINATED -- I/O ERROR ON SYSIN.

Explanation: Additional input control statements cannot be read.

System Action: The job is terminated. (The return code is 16.)

IEH108I REQUEST TERMINATED -- PERMANENT I/O ERROR WHILE READING DATA SET.

Explanation: IEHLIST's error exit was taken while a volume table of contents, a catalog, or a partitioned data set was being read.

System Action: The job is terminated. (The return code is 12.)

IEH109I SYSIN CANNOT BE OPENED -- CHECK SYSIN DD CARD

Explanation: Either the SYSIN DD statement was inadvertently omitted form the job step or it was included but the SYSIN ddname was coded incorrectly.

System Action: The job is terminated. (The return code is 16.)

IEH110I JOB TERMINATED -- INVALID DCB PARAMETER

Explanation: The SYSIN DD statement specifies a block size that is not a multiple of the specified logical record length.

System Action: The job is terminated. (The return code is 16.)

IEH111I LL FIELD OF TTRLL IS NEGATIVE VALUE -- BYTES PRINTED ARE INCORRECT.

Explanation: A negative value is given for the number of bytes remaining on the track following the block.

System Action: Processing continues.

IEH112I MEMBERS OF SPECIFIED PDS NOT CREATED BY LINKAGE EDITOR - DUMP OPTION OUTPUT GENERATED.

Explanation: The directory entry is less than 34 bytes, indicating that the members were not created by the Linkage Editor.

System Action: No immediate action is taken. Processing continues as if the DUMP option were specified.

User Response: None.

The IEHPROGM Program

IEH201I INVALID REQUEST ... STATEMENT IGNORED.

Explanation: The utility statement contains an invalid operation.

System Action: The request is ignored. (The return code is 8.)

IEH202I INVALID KEYWORD OR CONTROL STATEMENT SYNTAX

Explanation: The utility control statement contains an invalid or incorrectly coded keyword in the operand field.

System Action: The request is ignored. (The return code is 8.)

IEH203I THE SYSCTLG DATA SET IS NOT AVAILABLE OR FORMS A LOOP

Explanation: No catalog exists on the specified control volume, or control volumes are connected incorrectly to each other.

System Action: The request is ignored. (The return code is 8.)

IEH204I STATUS OF THE REQUESTED TASK CANNOT BE DETERMINED. AN UNDEFINED ERROR CODE HAS BEEN ENCOUNTERED.

Explanation: The return code from a system macro instruction is invalid.

System Action: The request is ignored. (The return code is 8.)

IEH205I INFORMATION IN CONTROL STATEMENT IS (REDUNDANT/NOT SUFFICIENT).

Explanation: Information on the utility statement is either redundant or inadequate.

System Action: The request is ignored. (The return code is 8.)

IEH206I CVOL IS NOT DIRECT-ACCESS.

Explanation: The volume identified by CVOL is not on a direct access device.

System Action: The request is ignored. (The return code is 8.)

IEH207I STATUS OF USERS REQUEST TO (SCRATCH/RENAME) DATA SET xxx...

```
VOLUME ID    ACTION TAKEN    REASON
   xxx           xxx           xxx
    .             .             .
    .             .             .
    .             .             .
END OF LISTING OF DATA SETS TO BE SCRATCHED OR RENAMED.
```

Explanation: An unusual condition occurred during a SCRATCH or RENAME operation. The message names the data set, identifies the volumes on which the data set resides, states the action taken on each volume, and describes the condition.

System Action: The request is ignored. (The return code is 8.)

or

IEH207I STATUS OF USERS REQUEST TO SCRATCH THE VOLUME TABLE OF CONTENTS...

```
DATA SET NAME ACTION TAKEN  REASON
   xxx            xxx         xxx
    .              .           .
    .              .           .
    .              .           .
END OF SCRATCH VTOC.
```

Explanation: Either (1) an unusual condition occurred during a SCRATCH VTOC operation or (2) a data set has been successfully scratched. The message names each data set, states the action taken, and describes the condition.

IEH208I LIST TRUNCATED TO 1 VOLUME FOR SCRATCH VTOC.

Explanation: More than 1 volume is identified in a SCRATCH VTOC statement.

System Action: Only the first volume in the list is considered. (The return code is 8.)

IEH209I THE MODEL DATA SET CONTROL BLOCK IS NOT AVAILABLE.

Explanation: The required DSCB for a BLDG operation cannot be located on the specified volume.

System Action: The request is ignored. (The return code is 8.)

IEH210I REQUEST CANNOT BE SERVICED ... reason.

Explanation: An unusual condition occurred during a catalog or index operation. The condition is described in detail.

System Action: The request is ignored. (The return code is 8.)

IEH211I REQUIRED VOLUME COULD NOT BE MOUNTED.

Explanation: A device was not allocated for the required volume.

System Action: The request is ignored. (The return code is 8.)

IEH212I I/O ERROR ON SYSIN DATA SET -- JOB TERMINATED

Explanation: An input/output error occurred while the SYSIN data set was being read.

System Action: The program is terminated. (The return code is 8.)

IEH213I JOB TERMINATED -- INVALID BLOCKSIZE SPECIFIED IN SYSIN DCB

Explanation: The SYSIN DD statement specifies a block size that is not a multiple of the specified logical record length.

System Action: The job is terminated. (The return code is 16.)

IEH214I CONTINUATION CARD EXPECTED -- REQUESTS CANNOT BE SERVICED

Explanation: A nonblank was found in column 72 of a control card with an invalid continuation card following.

System Action: The request is not processed. (The return code is 8.)

IEH215I SYNTAX ERROR ENCOUNTERED IN NAME FIELD OF CONTROL STATEMENT -- PROCESSING IS CONTINUED

Explanation: The name field contained one of the following errors:

1. The first character was not alphabetic.
2. A character was encountered that was not an alphameric or national.
3. The name field is longer than 8 characters.

System Action: Processing continues. (The return code is 4.)

User Response: To eliminate the message, correct the control statement to comply with utility rules governing the name field.

The IEHMOVE Program

IEH301I INCLUDE OP NOT VALID

Explanation: The INCLUDE statement is not valid with the specified operation.

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH302I EXCLUDE OP NOT VALID

Explanation: The EXCLUDE statement is not valid with the specified operation.

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH303I REPLACE OP NOT VALID

Explanation: The REPLACE statement is not valid with the specified operation.

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH304I SUBORDINATE REQ-SKIPPED

Explanation: No MOVE or COPY statement precedes the specified INCLUDE, EXCLUDE, REPLACE, or SELECT statement.

System Action: The request is ignored. (The return code is 8.)

IEH305I MULTIPLE KEYWORD ERROR

Explanation: A control statement contains duplicate or conflicting keywords.

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH307I KEYWORD NOT PERMITTED

Explanation: A control statement contains an invalid keyword.

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH308I INVALID PARAMETER ERROR

Explanation: an invalid parameter (such as spelling, device type or serial number error, etc.) is specified on the listed control statement.

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH309I SYNTAX ERROR

Explanation: The syntax of the listed control statement is in error.

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH310I LENGTH ERROR

Explanation: Either the physical length of a keyword value is incorrect (for example, DSNAME=NINECHARS), or the EXPAND keyword specifies a number not falling between 1 and 99 decimal (for example, EXPAND=00 or EXPAND=100).

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH311I INCOMPLETE REQUEST.

Explanation: The control statement does not contain adequate information to perform the MOVE/COPY operation.

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH313I DATA SET xxx HAS INCORRECT FORMAT FOR UNLOADED DATA SET

Explanation: The request to move or copy an unloaded data set is ignored because its format is incorrect. The records are apparently out of sequence.

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH315I UNABLE TO FIND FROM VOLUME

Explanation: The program was unable to locate the "FROM" volume. The FROM keyword was probably omitted from the MOVE or COPY utility control statement.

System Action: The MOVE/COPY request is ignored. (The return code is 4.)

IEH316I MODEL DSCB FOR GENERATION DATA GROUP CANNOT BE WRITTEN

Explanation: An error (e.g., a permanent I/O error) occurred when an attempt was made to create the model DSCB.

System Action: The MOVE/COPY request is terminated. (The return code is 8.)

IEH319I MEMBER xxx NOT MOVED/COPIED. DUPLICATE NAME IN OUTPUT DATA SET.

Explanation: A member with the same name is contained in the output partitioned data set.

System Action: The request is ignored. (The return code is 4.)

IEH320I MEMBER xxx NOT FOUND IN DATA SET xxx.

Explanation: The named member cannot be located in the partitioned data set.

System Action: The request is ignored. (The return code is 8.)

IEH321I MEMBER xxx NOT MOVED/COPIED. OUTPUT DIRECTORY IS FULL.

Explanation: The directory of the output partitioned data set is full.

System Action: The named member was not moved or copied. (The return code is 8.)

IEH322I I/O ERROR ENCOUNTERED IN MEMBER xxx OF INPUT DATA SET xxx.

Explanation: A permanent error was detected while the named member was being read.

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH323I I/O ERROR ENCOUNTERED IN MEMBER xxx OF OUTPUT DATA SET xxx.

Explanation: A permanent error was detected while the named member was being written.

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH325I INVALID CATLG REQUEST IGNORED.

Explanation: The specified receiving volume is not direct access.

System Action: The moved or copied data set was not cataloged on this volume as requested. (The return code is 8.)

IEH326I I/O ERROR ENCOUNTERED IN OUTPUT DATA SET xxx.

Explanation: A permanent error was detected while the named data set was being written.

System Action: The request is terminated. (The return code is 8.)

IEH331I USER LABELS ARE NOT MOVED/COPIED. NO USER LABEL TRACK ALLOCATED FOR INPUT.

Explanation: A preallocated data set did not provide a user label track.

System Action: User labels are ignored. Normal Move/Copy operations continue.

IEH332I PERMANENT I/O ERROR WHILE READING USER INPUT HEADER LABELS. NO MORE LABELS WILL BE PROCESSED.

Explanation: OPEN encountered a permanent I/O error while attempting to read user input header labels.

System Action: The utility returns to the user pointing him to the label in error, ignores his return code and terminates operation.

IEH333I PERMANENT I/O ERROR WHILE READING USER INPUT TRAILER LABELS. NO MORE LABELS WILL BE PROCESSED.

Explanation: EOV encountered a permanent I/O error while attempting to read user input trailer labels.

System Action: The utility returns to the user pointing him to the label in error, ignores his return code and terminates operation.

IEH334I PERMANENT I/O ERROR WHILE WRITING USER OUTPUT HEADER LABELS. NO MORE LABELS WILL BE PROCESSED.

Explanation: OPEN encountered a permanent I/O error while attempting to write user output header labels.

System Action: The utility returns to the user pointing him to the label in error, ignores his return code and terminates operation.

IEH335I PERMANENT I/O ERROR WHILE WRITING USER OUTPUT TRAILER LABELS. NO MORE LABELS WILL BE PROCESSED.

Explanation: OPEN encountered a permanent I/O error while attempting to write user output trailer labels.

System Action: The utility returns a pointer to the user referring to the label in error, ignores his return code and terminates operation.

IEH336I AN UNCORRECTABLE ERROR OCCURED WHILE READING DATA SET xxxx

Explanation: The DECB for input data set indicated that an error occurred for the record just read that was something other than an I/O error or a record length check. This applicable to direct data sets only.

System Action: The utility terminates any further processing of that function.

IEH346I CATALOG CANNOT BE LOCATED, OR CONTROL VOLUMES ARE CONNECTED TO EACH OTHER.

Explanation: No catalog exists on the specified control volume, or control volumes are connected incorrectly to each other.

System Action: The request is ignored. (The return code is 8.)

IEH348I I/O ERROR ENCOUNTERED IN CATALOG.

Explanation: A permanent error was encountered while reading or writing the catalog.

System Action: The request is terminated. (The return code is 8.)

IEH349I UNABLE TO MOUNT VOLUME xxx... action.

Explanation: No device was allocated for the specified volume.

System Action: The request is ignored. (The return code is 8.)

IEH351I DATA SET xxx NOT CATALOGED. SPACE NOT AVAILABLE IN THE CATALOG.

Explanation: The catalog is full.

System Action: The named data set was not cataloged. (The return code is 8.)

IEH354I DATA SET xxx NOT CATALOGED. INDEX STRUCTURE INCONSISTENT.

Explanation: Either an invalid index structure exists or the catalog already has an entry for the named data set.

System Action: The named data set was not cataloged. (The return code is 8.)

IEH356I DATA SET xxx NOT CATALOGED. INVALID DATA SET NAME.

Explanation: The specified data set name is inappropriate for cataloging.

System Action: The data set was not cataloged. (The return code is 8.)

IEH361I DATA SET xxxxxx NOT MOVED/COPIED TO VOLUME(S)

Explanation: Due to an abnormal condition (such as an I/O error), the named data set was not moved (or copied).

System Action: The MOVE/COPY request is terminated. (The return code is 8.)

IEH362I DATA SET xxxxxx MAY NOT BE SCRATCHED ON VOLUME(S)

Explanation: Due to an abnormal condition (such an an I/O error), the named data set was not scratched.

System Action: Processing continues with the next function to be performed, if any. (The return code is 4.)

IEH363I DATA SET JUST COPIED WAS NOT SUCCESSFULLY UNCATALOGED

Explanation: The data set was copied but not uncataloged. A permanent I/O error occurred during the uncatalog operation.

System Action: Processing continues with the next function to be performed, if any. (The return code is 4.)

IEH364I THE DATA SET JUST COPIED WAS NOT SUCCESSFULLY CATALOGED

Explanation: The data set was copied but not cataloged on the "TO" volume because of one of the following conditions:

- The SYSCTLG data set does not exist on the specified volume.
- The existing index structure does not permit the cataloging of the data set.

- No space is available in the catalog.
- A permanent I/O error occurred during the catalog operation.
- The data set may already be catalogued on the receiving volume.

System Action: Processing continues with the next function to be performed, if any. (The return code is 4.)

IEH365I DATA SET xxxxxx MAY STILL EXIST ON VOLUME(S)

Explanation: The named data set was moved but not scratched from the source volume(s). An unusual condition, such as a permanent I/O error occurred during the scratch operation.

System Action: Processing continues with the next function to be performed, if any. (The return code is 4.)

IEH366I THE DATA SET JUST MOVED MAY EXIST WITH AN INTERNALLY GENERATED NAME ON VOLUME(S)

Explanation: Due to an unusual condition, such as a permanent I/O error, a specified rename operation was not successful. An internally generated name may have been assigned to the moved data set.

System Action: Processing continues with the next function to be performed, if any. (The return code is 8.)

IEH367I THE DATA SET JUST MOVED WAS NOT SUCCESSFULLY UNCATALOGED

Explanation: The data set was moved but not uncataloged. A permanent I/O error occurred during the uncatalog operation.

System Action: Processing continues with the next function to be performed, if any. (The return code is 4.)

IEH368I THE DATA SET JUST MOVED WAS NOT SUCCESSFULLY RECATALOGED

Explanation: The data set was moved, but the catalog was not updated. Either an I/O error occurred during the catalog operation, or the existing index structure in the catalog does not permit the cataloging of the data set.

System Action: Processing continues with the next function to be perfomred, if any. (The return code is 4.)

IEH372I I/O ERROR ENCOUNTERED IN WORK DATA SET.

Explanation: A permanent input/output error was detected while reading or writing the work data set.

System Action: The request is terminated. (The return code is 12.)

IEH373I UNABLE TO MOUNT VOLUME xxx. SOME INCLUDE OR REPLACE REQUESTS IGNORED.

Explanation: The program cannot mount the named volume.

System Action: The INCLUDE or REPLACE requests that refer to the specified volume were ignored. (The return code is 8.)

IEH374I DATA SET xxx NOT FOUND ON VOLUME xxx. INCLUDE OR REPLACE REQUEST IGNORED.

Explanation: The named data set does not reside on the specified volume.

System Action: The INCLUDE or REPLACE statements that refer to the specified data set were ignored. (The return code is 8.)

IEH375I DATA SET xxx IS NOT PARTITIONED. INCLUDE OR REPLACE REQUEST IGNORED.

Explanation: The named data set is not partitioned.

System Action: The INCLUDE request or the including part of the REPLACE request was ignored. (The return code is 8.)

IEH376I RECORD CHARACTERISTICS NOT COMPATIBLE (xxx). INCLUDE OR REPLACE REQUEST IGNORED.

Explanation: An attribute (xxx) of the output data set is not compatible with that of the input data set.

System Action: The INCLUDE request or the including part of the REPLACE request was ignored. (The return code is 8.)

IEH380I MEMBER xxx NOT FOUND IN DATA SET xxx. INCLUDE OR REPLACE REQUESTS IGNORED.

Explanation: The named member is not contained in the named partitioned data set.

System Action: The INCLUDE request or the including part of the REPLACE request is ignored. (The return code is 8.)

IEH381I ERROR ENCOUNTERED IN SCRATCHING WORK FILES

Explanation: A work file or files could not be scratched. Either a work file could not be located, or an I/O error occurred during the scratch operation.

System Action: The MOVE/COPY program is terminated. (The return code is 12.)

IEH383I INVALID DEVICE NAME

Explanation: A device name on the utility statement is invalid.

System Action: The request is ignored. (The return code is 8.)

IEH384I GENERIC DEVICE NAME ERR

Explanation: A device name on the utiltiy control statement is invalid.

System Action: The request is ignored. (The return code is 8.)

IEH385I SELECT OP NOT VALID

Explanation: The SELECT statement is not compatible with the specified utility control statement.

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH388I UNABLE TO ALLOCATE IEHMOVE WORK FILES

Explanation: The IEHMOVE program was unable to allocate space for the work files. Either no SYSUT1 DD statement was included with the job step or there was insufficient space on the direct access volume assigned to the SYSUT1 DD statement.

System Action: The IEHMOVE program is terminated. (The return code is 12.)

IEH389I I/O ERROR ENCOUNTERED IN INPUT DATA SET

Explanation: A permanent error was detected while the input data set was being read.

System Action: The request is terminated. (The return code is 8.)

IEH390I NON-ALPHABETIC FIRST CHAR IN RENAME

Explanation: In a job control statement, the first character in a level of a rename is not alphabetic.

System Action: Processing continues with the next function to be performed, if any. (The return code is 8.)

IEH401I through IEH429I DATA SET xxx {UNLOADED|NOT MOVED/COPIED}--reason

Explanation: The named data set was unloaded or it was not moved or copied, for the reason given.

System Action: A request to move or copy is ignored or the data set is unloaded, whichever the case may be. (The return code is 4.)

IEH433I DATA SET NOT MOVED/COPIED BECAUSE INCLUDE, EXCLUDE, SELECT OR REPLACE REQUEST WHILE LOADING/UNLOADING

Explanation: Subordinate requests cannot be processed during a load or unload of a data set.

System Action: The MOVE/COPY request is ignored. (The return code is 4.)

IEH435I ERROR ENCOUNTERED WHILE ANALYZING THE SYSCTLG DATA SET

Explanation: The catalog to be moved is in error.

System Action: The request is terminated. (The return code is 8.)

IEH436I DATA SET xxx, VOLUME xxx, NOT SCRATCHED DUE TO I/O ERROR.

System Action: The named data set was not scratched after the MOVE operation. (The return code is 8.)

IEH450I REQUEST TERMINATED BECAUSE DATA SET SPANS MORE THAN 5 VOLUMES

Explanation: A data set extends over the maximum of five volumes.

System Action: A request to move or copy is ignored. (The return code is 8.)

IEH451I TRACK OVERFLOW FEATURE REQUIRED ON DEVICE THAT DOES NOT HAVE TRACK OVERFLOW FEATURE

Explanation: A data set to be moved or copied was originally written with track overflow, but the source device does not support the track overflow feature.

System Action: A request to move or copy is ignored. (The return code is 8.)

IEH452I THE DATA SET BEING MOVED/COPIED IS MARKED UNMOVABLE. UNMOVABLE DATA MUST BE UPDATED BEFORE ITS NEXT USE

Explantion: A data set being moved/copied from one direct access volume to another contains location dependent information (i.e., the unmovable bit in the DSORG field of the DSCB is on).

System Action: The data set is moved and processing continues normally.

User Response: Update location dependent information in the moved/copied version of the data set, e.g., the IEHIOSUP program is used to update TTR entries in the transfer control tables of the supervisor call library (SVC library) after this library has been moved.

IEH460I INVALID DATA SET ORGANIZATION

Explanation: The source data set is not a partitioned, physical sequential, or direct access (BDAM) data set; therefore, it cannot be processed by the utility program.

System Action: The request is terminated. (The return code is 12.)

IEH461I UNABLE TO OPEN {INPUT / SYSIN} DATA SET

Explanation: Either no DD statement was provided to define the input data set or the specified block size is not a multiple of the logical record length.

System Action: The request is terminated. (The return code is 12.)

IEH462I NO RECORD FOUND OCCURRED READING DATA SET xxxxxxxx.

Explanation: One of the following situations was encountered while reading a direct organization data set:

- The record format of the data set is fixed (F) and a track within the data set is not completely filled with records.
- The record format is variable (V) or undefined (U) and not all tracks were initialized when the data set was created.
- An uncorrectable error occurred.

System Action: The system produces an additional message (IEH361I) and issues a return code of 8.

User Response: Check the data set to ensure that it conforms to the standards for a direct organization data set and rerun the program.

The IEHINITT Program

IEH601I INVALID CONTROL STATEMENT

Explanation: An INITT utility control statement is coded incorrectly.

System Action: Processing continues with the next INITT utility control statement. (The return code is 8.)

User Response: Correct the control statement and rerun the job to label those tapes that were bypassed.

IEH602I INVALID KEYWORD

Explanation: An INITT utility control statement contains an invalid keyword in the operand field.

System Action: Processing continues with the next INITT control statement. (The return code is 8.)

User Response: Correct the control statement and rerun the job to label those tapes that were bypassed.

IEH603I INVALID PARAMETER VALUE

Explanation: An INITT utility control statement contains an invalid parameter in the operand field.

System Action: Processing continues with the next INITT control statement. (The return code is 8.)

User Response: Correct the control statement and rerun the job to label those tapes that were bypassed.

IEH604I OPERATOR SUPPRESSED VOLUME LABEL xxxxxx

Explanation: The tape that was to be labeled with serial number xxxxxx was not mounted by the operator. (The operator should indicate why the tape was not mounted.)

System Action: The current serial number is reserved for the unmounted tape and the next number is used for the next tape to be labeled.

IEH605I INVALID DEVICE ALLOCATED ON xxx

Explanation: Either an allocated device is unacceptable or it is not on line; that is, it was removed from operation.

System Action: The device is removed from the list of devices allocated to this job step by the associated DD statement. (The return code is 8.)

User Response: Check the applicable DD statement for correct parameters. If the DD statement is correct, have the computing system checked.

IEH606I PERMANENT I/O ERROR ON xxx

Explanation: An input/output error was detected on the indicated device (xxx).

System Action: The device is removed from the list of devices allocated to this job step by the associated DD statement. (The return code is 8.)

User Response: Have the computing system checked.

IEH607I ALLOCATED DEVICES EXHAUSTED

Explanation: All devices allocated to this job step (by the DD statement associated with the control card being processed) have been eliminated as mountable devices.

System Action: Processing continues with the next INITT control statement. (The return code is 8.)

User Response: Check the applicable DD statement for correct parameters. If the DD statement is correct, have the computing system checked.

IEH608I I/O ERROR ON SYSIN. JOB TERMINATED.

Explanation: A permanent error was encountered when the SYSIN data set was being opened, or when it was being read.

System Action: The job is terminated. (The return code is 16.)

User Response: Check the SYSIN DD statement for correct parameters. If the DD statement is correct, the error probably occurred when the data set was being read. Have the job step rerun.

The IEHIOSUP Program

IEH700I THE LOAD MODULES OF xxxxxxxx HAVE NOT BEEN UPDATED

Explanation: Due to an unrecoverable error condition, the named modules (up to seven) have not been updated.

IEH701I A PERMANENT I/O ERROR DETECTED ON A COMMAND CHAIN OF SEARCH ID=,TIC,READ DATA. MBBCCHHR=xxxxxxxx

Explanation: A permanent I/O error occurred while the program was searching for a module. xxxxxxxx is the absolute address of the module for which the search was being made.

System Action: The job step is terminated. (The return code is 12.)

IEH702I NO FOUND CONDITION SEARCHING FOR xxxxxxxx

Explanation: The named member (xxxxxxxx) was not found in the directory of the SYS1.SVCLIB data set.

System Action: The job step is terminated. (The return code is 12.)

IEH703I A PERMANENT I/O ERROR DETECTED CN A COMMAND CHAIN OF SEARCH ID=,TIC,WRITE DATA. MBBCCHHR=xxxxxxxx

Explanation: A permanent write error occurred while updating the named member (xxxxxxxx).

System Action: The job step is terminated. (The return code is 12.)

IEH704I BLDL HAS DETECTED A PERMANENT I/O ERROR

Explanation: A permanent I/O error occurred during the execution of the BLDL macro-instruction.

System Action: The job step is terminated. (The return code is 12.)

IEH705I ZERO LENGTH RECORD READ AT ADDRESS MBBCCHR xxxxxxxx

Explanation: An unexpected zero length record was found by IEHIOSUP during an update.

System Action: The job step is terminated. The return code is 12.

The IEHDASDR Program

IEH800I INVALID CONTROL STATEMENT, LAST COLUMN SCANNED= (decimal number from 1 to 71)

Explanation: An IEHDASDR utility control statement is coded incorrectly (e.g., a syntax error was encountered). The column number of the last column scanned is included in the message.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: Correct the control statement and rerun the operation.

IEH801I INVALID COMMAND= (command)

Explanation: An IEHDASDR utility control statement contains an operation that is not valid; for example, RESTERE, rather than RESTORE.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: Correct the command and rerun the operation.

IEH802I INVALID KEYWORD= (keyword)

Explanation: An IEHDASDR utility control statement contains an invalid keyword in the operand field; for example, CPYVULID=YES, rather than CPYVOLID=YES.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: Correct the keyword and rerun the operation. Rep

IEH803I INVALID PARAMETER= (parameter)

Explanation: An IEHDASDR utility control statement contains an invalid parameter value in the operand field; for example, CPYVOLID=YSE, rather than CPYVOLID=YES.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: Correct the parameter and rerun the operation.

IEH804I REQUIRED KEYWORD(S) MISSING

Explanation: A necessary keyword or keywords are omitted from an IEHDASDR utility control statement.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: Correct the control statement and rerun the operation.

IEH805I DDNAME= ddname CANNOT BE OPENED

Explanation: The named DD statement does not exist or is coded incorrectly.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: Correct the DD statement and rerun the operation.

IEH806I { RESTORE TO / DUMP TO / LABEL OF / GETALT ON / FORMAT OF / ANALYSIS OF } { DDNAME=ddname IS COMPLETE. [VOLUME SERIAL NO.=xxxxxx] }

Explanation: The indicated function has completed successfully on the device specified on the indicated DD statement. If the "to" volume is a direct access volume, the volume serial number is indicated in the message.

System Action: Processing continues with the next IEHDASDR utility control statement.

Response: None.

IEH807D cuu (or cuu/b) HAS UNEXPIRED DATA SETS, serial, jobname, stepname.

Explanation: The user specified the PURGE keyword and the IEHDASDR program encountered one or more unexpired data sets on the indicated volume while attempting to perform an ANALYZE, FORMAT, DUMP, or RESTORE operation.

System Action: Processing continues according to the operator's response.

Response: To continue processing, response REPLY xx,'U'. To terminate the operation and continue processing with the next function, REPLY xx,'T'.

IEH808I REPLY IN ERROR. REPLY WITH 'U' OR 'T', jobname, stepname.

Explanation: The operator issued an invalid reply to message IEH807A or IEH841A.

Response: Enter the correct reply when message IEH807A or IEH841A is again issued.

IEH809I {R}{cuu (or cuu/b), serial,
{N}{jobname, stepname {,NOW OFFLINE}

Explanation: R indicates that the specified volume, cuu (or cuu/b for a 2321 volume), is to be demounted. The previous IEHDASDR operation resulted in identical serial numbers being placed on two or more direct access volumes. If the duplicate serial number was placed on a 2321 or a nondemountable volume, that volume is also placed offline. N indicates that the specified volume, cuu (or cuu/b), was assigned the indicated serial number.

Response: R -- demount the specified volume. N -- no response necessary.

IEH810I TODD DDNAME=ddname IS NOT DIRECT ACCESS

Explanation: The device defined by the indicated DD statement is not a direct access device.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: Correct the DD statement so that it describes a direct access device.

IEH811I FROM DD DDNAME=ddname IS NOT A TAPE

Explanation: The device defined by the indicated DD statement is not a magnetic tape drive.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: Correct the DD statement so that it describes a magnetic tape drive.

IEH812I UNABLE TO MATCH DDNAME=ddname

Explanation: A ddname specified in an IEHDASDR utility control statement has no corresponding ddname in a DD statement. Either a necessary DD statement is missing or a ddname is misspelled in an existing DD statement.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: Correct the mispelling or supply the required DD card.

IEH813I I/O ERROR ENCOUNTERED DURING SVC{82}{29}, ddname

or

I/O ERROR jobname, stepname, unit address, device type, ddname, operation attempted, error description, last seek address or block count, access method.

Explanation: Either an I/O error occurred during an SVC 29 or 82 (as indicated in the message) or an I/O error occurred while processing otherwise on the named device. If possible, error analysis information such as jobname, stepname, unit address, device type, etc., is included in the message.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: For I/O errors on a direct access TODD volume, the volume should be varied offline, and the full surface analysis performed using the ANALYZE function of IEHDASDR. If the direct access volume can't be analyzed successfully, call a Field Engineer. For an I/O error on a TODD tape volume, mount another tape or move the tape to another drive and/or clean the read/write heads.

For an I/O error on a direct access FROMDD volume, move it to another drive (if possible) and rerun the operation. If the I/O error is due to a data check or a missing address marker, run the IEHATLAS utility program or the Independent utility program Recover/Replace. For an I/O error on a FROMDD tape volume, move it to another drive, and/or clean the read/write heads.

IEH814I { GETALT ON / RESTORE TO / DUMP TO / ANALYSIS OF / FORMAT OF / LABEL OF } { SYSTEM RESIDENCE / IS NOT ALLOWED. / DDNAME=ddname }

Explanation: One of the named functions is specified for the system residence volume. The ddname identifies the DD statement defining the system residence volume.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: Correct the DD statement so it does not describe the system resident volume.

IEH815I INCORRECT DEVICE TYPE ON RESTORE. DDNAME=ddname

Explanation: The device type of the direct access device being restored (ddname) does not match the device type from which the restore tape was created.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: Correct the DD statement or mount the proper tape or direct access volume.

IEH816I NOT A RESTORE TAPE ON DDNAME=ddname

Explanation: The magnetic tape volume mounted on the device defined by the indicated DD statement is not a "restore" tape volume; that is, it does not contain dumped direct access data.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: Correct the DD statement or mount the proper tape.

IEH817I FORMAT 5 DSCB IN VTOC FOUND TO BE INCORRECT FOR FROMDD=ddname

Explanation: Prior to a DUMP operation, the entries in the Format 5 DSCB were found to be invalid on the source volume.

System Action: All tracks within the specified range of tracks are dumped, whether they are "owned" or not. Processing continues. (The return code is 4.)

Response: List the VTOC of the volume using IEHLIST, DUMP format, and contact a customer engineer.

IEH818I MAIN STORAGE REQUIREMENTS NOT AVAILABLE FOR THIS FUNCTION

Explanation: Insufficient main storage space was available for this function. The function was terminated.

System Action: Processing continues with the next IEHDASDR control statement. (The return code is 8.)

Response: Enlarge the partition, region or priority and rerun the operation.

IEH819I FROMDD=ddname IS NOT DIRECT ACCESS

Explanation: An attempt was made to dump a volume other than a direct access volume. The named DD statement defines the device containing the volume that was to be dumped.

System Action: Processing continues with the next IEHDASDR control statement. (The return code is 8.)

Response: Correct the DD statement to describe a direct access volume.

IEH820I INVALID DUMP DEVICE SPECIFIED. DDNAME=ddname

Explanation: A device other than an identical type direct access device, a magnetic tape drive, or a system output device was defined as the receiving device for a DUMP operation. The named DD statement defines the device that was to be the receiving device.

System Action: Processing continues with the next IEHDASDR control statement. (The return code is 8.)

Response: Correct the FROMDD statement or the TODD statement referenced.

IEH821I INVALID COPY REQUEST. DDNAME=ddname

Explanation: (1) An ANALYZE, FORMAT, DUMP or RESTORE utility control statement defines devices representing two or more device types or (2) multiple copies were requested where not permitted. The named DD statement defines the invalid device.

System Action: Processing continues. (The return code is 8.)

Response: Correct the TODD statement to reflect like device types, and then rerun the operation.

IEH822I INVALID TRACK ADDRESS SPECIFIED. DDNAME=ddname

Explanation: An invalid track address has been specified on either a DUMP statement (BEGIN or END address) or a GETALT statement (TRACK address). The named DD statement defines the device containing the volume to which the invalid track pertains.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: Correct the track address on the control statement and rerun the operation.

IEH823I THE DEVICE SPECIFIED BY TODD=ddname IS A DRUM DEVICE. AN ALTERNATE TRACK MUST BE ASSIGNED MANUALLY.

Explanation: Either a defective track has been encountered on a drum device during processing of an ANALYZE statement, or the user has defined a drum device through the use of a GETALT statement. The ddname is that of the DD statement defining the drum device.

System Action: Processing continues with the next IEHDASDR utility control statement, unless the operation is an ANALYZE operation, in which case processing continues with the same statement. (The return code is 8.)

Response: Have a customer engineer manually assign an alternate track.

LABEL
IEH824I ANALYZE TERMINATED. DEVICE NOT OFF-LINE AND CONFIRMED. TODD=cuu (or cuu/b for a 2321 volume).

Explanation: The function is not performed because the specified device was not placed offline prior to the execution of the job step and confirmed by the operator during the job step or the referenced UCB is not in the system. TODD specifies the channel and unit address of the device containing the volume to be analyzed or labeled.

System Action: Processing continues with the next IEHDASDR control statement. (The return code is 8.)

Response: Vary the device offline and rerun the operation, confirming the request by responding 'REPLY' xx,'U' to message IEH841D.

IEH825I INVALID VTOC LIMITS SPECIFIED FOR TODD=ddname

Explanation: The VTOC=xxxxx keyword or EXTENT=xxxxx keyword in an ANALYZE or FORMAT control statement specifies an invalid starting address or extent. The indicated ddname is that of the DD statement defining the device containing the volume to be analyzed or formatted.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: Correct the VTOC or EXTENT parameter. The VTOC cannot begin on track 0 (or on track 1 for 2302 or 2311 with IPL text.)

IEH826I IPL TEXT NOT FOUND OR APPLICABLE FOR TODD=ddname

Explanation: Either an attempt was made to supply IPL text for a volume other than a 2301, 2303, 2311, or 2314 volume, or IPL text is to be written on a valid volume but the IEHDASDR program cannot locate the source copy of the IPL text. Also, the END card for IPL text may be missing or incorrect (not in columns 2-4).

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: Correct the direct access DD statement referenced by the IPLDD keyword to point to the correct data set, or remove the IPLDD Keyword and follow the first control statement with the IPLTXT statement and the IPL text, or remove all references to IPL text.

IEH827I NO MORE ALTERNATE TRACKS AVAILABLE FOR TODD=ddname

Explanation: An attempt was made to assign an alternate track; however, there are no unassigned alternates available. The named DD statement defines the device containing the volume on which an alternate track was to be assigned.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: ANALYZE the direct access device with FLAGTEST=NO.

IEH828I TRACK ZERO IS DEFECTIVE ON TODD=ddname. THIS VOLUME IS NON-IPLABLE.

Explanation: Track 0 on a volume that is to be analyzed, dumped to, or restored to is defective. The volume cannot be "IPLed." The named DD statement defines the device containing the volume that has the defective track.

System Action: For the ANALYZE/FORMAT operations, if IPL text was not requested, processing continues with this control statement. If it was requested, processing continues with the next IEHDASDR utility control statement and the return code is 8. This message is also issued as a warning by the DUMP and RESTORE functions.

Response: If doing an ANALYZE or FORMAT operation and IPLTXT is required, mount another volume and rerun the operation.

IEH829I HA-R0 AREA WAS DEFECTIVE. TODD=ddname

Explanation: During an ANALYZE or FORMAT operation, the home address/record zero area of a track was found defective. The named DD statement defines the device containing the volume that has the defective HA-R0 area on one of its tracks.

System Action: The action taken depends on the device type of the volume having the defective HA-R0 area:

2311
or
2302 -- The function is terminated. (The return code is 8.)

2314
or
2321 -- An attempt is made to move the HA-R0 area to another spot on the track. If the attempt is unsuccessful, a return code of 8 is

Response: For a mountable volume, move it to another drive. If the error persists, call a customer engineer.

Notes: Message IEH831I is issued to identify the defective track.

Message IEH829I is not issued if the volume having the defective HA-R0 area is either a 2301 or a 2303 volume. (Instead, message IEH823I is issued.)

IEH830I THE VOLUME SPECIFIED BY TODD=ddname HAS BECOME UNUSABLE

Explanation: GETALT has been specified for the VTOC area (all data on the specified track is lost), or an ANALYZE, FORMAT, RESTORE, or DUMP operation has not completed successfully, thus leaving the volume in an unusable condition. For a direct access volume, the named DD statement defines the device containing the volume that has become unusable. For a tape volume, the named DD statement defines the device containing the volume that has become unusable as a restore tape.

System Action: Processing continues with the next IEHDASDR control statement. (The return code is 8.)

User Response: In the case where the TODD volume is a direct access volume, the volume should be analyzed offline.

IEH831I DEFECTIVE TRACK ON TODD=ddname WAS ccccchhhh

Explanation: This message lists the track address (ccccchhhh) of a track found defective during an ANALYZE or FORMAT operation or a track specified on a GETALT statement. The named DD statement defines the device containing the volume that has the defective track.

System Action: Processing continues.

Response: None.

IEH832I ALTERNATE TRACK ASSIGNED ON TODD=ddname is {ccccchhhh / N/A}

Explanation: This message lists the alternate track address of a track assigned for a track found defective during an ANALYZE or FORMAT operation or for a track specified in a GETALT statement. The named DD statement defines the device containing the volume on which the alternate is assigned. If N/A appears in the message, no alternate track was assigned because the defective track is in the alternate area.

System Action: Processing continues.

Response: None.

IEH833I 3 BLANK TRACKS {ON HEAD CHECK. / ON CYLINDER CHECK. / ON STRIP CHECK. / ON SUBCELL CHECK.} TRK=ccccchhhh,ddname

Explanation: This message is issued during an ANALYZE operation on a 2321 volume for one of the following reasons:

a) 3 BLANK TRACKS ON HEAD CHECK -- Three blank tracks were encountered during an analysis or address compare test on each track of a cylinder.

b) 3 BLANK TRACKS ON CYLINDER CHECK -- Three blank tracks were encountered during an address compare test of the first track of each cylinder on a strip.

c) 3 BLANK TRACKS ON STRIP CHECK -- Three blank tracks were encountered during an address compare test of the first track on each strip of a subcell.

d) 3 BLANK TRACKS ON SUBCELL CHECK -- Three blank tracks were encountered during an address compare test of the first track of each subcell of a cell.

Note: This message usually indicates that a 2321 has failed to "pick" a strip.

System Action: Processing continues with the next IEHDASDR control statement. (The return code is 8.)

Response: If the error persists, call a customer engineer.

IEH834I DIRECT ACCESS DEVICE NOT SUPPORTED. DDNAME=ddname

Explanation: A direct access device type other than a supported device type has been specified. The named DD statement defines the invalid device.

System Action: Processing continues with the next IEHDASDR control statement. (The return code is 8.)

Response: Correct the DD statement referenced to reflect a supported device type.

IEH835I TAPE DD STATEMENT DOES NOT SPECIFY CORRECT LABEL INFORMATION FOR SECURITY PROTECTION

Explanation: A DUMP operation from direct access to tape involved password protected data sets; however, one or more of the following error conditions exists in the DD statement defining the magnetic tape volume:

- The LABEL parameter specifies a label other than a standard label.

- The PASSWORD subparameter is omitted from the LABEL parameter.

- The LABEL parameter specifies a file number other than 1.

System Action: The DUMP operation is terminated. Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: Correct the DD statement, the LABEL parameter or the PASSWORD suoparameter, and rerun the operation.

IEH836I INCORRECT PASSWORD WAS GIVEN FOR A DATA SET ON DDNAME=ddname

Explanation: The operator did not provide the correct password for a password protected data set on the specified volume.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: Provide the operator with a list of data set names and corresponding passwords, and reun the operation.

IEH837I UNEXPIRED DATA SET(S) NOT CONFIRMED ON TODD DDNAME=ddname

Explanation: The IEHDASDR program encountered one or more unexpired data sets on a direct access volume. Either the user did not code the PURGE keyword or the operator did not respond with REPLY xx,'U' after message IEH807A was issued.

System Action: The operation is terminated. Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: Mount the proper direct access volume or code PURGE=YES on the utility control statement and respond REPLY xx,'U' after message IEH807D.

IEH838I INVALID BLOCK SIZE SPECIFIED. DDNAME={SYSIN / SYSPRINT}

Explanation: The block size specified on the indicated DD statement is not a multiple of the logical record length (LRECL). SYSIN block size must be a multiple of 80; SYSPRINT block size should be a multiple of 121.

System Action: If an invalid block size is specified for the SYSIN data set, the IEHDASDR program is terminated. (The return code is 16.)

If an invalid block size is specified for the SYSPRINT data set, a default block size of 121 is assigned. (The return code is 4.)

Response: Correct the blocksize on the DD statement.

IEH839I HIGHEST RETURN CODE ENCOUNTERED WAS xx

Explanation: The IEHDASDR program has completed operation. The highest return code encountered was xx.

Response: None.

IEH840I NO DD CARD PROVIDED FOR SECURITY DATA SET ON FROMDD=ddname

Explanation: The user has failed to provide a DD statement defining a security protected data set on the named device.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 8.)

Response: Provide a DD statement for each security protected data set on the volume, and rerun the operation.

IEH841D cuu (or cuu/b for a 2321 volume) CONFIRM REQUEST TO {INITIALIZE / LABEL}

Explanation: A volume is to be initialized or labeled offline and the operator is requested to verify that the desired volume is mounted on the specified offline device (cuu or cuu/b).

System Action: Processing continues according to the operator's response.

Response: To continue processing, respond REPLY xx,'U'. To terminate the operation and continue with the next function, respond REPLY xx,'T'.

IEH842I DATA CHECK IN KEY FIELD ON TRK=ccccHHHH,ddname (Followed by defective record.)

Explanation: When dumping to SYSOUT, the DUMP function was unable to read the specified field without a data check. If there is a data check in both the key and data fields, this message appears twice before it is followed by the defective record.

System Action: The system output device contains the record as read. Processing continues. If data check is in count field on a 2301, the operator terminates.

Response: Use the IEHATLAS utility to assign an alternate to the defective track, copy and correct the defective data.

IEH843I DATA CHECK IN COUNT FIELD AND POSSIBLY IN KEY AND DATA FIELDS ON TRK=ccccHHHH,ddname (Followed by the defective record and the remainder of the track.)

Explanation: When dumping to SYSOUT, the DUMP function was unable to read the count field without a data check. Using the key and data lengths from this count, DUMP was unable to read the key and/or data fields without a data check. The original data check may have been in the length fields of the count, or there may actually be a data check in the key and/or data fields.

System Action: The system output device contains the record as read and the remainder of the track (as if one record). Processing continues.

Response: Use the IEHATLAS utility to assign an alternate for the defective track, copy and correct the defective data.

IEH844I MISSING ADDRESS MARKER ON TRK=ccccHHHH,ddname (Followed, if possible, by the defective record and the remainder of the track.)

Explanation: When dumping to SYSOUT, the DUMP function was unable to read the specified track without an address marker.

System Action: The system output device contains the last record as read and the remainder of the track (as if one record). Processing continues.

Response: Use the IEHATLAS utility to assign an alternate for the defective track, copy and correct the defective data.

IEH845I DEVICE NOT SUPPORTED FOR OFFLINE QUICK DASDI FEATURE. TODD=cuu.

Explanation: The function is not performed because the specified offline device was other than a 2314 volume.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 08.)

Response: Vary the device online, changing the TODD keyword to indicate a ddname, and rerun the operation; or, remove the PASSES=0 keyword from the control card, and rerun the operation requesting full surface analysis by the ANALYZE function.

IEH846I INVALID REQUEST FOR OFFLINE QUICK DASDI FUNCTION. TODD=cuu.

Explanation: The "QUICK DASDI" feature of the IEHDASDR ANALYZE function could successfully read the volume label of the specified offline 2314 volume.

System Action: Processing continues with the next IEHDASDR utility control statement. (The return code is 08.)

Response: Vary the 2314 volume online, changing the TODD keyword to indicate a ddname, and rerun the operation. You may also wish to add the PURGE=YES keyword for the online operation.

IEHATLAS Program

IEH900I SUCCESSFUL COMPLETION. AN ALTERNATE TRACK HAS BEEN ASSIGNED. COMPLETION CODE=00

Explanation: An alternate track has been assigned and data has been transferred from the bad track to the alternate.

System Action: The job step is terminated.

User Response: None

IEH901I SUCCESSFUL COMPLETION. NO ALTERNATE TRACK ASSIGNED. COMPLETION CODE=00

Explanation: The utility successfully rewrote the record in error.

System Action: The job step is terminated.

User Response: None

IEH902I I/O ERROR IN ALTERNATE TRACK ASSIGNMENT. COMPLETION CODE=16

Explanation: An alternate track was not assigned due to I/O errors after n attempts at assigning an alternate. N is equal to the lesser of 10% of the assigned alternates or the number of alternates available at the time IEHATLAS is called.

System Action: The job step is terminated.

User Response: ANALYZE the pack and rerun job.

IEH903I REQUIRED DD CARD MISSION. COMPLETION CODE=16

Explanation: Either the SYSUT1 or SYSIN data set could not be opened. The DD statement defining the data set was not included in the input stream.

System Action: The job step is terminated.

User Response: Add the required DD card and rerun the jobstep.

IEH904I INVALID DCB PARAMETERS FOR SYSIN. COMPLETION CODE=16

Explanation: SYSIN DCB blocksize was not a multiple of LRECL (80 bytes).

System Action: The job step is terminated.

User Response: Correct the blocksize for the SYSIN DD card and rerun the job step.

IEH905I INVALID OR MISSION CONTROL CARD KEYWORD. COMPLETION CODE=16

Explanation: The control card keyword is missing or is invalid as it appears. The entire control card may be missing. Check for a misspelled keyword or a character in column 1.

System Action: The job step is terminated.

User Response: Correct the control card and rerun the job step.

IEH906I INVALID CHARACTER IN USER-INPUT RECORD. COMPLETION CODE=16

Explanation: A character in the user input record cannot be translated into valid internal code (i.e., the character is other than 0-9 or A-F).

System Action: The job step is terminated.

User Response: Examine track of VTOC utility control card for 10 bytes of hexadecimal information or check the input record for an invalid character and rerun the job step.

IEH907I DEVICE DOES NOT HAVE SOFTWARE ASSIGNABLE ALTERNATES. COMPLETION CODE=16

Explanation: The unit type specified in the UCB is other than a 2311, 2302, 2321, or 2314. It has no software assignable alternates.

System Action: The job step is terminated.

User Response: None

IEH908I ALL ALTERNATE TRACKS FOR THIS DEVICE HAVE BEEN ASSIGNED. COMPLETION CODE=16

Explanation: The format 4 DSCB shows that this device has no alternate tracks available for assignment.

System Action: The job step is terminated.

User Response: ANALYZE the pack and return job.

IEH909I REQUESTED STORAGE IS NOT AVAILABLE. COMPLETION CODE=16

Explanation: Necessary core for a work area was not available at the time the GETMAIN was issued.

System Action: The job step is terminated.

User Response: Enlarge the partition or region size and return job.

IEH910I MESSAGE TEXT PROVIDED BY SYNADAF MACRO - I/O ERROR. COMPLETION CODE=16

Explanation: A permanent error was detected while reading the SYSIN data set.

System Action: The job step is terminated.

User Response: Check the DCB parameters on the SYSIN DD card.

IEH911I TRACK ADDRESS PROVIDED DOES NOT BELONG TO DATA SET. COMPLETION CODE=16

Explanation: The address of the record provided on the control card does not belong to the specified data set.

System Action: The job step is terminated.

User Response: Examine the utility control statement to ensure that the cylinder and track address is within the extents of the SYSUT1 data set.

IEH912I INCORRECT NUMBER OF CHARACTERS IN USER-INPUT RECORD. COMPLETION CODE=16

Explanation: Too few or too many data cards are in the input stream. Check for incorrect record length.

System Action: The job step is terminated.

User Response: Check data cards for the accurate number of characters.

IEH913I CONDITION OTHER THAN DATA CHECK OR MISSING ADDRESS MARKER. COMPLETION CODE=16

Explanation: An invalid sense byte indication has been detected for the user's channel program or for another channel program to process data on the bad track. IEHATLAS cannot handle the error condition.

System Action: The job step is terminated.

User Response: Error condition is unrecoverable with IEHATLAS.

IEH914I FORMAT 4 DSCB CANNOT BE READ. COMPLETION CODE=16

Explanation: A permanent I/O error was detected when reading the format 4 DSCB. Information concering the number of alternates available or the address of the next available alternate cannot be retrieved.

System Action: The job step is terminated.

User Response: Re-initialize the volume using either IEHDASDR or IBCDASDI.

IEH915I RECORD IN ERROR IS FORMAT 4 DSCB. COMPLETION CODE=16

Explanation: The utility did not successfully rewrite the user's record which is the format 4 DSCB. No alternate track information is available.

System Action: The job step is terminated.

User Response: Re-initialize the volume using either IEHDASDR or IBCDASDI.

IEH916I ERROR FOUND IN COUNT FIELD OF LAST RECORD ON TRACK. COMPLETION CODE=16

Explanation: Count field information cannot be recovered for the last record on a track unless that record is the error record input to the utility or the CCHHRKDD has been passed to the ATLAS SVC (86). IEHATLAS also requires information regarding track overflow records.

System Action: The job step is terminated.

User Response: Either IEHATLAS or the ATLAS SVC (86) should be given as input to the CCHHRKDD of the last record. Track overflow information is also required if the last record is part of a track overflow data set.

IEH917I HA OR RO ERRORS. COMPLETION CODE=16

Explanation: The ATLAS SVC (86) will not accept an R0 error record unless the SVC has been entered via the utility. An I/O error in HA or R0 prevents further use of the track on which the error exists.

System Action: The job step is terminated.

User Response: Re-initialize the volume using either IEHDASDR or IBCDASDI.

IEH918I ERROR/ERRORS ENCOUNTERED ALTERNATE ASSIGNED. COMPLETION CODE=16

Explanation: An error or errors was/were encountered while transfering the data from the bad track to the alternate. Such a condition will not prevent assignment of an alternate.

System Action: The job step is terminated.

User Response: A DA dump should be taken to check data validity on the alternate.

IEH919I ALTERNATE TRACK ASSIGNED. I/O ERROR IN RE-EXECUTING USER CHANNEL PROGRAM COMPLETION CODE=16

Explanation: An alternate track has been assigned but because the user's channel program could not be re-executed, the error condition still exists for the original record in error.

System Action: The job step is terminated.

User Response: A DA dump should be taken to check data validity on the alternate and IEHATLAS used, if necessary, to update in place the defective record, if the user's channel program cannot be successfully re-executed.

IEH920I THE SYSTEM DOES NOT SUPPORT TRACK OVERFLOW. COMPLETION CODE=16

Explanation: Track overflow support was not included in the system at system generation time.

System Action: The job step is terminated.

User Response: The user has a track overflow indication in the DCB but the UCB indicates that the device does not support track overflow.

IEH921I NO ERROR IN SPECIFIED VTOC RECORD. COMPLETION CODE=00

Explanation: User's VTOC record has been read without error. No alternate track has been assigned.

System Action: The job step is terminated.

User Response: None

IEH922I ERROR/ERRORS ENCOUNTERED CANNOT BE HANDLED. NO ALTERNATE ASSIGNED. COMPLETION CODE=16

Explanation: The conditions which produce this message are: (1) Count field errors on more than three records. (2) Error on an EOF record when it is not the record specified by the utility. (3) An error in the KDD of the count field of a record other than the specified error record.

System Action: The job step is terminated.

User Response: Check the return parameter list to determine the record numbers of error records. The last record indicated in the list is the record which caused the unrecoverable condition. If three error records have been listed, then the possiblity exists that a fourth read count error was also encountered.

IEH923I NO ERRORS FOUND ON TRACK. NO ALTERANTE ASSIGNED. COMPLETION CODE=16

Explanation: The ATLAS SVC successfully read the indicated error track and therefore did not assign an alternate.

System Action: The job step is terminated.

User Response: The cylinder and track address passed to the ATLAS SVC must specify a track containing an error record.

IFCDIP00 Program

IFC001I D=ddd N=x F=trck* L=trck* S=recd**

Explanation: Produced by the IFCDIP00 system utility program during initialization of the SYS1.LOGREC data set, this message describes the limits of the data set.

In the message text, ddd is the type of device containing the SYS1.LOGREC data set; x is the number, in decimal, of uniquely addressable input/output devices in the system; trck in F=trck is the address of the first track of the extent; trck in L=trck is the address of the last track of the extent; and recd is the starting address of the record entry area within the data set. One asterisk indicates that hexadecimal representation causes 8-character printout. Two asterisks indicate that hexadecimal representation causes 10-character printout.

Operator Response: Retain this message; the information given in this message should be used as input parameters to a later execution of IFCEREP0 system utility program or to a re-exectuion of the IFCDIP00 program.

IFC002I INVALID INPUT

Explanation: The EXEC statement requesting execution of the IFCDIP00 system utility program or the following DD statement contains an error:

- In the EXEC statement, the number of uniquely addressable input/output devices was specified as zero.
- In the EXEC statement, the device type for the system residence volume is wrong.
- In the EXEC statement, the PARM parameter did not contain five characters.
- In the DD statement, the data definition name in the name field was misspelled.

Operator Response: Correct the EXEC or DD statement. (The formats of the statements are given in the publication IBM System/360 Operating System: System Generation, GC28-6554, in the discussion on reinitialization of the SYS1.LOGREC data set. The correct values for the parameters of the EXEC statement were given in message IFC001I.) Then execute the IFCDIP00 program again.

IFC003I I/O ERRORS IN FORMATTING DISK

Explanation: While the IFCDIP00 system utility program was formatting the SYS1.LOGREC data set, an uncorrectable input/output error occurred.

System Action: Execution of the IFCDIP00 program is terminated.

Operator Response: Execute the IFCDIP00 program again. If the error persists, call a customer engineer.

IFC004I END OF DATA SET BEFORE PROGRAM COMPLETE

Explanation: During formatting of the SYS1.LOGREC data set, the IFCDIP00 system utility program found that the data set is not large enough for the parameters specified in the EXEC statement.

Probably, the number of tracks previously allocated to the SYS1.LOGREC data set is too small for the number of uniquely addressable input/output devices specified in the EXEC statement.

Operator Response: Correct the EXEC statement. (The format of the EXEC statement is given in the publication IBM System/360 Operating System: System Generation, GC28-6554, in the discussion on reinitialization of the SYS1.LOGREC data set. The correct values for the parameters of the EXEC statement were given in message IFC001I.) Then execute the IFCDIP00 program again.

The IFCEREP0 Program

IFC005I I/O ERROR ON OUTPUT DEVICE

Explanation: A permanent error has occurred on the output device.

System Action: The program is terminated. (No return code is provided.)

IFC006I HEADER RECORD INCORRECT

Explanation: A validity check of the SYS1.LOGREC data set has uncovered an error in the header record.

System Action: If possible, the IFCEREP0 program resumes processing. The program does not clear selected records to zeros in the SYS1.LOGREC data set. If the program is unable to resume processing, it is terminated. (No return code is provided.)

User Response: Execute the IFCDIP00 program to initialize the SYS1.LOGREC data set.

IFC007I END OF DATA SET BEFORE PROGRAM COMPLETE

Explanation: The IFCEREP0 program referred to a disk address that was not within the SYS1.LOGREC data set.

System Action: The program is terminated. (No return code is provided.)

User Response: Execute the IFCDIP00 program to initialize the SYS1.LOGREC data set.

IFC008I READ DISK FAILURE

Explanation: An uncorrectable input/output failure has occurred while reading a record from the SYS1.LOGREC data set. If the record is the header record, message IFC006I is also printed.

System Action: The record that was being read when the failure occurred is skipped. The program attempts to resume processing with the next record.

IFC009I WRITE DISK FAILURE

Explanation: An uncorrectable input/output failure has occurred while the program was attempting to clear a record to zeros.

System Action: Records are no longer cleared to zeros in the SYS1.LOGREC data set. However, the remaining records are processed.

IFC00AI PARAMETER FIELD ERROR

Explanation: The PARM parameter in the EXEC statement was written incorrectly.

System Action: The program is terminated. (No return code is provided.)

User Response: Correct the PARM parameter and rerun the job.

IFC00B2 STAT RECORD KEY DIFFERS FROM THE EXPECTED -- EXPCD xxx REC xxx

Explanation: A statistical count record is out of sequence. The message identifies the expected record (EXPCD xxx) and the received record (REC xxx).

System Action: Processing continues.

IFC00CI INPUT NOT ACCUMLATIVE DATA SET

Explanation: The input data set is specified as being a history data set; however, it is not a history data set.

System Action: The job step is terminated.

User Response: Mount the correct volume and rerun the job step.

IFC00DI ACCUMULATION OUTPUT TERMINATED - ERROR

Explanation: An unrecoverable write error occurred while writing into an accumulated data set.

System Action: No additional accumulation is performed. The job step is terminated unless an additional function or functions (e.g., summarization or full record printing) are specified.

IFC00EI JOB TERMINATED DUE TO INPUT ERRORS

Explanation: Unrecoverable I/O errors occurred while reading an input history data set. This message is generated when at least 40 records have been read and the number of input errors exceeds 12.5% of the total number of records read.

System Action: The job step is terminated.

IFC00FI ddname DD STATEMENT INCORRECT STEP TERM

Explanation: The named DD statement is coded incorrectly or is missing from the input stream.

System Action: The job step is terminated.

Index

Indexes to systems reference library manuals are consolidated in the publication IBM System/360 Operating System: Systems Reference Library Master Index, GC28-6644. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

System/360 OS Utilities (S360-32) Printed in U.S.A. GC28-6586-11

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

IBM System/360 Operating System:
Utilities

Order No. GC28-6586-11

- Is the material:

	Yes	No
Easy to read?	☐	☐
Well organized?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Accurate?	☐	☐
Suitable for its intended audience?	☐	☐

- How did you use this publication?
  - ☐ As an introduction to the subject
  - ☐ For additional knowledge

  Other .....

- Please check the items that describe your position:
  - ☐ Customer personnel
  - ☐ IBM personnel
  - ☐ Manager
  - ☐ Systems Analyst
  - ☐ Operator
  - ☐ Programmer
  - ☐ Customer Engineer
  - ☐ Instructor
  - ☐ Sales Representative
  - ☐ Systems Engineer
  - ☐ Trainee

  Other .....

- Please check specific criticism(s), give page number(s), and explain below:
  - ☐ Clarification on page(s) .....
  - ☐ Addition on page(s) .....
  - ☐ Deletion on page(s) .....
  - ☐ Error on page(s) .....

Explanation:

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Note: 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 Fold

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY ...

IBM Corporation
P.O. Box 390
Poughkeepsie, N.Y. 12602

Attention: Programming Systems Publications
Department D58

Fold Fold

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

Cut Along Line