

Chapter
2

**IEBGENER:
SEQUENTIAL COPY/
GENERATE DATA SET
PROGRAM**

*Get on the
Fast Track!*



TM

**SYS-ED/
COMPUTER
EDUCATION
TECHNIQUES, INC.**

Objectives

You will learn:

- C Input and output.
- C IEBGENER job control statements.
- C Utility control statements.
- C GENERATE statement.
- C MEMBER statement.
- C RECORD statement.
- C Printing a sequential data set.
- C Create a partitioned data set from sequential input.
- C Convert sequential input into partitioned members.
- C Copying a data set.
- C Copying one or more members in a PDS.
- C Printing a data set.
- C Rebooting a data set.
- C Selecting records to be copied.

1 Purpose and Function

IEBGENER is used to:

- C Create a backup copy of a sequential data set or a member of a partitioned data set or PDSE.
- C Produce a partitioned data set or PDSE, or a member of a partitioned data set or PDSE, from a sequential data set.
- C Expand an existing partitioned data set or PDSE by creating partitioned members and merging them into the existing data set.
- C Produce an edited sequential or partitioned data set or PDSE.
- C Manipulate data sets containing double-byte character set data.
- C Print sequential data sets or members of partitioned data sets or PDSEs.
- C Reblock or change the logical record length of a data set.

1.1 Input and Output

IEBGENER uses the following input:

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

IEBGENER produces the following output:

- C An output data set, which can be either sequential or partitioned.

2 IEBGENER Job Control Statements

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

2.1 SYSUT1 DD Statement

The input data set for IEBGENER, as specified in SYSUT1, can contain fixed, variable, undefined, or variable spanned records.

Concatenated data sets cannot be used with unlike attributes except for block size or device type as input to IEBGENER.

The default record format is undefined (U) for the input data set. Record format must be specified if the data set is new, undefined, or a dummy data set.

2.2 SYSUT2 DD Statement

The output data set for IEBGENER, as specified in SYSUT2, can contain fixed, variable, undefined, or variable spanned records (except partitioned data sets or PDSEs, which cannot contain variable spanned records).

These records can be reblocked by the specification of a new maximum block length on the SYSUT2 DD statement.

When the data set is on DASD and record format and logical record length are not specified in the JCL for the output data set, or no data class is assigned to the output data set, values for each are copied from the input data set.

The output logical record length must be specified when editing is to be performed and the record format is fixed blocked, variable spanned or variable blocked spanned. Logical record length must also be specified when the data set is new, or a dummy data set, or from a printer.

3 Utility Control Statements

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

When the output is to be sequential and editing is to be performed, one GENERATE statement and as many RECORD statements as required are used. If you are providing exit routines, an EXITS statement is required.

When the output is to be partitioned, one GENERATE statement, one MEMBER statement per output member, and RECORD statements, as required, are used. If you are providing exit routines, an EXITS statement is required.

A continuation line must start in columns 4 to 16, and a nonblank continuation line in column 72 is optional.

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

3.1 GENERATE Statement

The GENERATE statement is required when:

C	output is to be partitioned.	C	editing is to be performed.
C	user routines are provided.	C	label processing is specified.

The GENERATE statement must appear before any other IEBGENER utility statements.

label	GENERATE	[,MAXNAME=n] [,MAXFLDS=n] [,MAXGPS=n] [,MAXLITS=n] [,DBCS={YES NO}]
where:		
MAXNAME=n		Specifies a number, from 1 to 3276, that is greater than or equal to the total number of member names and aliases appearing in subsequent MEMBER statements. MAXNAME is required if there are one or more MEMBER statements.
MAXFLDS=n		Specifies a number, from 1 to 4095, that is greater than or equal to 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, from 1 to 2520, that is greater than or equal to the total number of IDENT or IDENTG parameters appearing in subsequent RECORD statements. MAXGPS is required if there are any IDENT or IDENTG parameters in subsequent RECORD statements.
MAXLITS=n		Specifies a number, from 1 to 2730, that is greater than or equal to the total number of characters contained in the FIELD literals of subsequent RECORD statements. Any DBCS characters used as literals on FIELD parameters count as two characters each.
MAXLITS		Is required if the FIELD parameters of subsequent RECORD statements contain literals. MAXLITS does not apply to literals used in IDENT or IDENTG parameters.

3.2 MEMBER Statement

The MEMBER statement is used when the output data set is to be partitioned. One MEMBER statement must be included for each member to be created by IEBGENER.

The MEMBER statement provides the name and alias names of a new member.

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

[label]	MEMBER	NAME=(name[,alias 1][,alias 2][,...])
where:		
NAME=(name[,alias][,...])		Specifies a member name followed by a list of its aliases. If no aliases are specified in the statement, the member name need not be enclosed in parentheses.

3.3 RECORD Statement

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

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

If no RECORD statement is used, the entire input data set or member is processed without editing.

More than one RECORD statement may appear in the control statement stream for IEBGENER.

Syntax:

[label]	RECORD	<pre> {{IDENT IDENTG}=(length,'name',input-location) [,FIELD=([length],[input-location 'literal']], conversion],[output-location])} [,FIELD=...] [,LABELS=n] </pre>
where:		
	<pre> {{IDENT IDENTG}=(length,'name',input-location) </pre>	<p>Identifies the last record of a collection of records in the input data set. This parameter is used to identify the last record to be edited according to the FIELD parameters on the same RECORD statement. If a partitioned data set or PDSE is being created, this parameter will identify the last record to be included in the partitioned data set or PDSE member named in the previous MEMBER statement.</p> <p>IDENT is used to identify a standard, single-byte character string.</p> <p>IDENTG is used to identify a double-byte character string.</p> <p>The values for IDENT or IDENTG can be coded:</p> <p>length Specifies the length (in bytes) of the identifying name. The length of the identifier cannot be greater than eight.</p> <p>For IDENTG, the length must be an even number.</p> <p>'name' Specifies the literal that identifies the last input record of a group of records. 'Name' must be coded within single apostrophes.</p>
	<pre> FIELD=([length], [input-location 'literal']], [conversion], [output-location]) </pre>	<p>Specifies field-processing and editing information. Only the contents of specified fields in the input record are copied to the output record; that is, any field in the output record that is not specified will contain meaningless data.</p> <p>The variables on the FIELD parameter are positional; if any of the options are not coded, the associated comma preceding that variable must be coded.</p>

length	Specifies the length (in bytes) of the input field or literal to be processed. If length is not specified, a length of 80 is assumed. If a literal is to be processed, a length of 40 or less must be specified.
input-location	Specifies the starting position of the field to be processed. Input-location should be coded as a whole decimal number. If input-location is not specified, it defaults to 1.
'literal'	Specifies a literal (maximum length of 40 bytes) to be placed in the specified output location. If a literal contains apostrophes, each apostrophe must be written as two consecutive apostrophes.
conversion	<p>A literal can be specified in hexadecimal by coding X'literal'. A double-byte character string can set as the literal.</p> <p>The values that can be coded are:</p> <p>CG Specifies that shift-out/shift-in characters are to be removed, but that DBCS data is not to be validated. DBCS=YES must be specified on the GENERATE statement.</p> <p>CV Specifies that DBCS data is to be validated, and that the input records contain single-byte character set data as well as double-byte. DBCS=YES must be specified on the GENERATE statement.</p> <p>GC Specifies that shift-out/shift-in characters are to be inserted to enclose the DBCS data. DBCS=YES must be specified on the GENERATE statement.</p> <p>GV Specifies that DBCS data is to be validated, and that the DBCS data is not enclosed by shift-out/shift-in characters. DBCS=YES must be specified on the GENERATE statement.</p> <p>HE Specifies that H-set BCDIC data is to be converted to EBCDIC.</p> <p>PZ Specifies that packed decimal data is to be converted to unpacked decimal data. Unpacking of the low-order digit and sign may result in an alphabetic character. You cannot unpack packed decimal data contained in records longer than 16K bytes.</p> <p>VC Specifies that DBCS data is to be validated, and that shift-out/shift-in characters are to be inserted to enclose the DBCS data. DBCS=YES must be specified on the GENERATE statement.</p> <p>VG Specifies that DBCS data is to be validated, and that shift-out/shift-in characters are to be eliminated from the records. DBCS=YES must be specified on the GENERATE statement.</p>

	<p>ZP Specifies that unpacked decimal data is to be converted to packed decimal data. 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 before the use of the ZP parameter.</p>
output-location	<p>Specifies the starting location of this field in the output records. Output-location should be coded as a whole decimal number.</p> <p>If output-location is not specified, the location defaults to 1.</p>
LABELS=n	<p>Is an optional parameter that indicates the number of records in the SYSIN data set to be treated as user labels.</p>

4 Walkthroughs/Examples

4.1 Printing a Sequential Data Set

```
//PRINT JOB ...
//STEP1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT1 DD DSN=DS80.DATA,DISP=SHR,
//SYSUT2 DD SYSOUT=A
```

4.2 Create a Partitioned Data Set from Sequential Input

```
//TAPEDISK JOB ...
//STEP1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=INSET,UNIT=tape,LABEL=(,SL),
//      DISP=(OLD,KEEP),VOLUME=SER=001234
//SYSUT2 DD DSN=NEWSET,UNIT=disk,DISP=(,KEEP),
//      VOLUME=SER=111112,SPACE=(TRK,(10,5,5)),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000)
//SYSIN DD *
      GENERATE MAXNAME=3,MAXGPS=2
      MEMBER NAME=MEMBER1
GROUP1 RECORD IDENT=(8,'FIRSTMEM',1)
      MEMBER NAME=MEMBER2
GROUP2 RECORD IDENT=(8,'SECNDMEM',1)
      MEMBER NAME=MEMBER3
/*
```

GENERATE indicates that three member names are included in subsequent MEMBER statements and that the IDENT parameter appears twice in subsequent RECORD statements.

- C The first MEMBER statement assigns a member name (MEMBER1) to the first member.
- C The first RECORD statement (GROUP1) identifies the last record to be placed in the first member. The name of this record (FIRSTMEM) appears in the first eight positions of the input record.
- C The remaining MEMBER and RECORD statements define the second and third members.
- C There is no RECORD statement associated with the third MEMBER statement, the remainder of the input file will be loaded as the third member.

4.3 Convert Sequential Input into Partitioned Members

```
//DISKTODK JOB ...
//STEP1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=INSET,UNIT=disk,DISP=(OLD,KEEP),
//      VOLUME=SER=111112,LABEL=(,SUL)
//SYSUT2 DD DSN=EXISTSET,UNIT=disk,DISP=(MOD,KEEP),
//      VOLUME=SER=111113
//SYSIN DD *
      GENERATE MAXNAME=3,MAXGPS=1
      EXITS INHDR=ROUT1,INTLR=ROUT2
      MEMBER NAME=(MEMX,ALIASX)
GROUP1 RECORD IDENT=(8,'FIRSTMEM',1)
      MEMBER NAME=MEMY
/*
```

GENERATE indicates that a maximum of three names and aliases are included in subsequent MEMBER statements and that one IDENT parameter appears in a subsequent RECORD statement.

EXITS defines the user routines that are to process user labels.

- C The first MEMBER statement assigns a member name (MEMX) and an alias (ALIASX) to the first member.
- C The RECORD statement identifies the last record to be placed in the first member. The name of this record (FIRSTMEM) appears in the first eight positions of the input record.
- C The second MEMBER statement assigns a member name (MEMY) to the second member.
- C The remainder of the input data set is included in this member.