JCL

Chapter c3
Using utility programs

Chapter c3

Using utility programs

Unit introduction.

Like procedures, utility programs can help you make better use of the system.

The Utilities Manual provides detailed information on the specific utility programs available with the installation.

This unit emphasizes the use of JCL to communicate with utilities, and how to interpret the messages utilities use to communicate with you.

Course objectives.

Be able to:

- **Use your Utilities Manual to identify utility programs available to accomplish a task.**

- **Identify the JCL statements needed to communicate with selected utilities.**

- **Specify the purpose of utility control statements.**

- **Identify utility control statements that have been coded correctly according to the syntax rules.**

- **Interpret informational and error messages produced by utilities.**

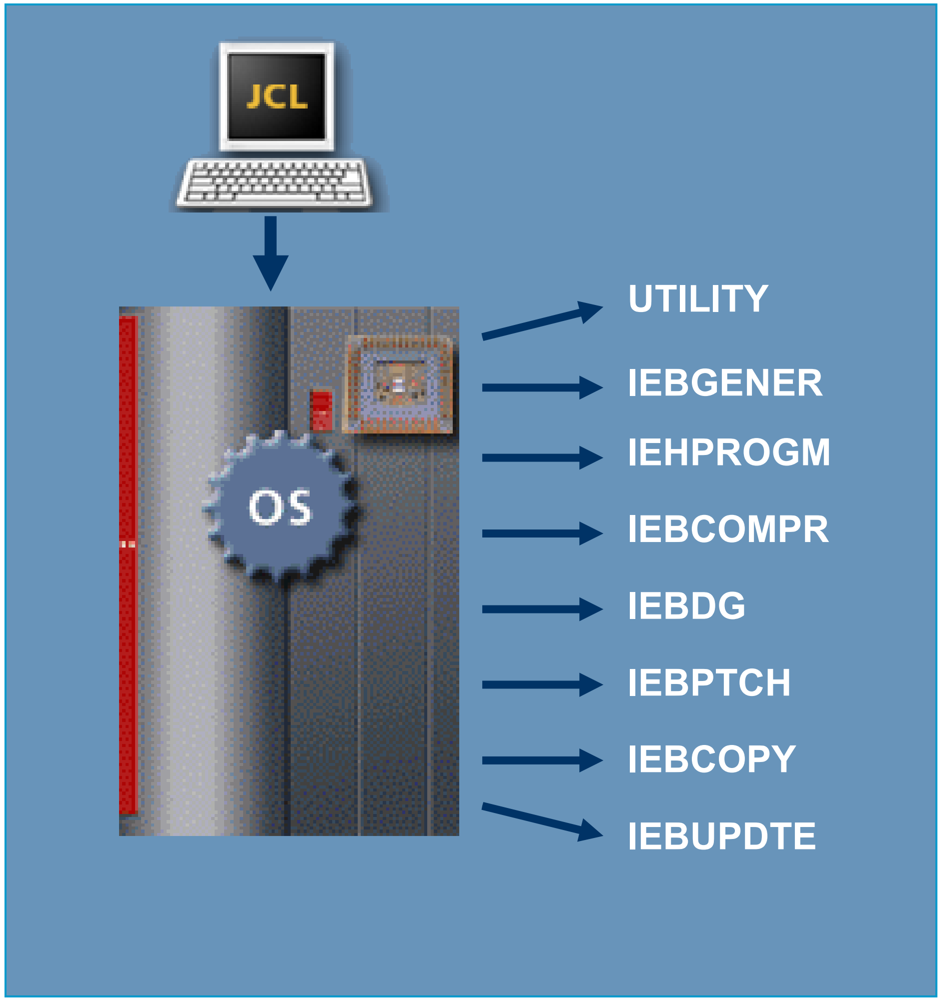
- **Correct control statements that were coded incorrectly.**

Choosing a utility.

What are utility programs?

Utility programs are general purpose programs that are a part of your OS. They are designed to help you reorganize, compare, or change data at the data set or record level.

Utilities have been in use for many years. Today, some of the functions that utilities have provided may be better performed with applications such as ISPF/PDF. However, utilities are still useful to perform functions in a way that will work in all MVS installations.



Communicating with utilities.

Choosing a utility – utilities manual.

Task	Options	Primary Utility	Secondary utility
Add	a password	IEHPROGM	
Alter in Place	a load module	IEBCOPY	
Catalog	a data set in CVOL	IEHPROGM	
Change	data set organization	IEBUPDTE	IEBGENER
			IEBTPCH
	logical record length	IEBGENER	
Compare	partitioned data sets	IEBCOMPR	
	sequential data sets	IEBCOMPR	
	PDSEs	IEBCOMPR	

It is easy to select a utility to meet your processing needs. Your Utilities Manual has a table that lists the tasks performed by each utility. A sample is shown above and continues on the next slide.

Communicating with utilities.

Choosing a utility – utilities manual.

Task	Options	Primary Utility	Secondary utility
Compress	a partitioned data set	IEBCOPY	
Convert to partitioned data set	an unloaded copy of a PDS	IEBCOPY	
	sequential data sets	IEBGENER	IEBUPDTE
	a PDSE	IEBCOPY	
Convert to sequential data set	a partitioned data set	IEBGENER	IEBUPDTE
	an indexed sequential data set	IEBDG	IEBISAM

If more than one utility will accomplish the task you need, you can use the one you prefer.

Communicating with utilities.

Are we on track?

Where are utility programs located?

A. In a procedure library.

B. On a tape volume.

C. Within the operating system.

Communicating with utilities.

Are we on track?

Refer to the tables on the previous pages or to your Utilities Manual. Match the utility with the task or tasks it can perform.

1. IEBGENER	A. Change data set organization.
2. IEHPROGM	B. Compress a partitioned data set.
3. IEBUPDTE	C. Convert a sequential data set to a partitioned data set.
4. IEBCOPY	D. Catalog a data set in CVOL.

Communicating with utilities.

General form for executing utilities.

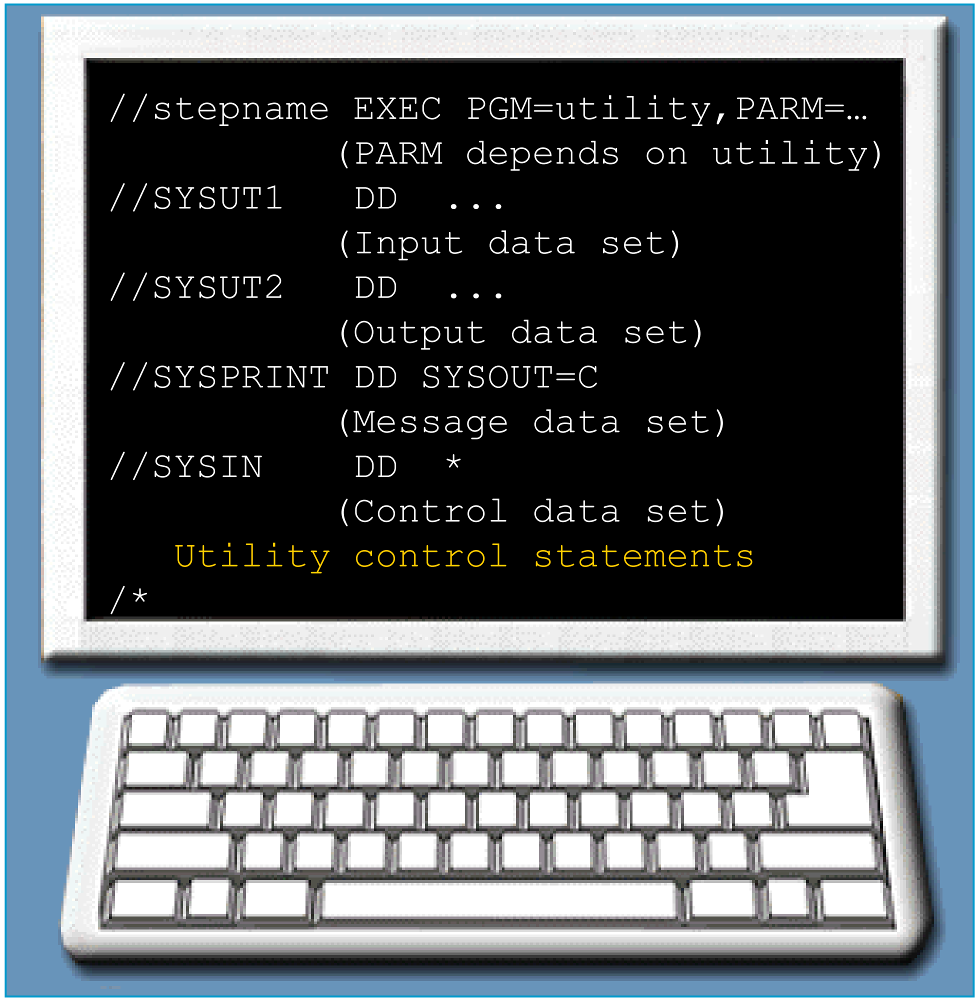
How to execute utility programs?

You execute utility programs with standard JCL statements:

```
//stepname   EXEC PGM=progname
//ddname     DD   parameters
```

A few utilities require PARM information to specify processing requirements. If so, code it on the EXEC statement invoking the utility.

```
//stepname   EXEC PGM=utility,
//                PARM=…
```



```
//stepname EXEC PGM=utility,PARM=…
          (PARM depends on utility)
//SYSUT1   DD  ...
          (Input data set)
//SYSUT2   DD  ...
          (Output data set)
//SYSPRINT DD SYSOUT=C
          (Message data set)
//SYSIN    DD  *
          (Control data set)
   Utility control statements
/*
```

Communicating with utilities.

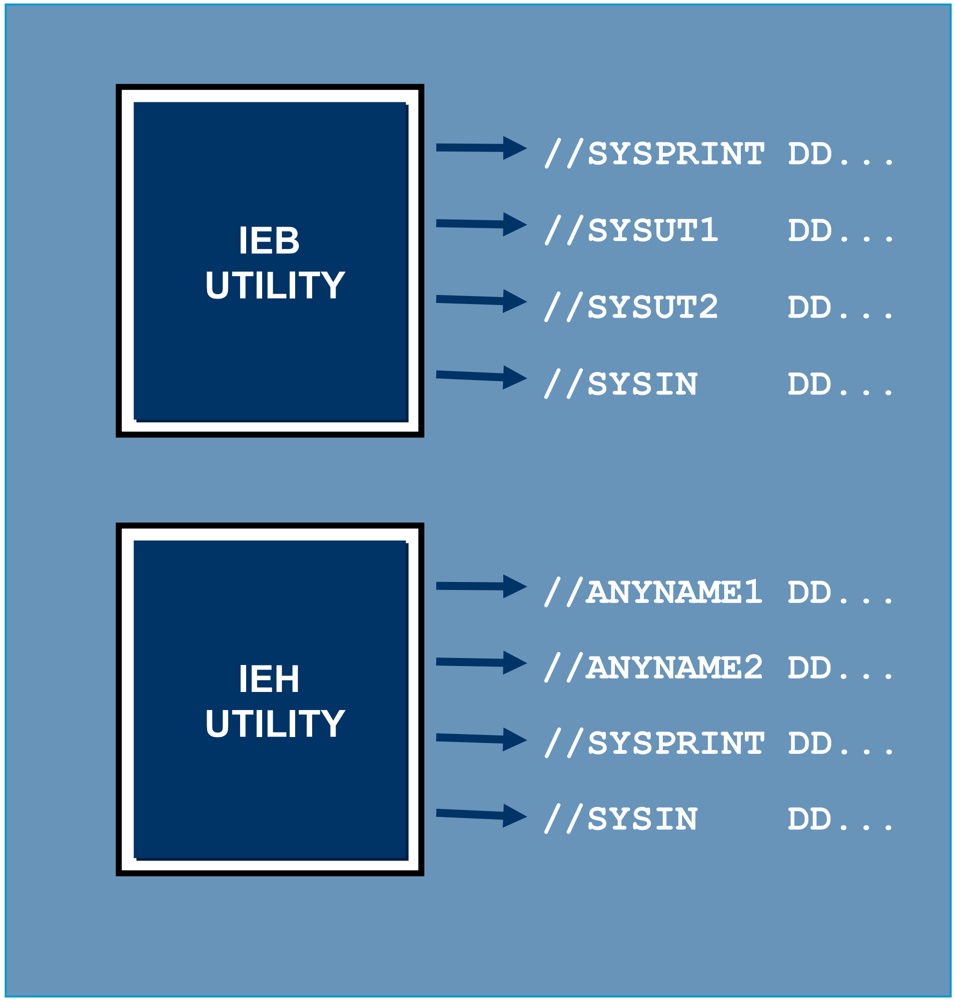
DDnames for utilities.

DDnames for utilities vary with the particular utilities. Two kinds of utilities are described below:

- IEB utility programs use DD statements with the DDnames SYSPRINT, SYSUT1, SYSUT2, and SYSIN.

- IEH utility programs allow you to specify your own DDnames. The actual DDnames have to be specified in utility control statements.

The DD statements can define a sequential data set, PDS, or member of a PDS; depending on the utility and application.



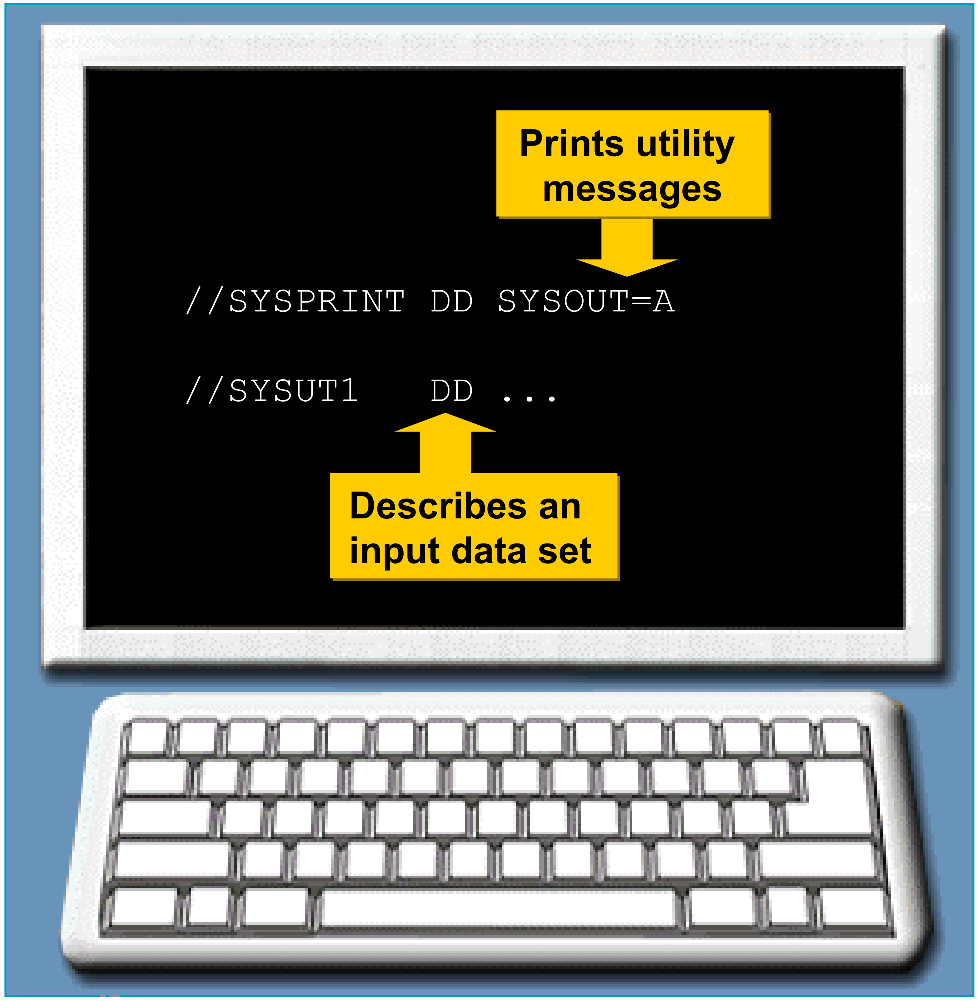
DDnames for IEB utility programs.

IEB utility programs use DD statements with the DDNAMES:

- SYSPRINT - To define a data set where actions and error conditions are reported.

- SYSUT1 - To specify the input data set to be processed, for all IEB utilities.

DCB subparameters may be required on the SYSUT1 DD statement.



```
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD ...
```

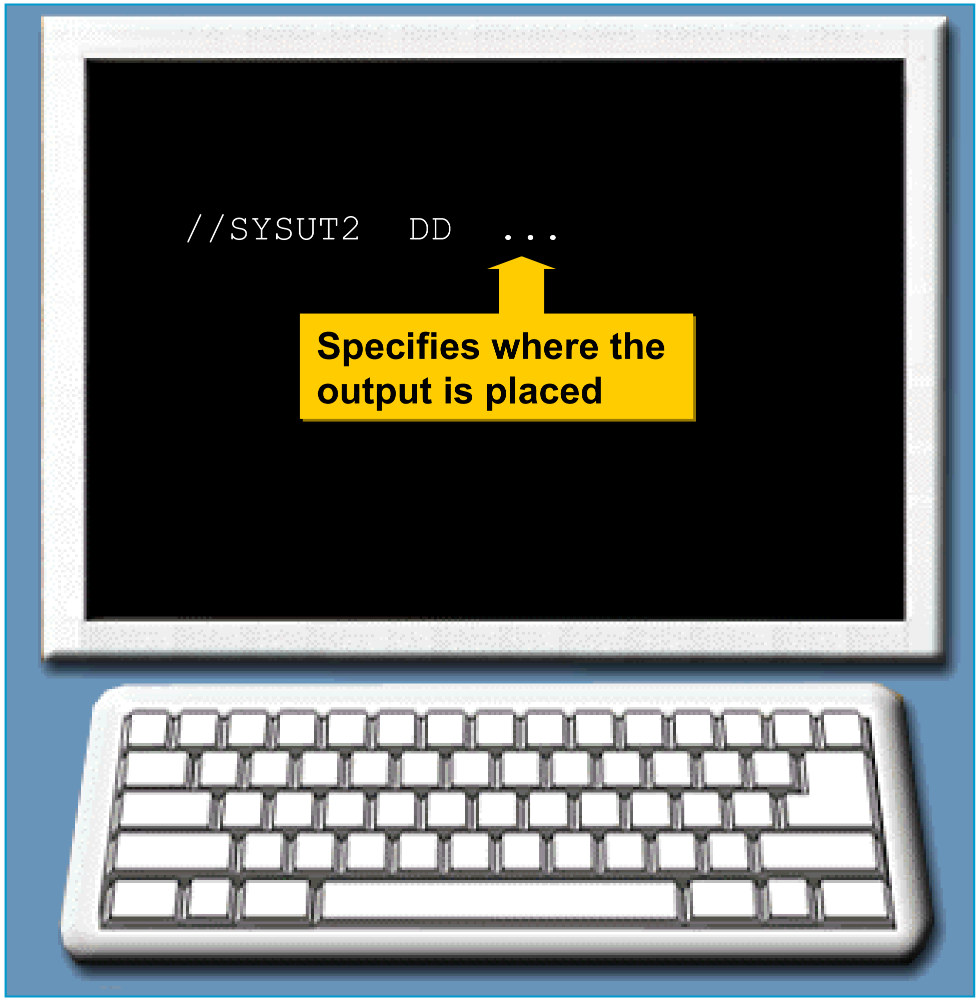
Communicating with utilities.

DDnames for IEB utility programs.

SYSUT2 specifies where the output created by the utility should be placed.

You need to specify output record size, record format, and blocksize as DCB information:

```
//SYSUT2 DD ...,DCB=(LRECL=...,
//          RECFM=...,BLKSIZE=...)
```



```
//SYSUT2  DD  ...
```

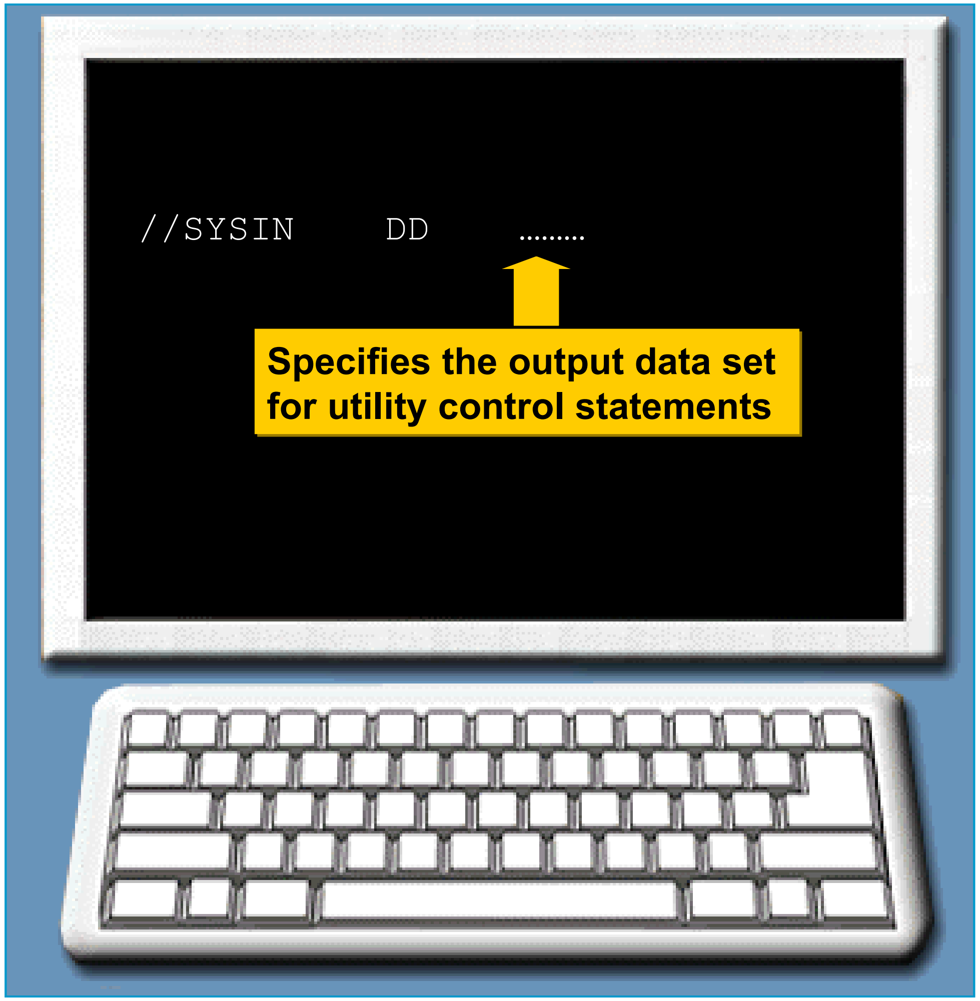
Communicating with utilities.

DDnames for IEB utility programs.

SYSIN defines the control data set, in which the utility control statements you code are placed. Usually, this data set is in the job stream.

If you do not need any control statements, you should code:

```
//SYSIN     DD      DUMMY
```



```
//SYSIN     DD     .........
```

Specifies the output data set for utility control statements

Are we on track?

For IEB utilities, the system by default creates the output data set with the same DCB attributes specified on the ___________ statement.

Are we on track?

Match the JCL DD statement with its function.

1. SYSUT2	A. Defines the control data set.
2. SYSPRINT	B. Defines the output data set of IEB utilities.
3. SYSIN	C. Defines the input data set of IEH utilities.
4. anyname1	D. Defines an output data set where information and error messages are reported.

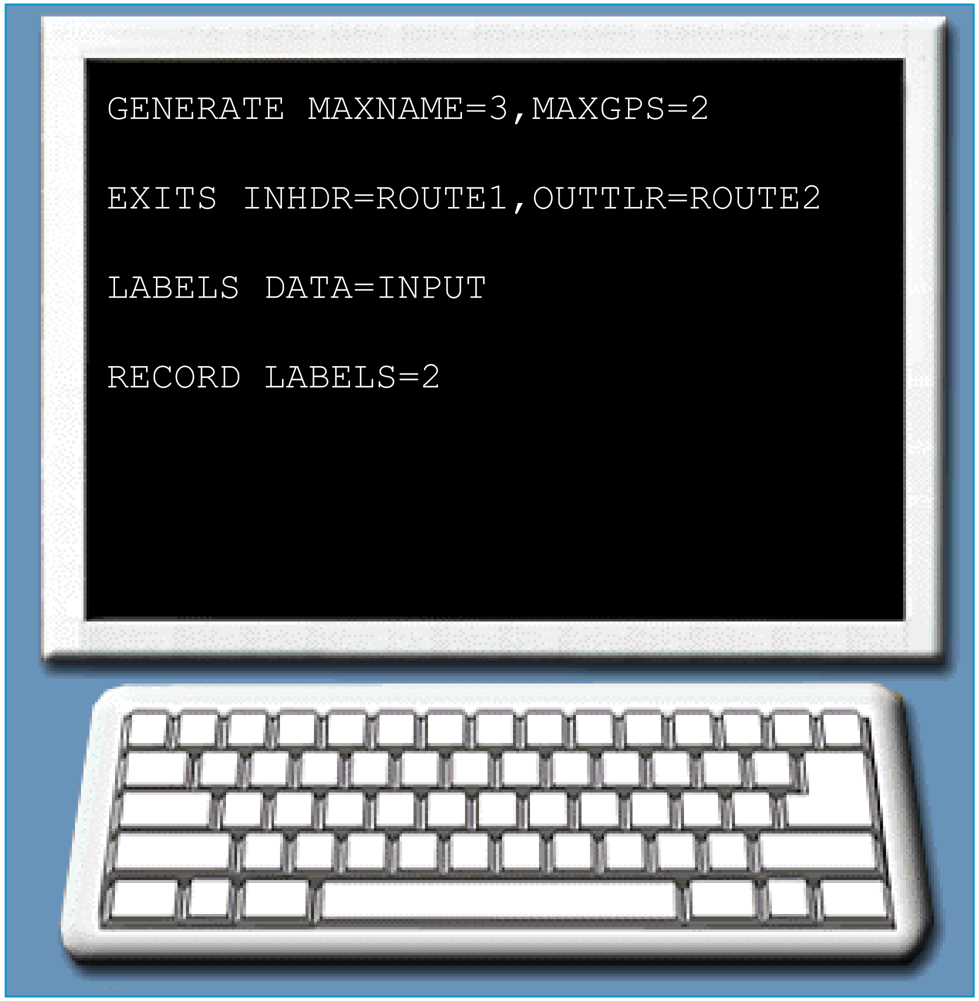
Communicating with utilities.

Utility control statements.

Why code a utility control statement?

Utility control statements are coded to specify to the utility the task you want to perform and, in some cases, the data set to be processed. Each utility has a list of available control statements.

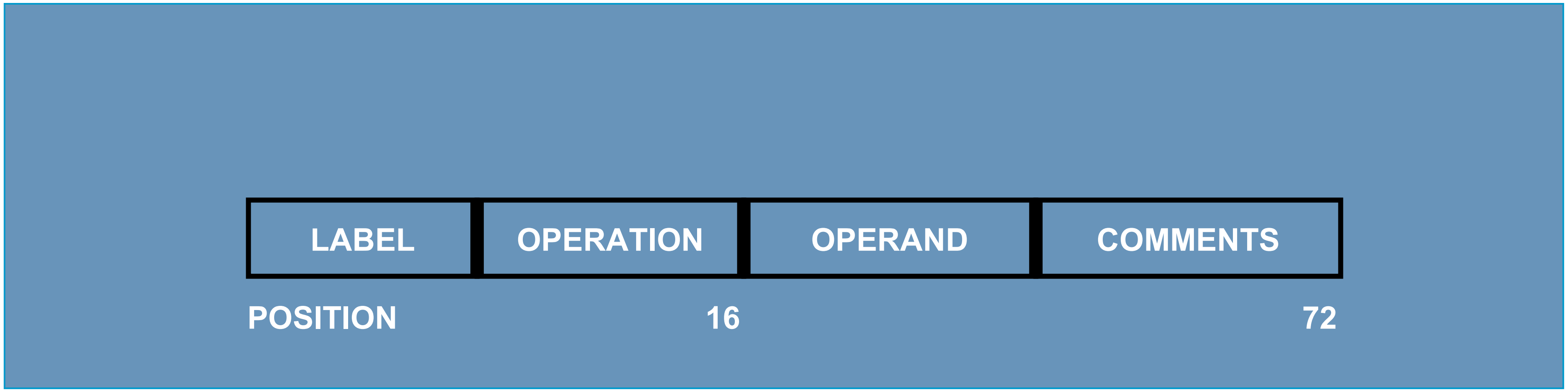
Examples of control statements used with IEBGENER are shown on the right.



```
GENERATE MAXNAME=3,MAXGPS=2

EXITS INHDR=ROUTE1,OUTTLR=ROUTE2

LABELS DATA=INPUT

RECORD LABELS=2
```

Communicating with utilities.

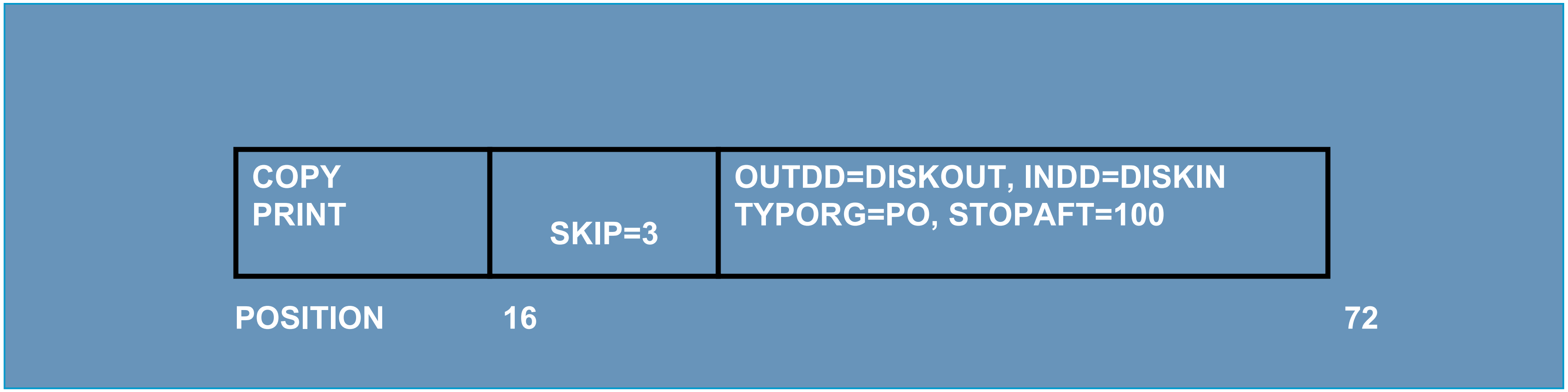
Utility control statements – general format.



The control statements used by all of the utilities (with the exception of IEBUPDTE) have the general format shown above indicating the standard coding positions, where:

- LABEL symbolically identifies the control statement. LABEL is optional in most cases.
- OPERATION identifies the type of control statement.
- The OPERAND is made up of one or more keyword parameters, separated by commas.

Utility control statements – standard coding positions.



The general form for standard coding positions are:

- Control statements are coded as in-stream data in columns 2 through 71.
- To continue a control statement, code a nonblank character in column 72.
- Then following standard coding procedures, you continue the statement in column 16 of the following line.

Notational conventions to code a special DD statements.

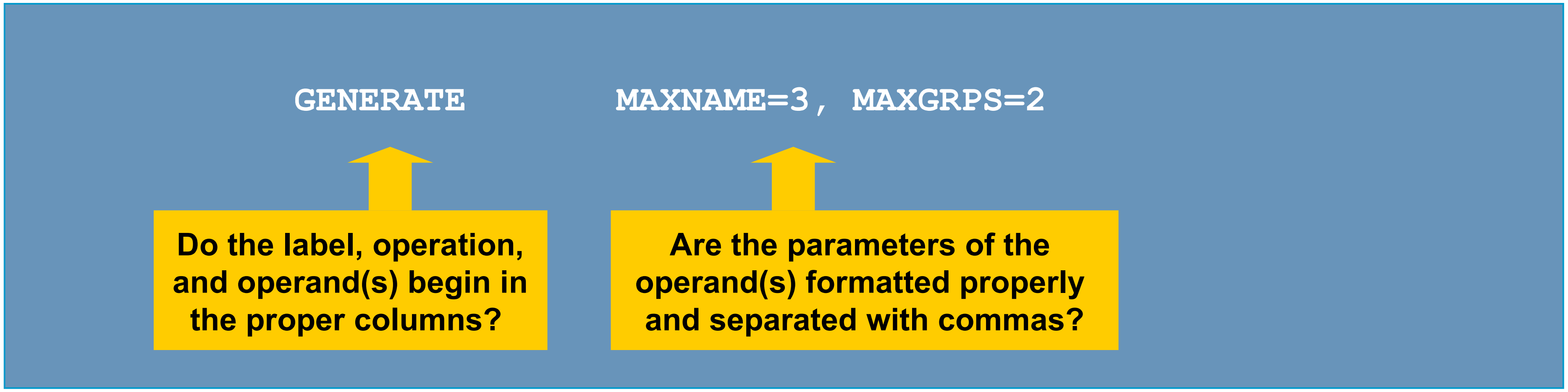
In the Utilities Manual, certain **symbols called notational conventions** indicate whether control statement labels, operands, or sub-operands are necessary or optional.

For example, brackets [ ] are sometimes used to indicate that entry is optional: [label]

The notational conventions to code a special DD statement are as follows:

[ ]	Brackets enclose an optional entry.
|	An OR sign (a vertical bar) separates alternative entries.
{ }	Braces enclose alternative entries. You can only use one of the entries.
"	Quotation marks indicate that a space must be left before the next parameter.

Utility control statements – syntax.



```
         GENERATE          MAXNAME=3, MAXGRPS=2
```

Do the label, operation, and operand(s) begin in the proper columns?

Are the parameters of the operand(s) formatted properly and separated with commas?

At execution, all of the utilities verify that the control statements you supply have valid syntax and content. If there are syntax errors, you should consider the following:

- Do the label, operation, and operand(s) begin in the proper columns?
- Are the continuation statements coded in the proper format?
- Are the parameters of the operand(s) formatted properly and separated with commas?

Communicating with utilities.

Are we on track?

The ____________________ field in a utility control statement defines the type of control statement.

Communicating with utilities.

Are we on track?

Select the statements that are valid for utility control statements.

A. They can specify the task the utility is to perform.

B. They can specify the format of the output.

C. They are coded in JCL.

D. They begin in position 16.

E. They are continued in position 16.

Glossary.

PARM
A parameter on the EXEC statement that passes control information (such as DEBUG) to the job step.

IEB Utility Programs
System utility programs that are used to list or change information related to data sets & volumes.

IEH Utility Programs
Data set utility programs that are used to reorganize, change, or compare data at the data set or record level.

LABEL
A DD statement parameter that contains information on a non-temporary data set, like volume identification.

Kinds of communications.

How do utility programs communicate?

Utility programs communicate with you through condition code settings and utility messages.

What do the messages indicate?

These messages indicate if the utility:

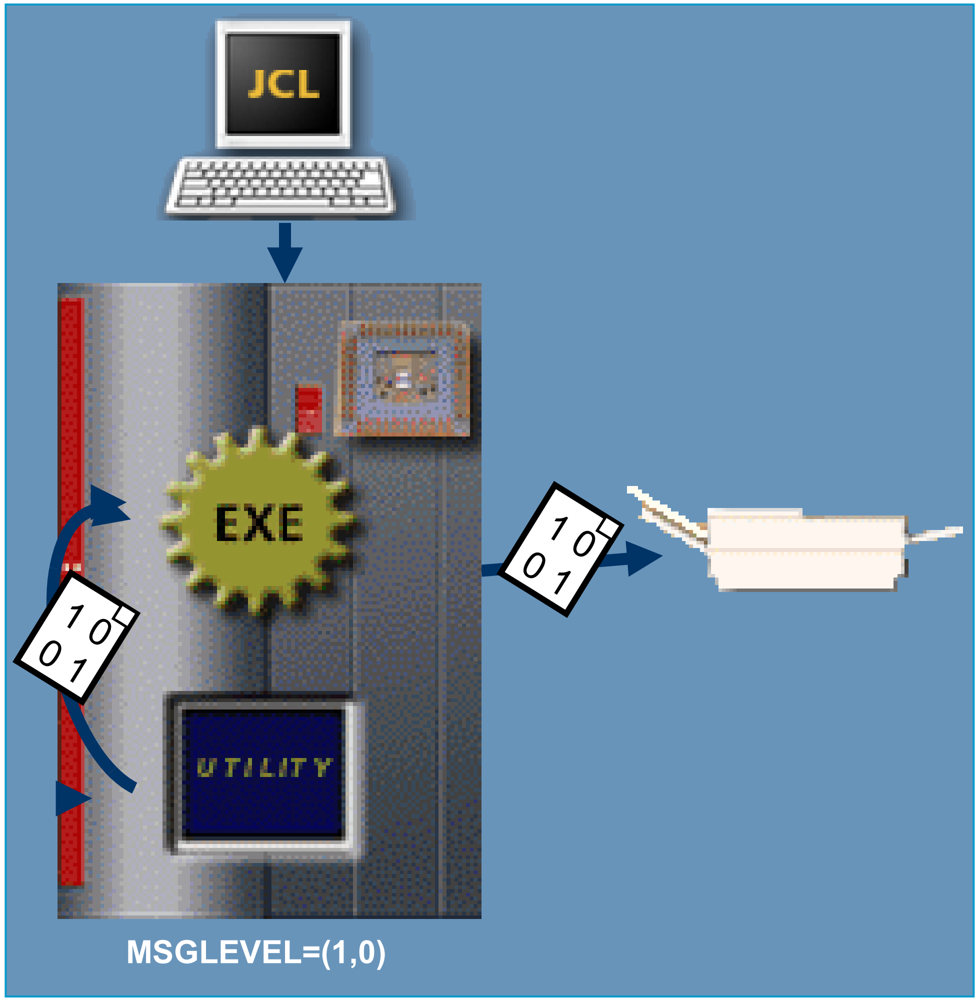
- Understood the request for processing.
- Completed the requested processing successfully.

Condition codes.

What are Condition codes?

Condition codes are produced by the utility as it concludes. They indicate whether the job was successfully completed.

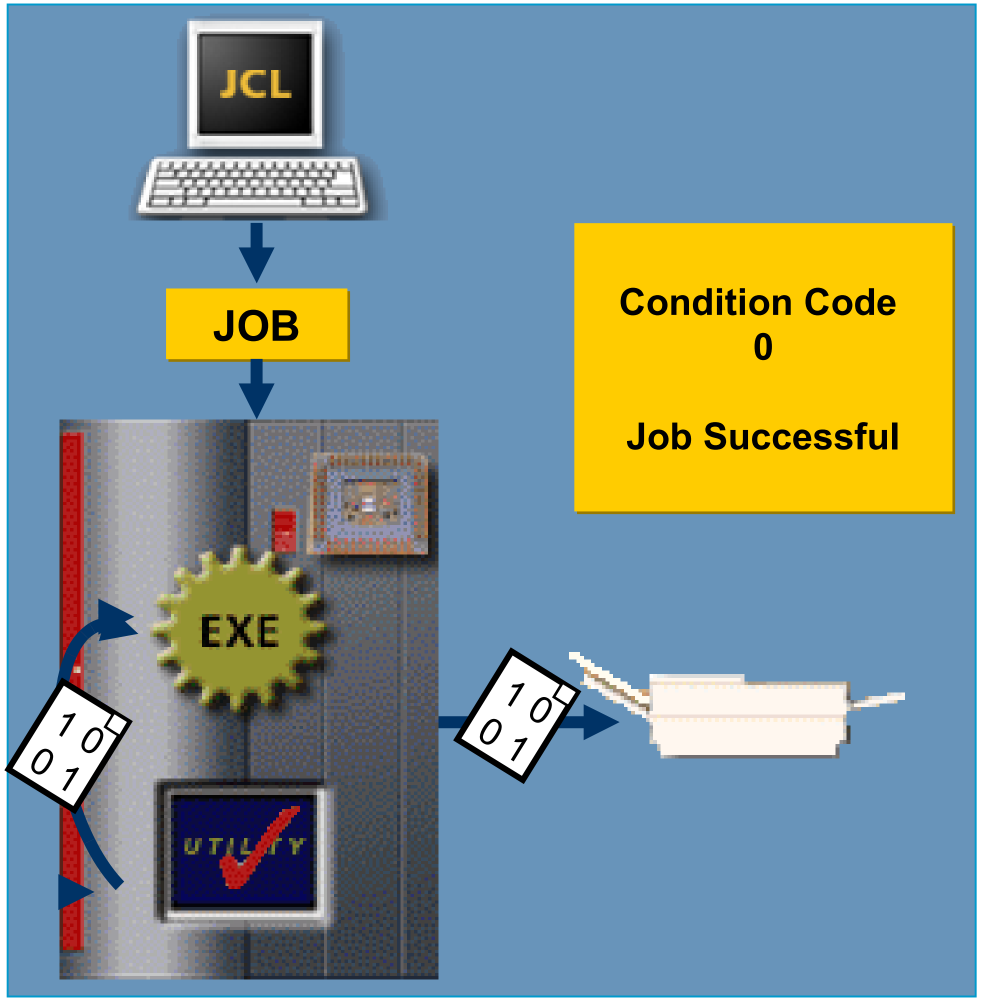
Condition codes are printed in the job log's allocation/termination listing. You print the job log by coding MSGLEVEL=(1,0) on the JOB statement.



Kinds of condition codes – zero condition code.

Zero Condition Code means that the utility detected no errors in the control statement information.

However, this does not necessarily mean that the utility did what you wanted it to do. (It may have assumed inappropriate default values for control statement parameters you did not specify.)



Interpreting utility communications.

Condition codes – sample.

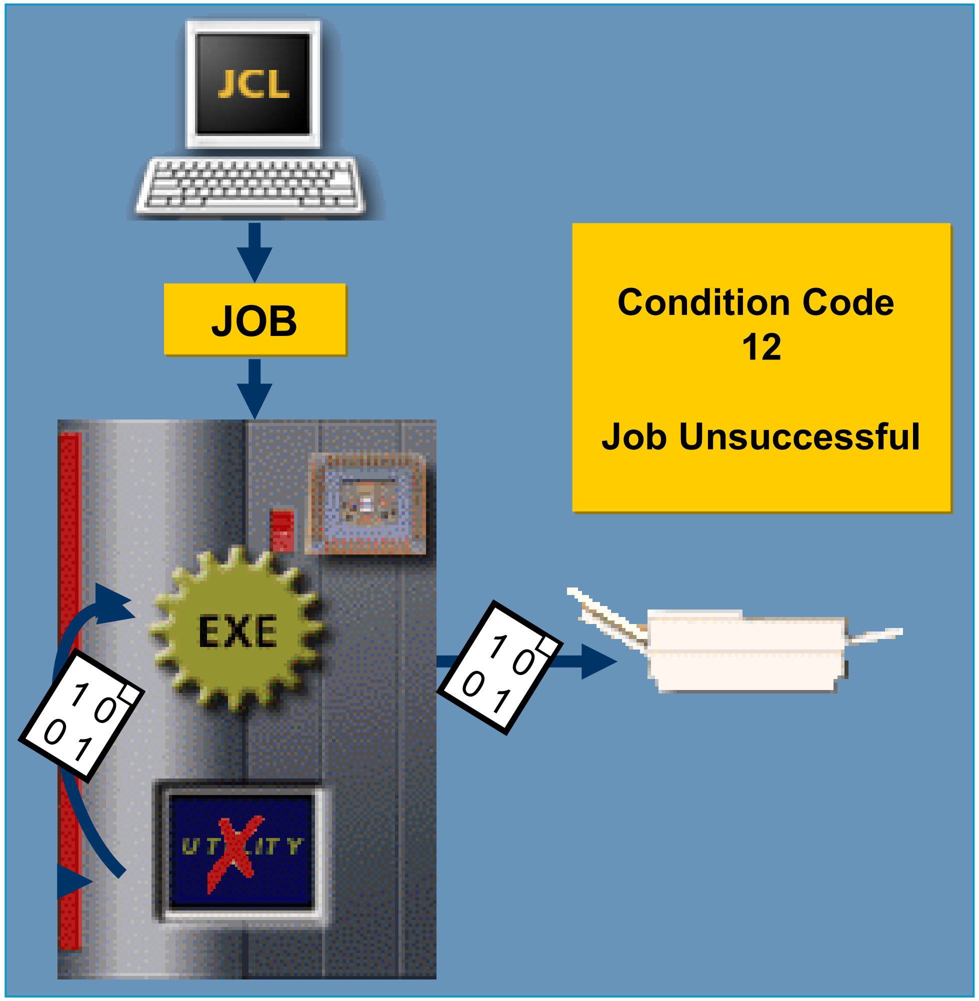
Utility	0	4	8	12	16
IEBGENER	Successful completion.	Warning. Probable completion	Processing ended at user’s request.	Unrecoverable error. Job step terminated.	Job step terminated.
IEBEDIT	Successful completion.	Error condition. Recovery may be possible.	Unrecoverable error.	Not used.	Not used.

The table above shows sample condition codes created by the IEBGENER and IEBEDIT utilities.

Kinds of condition codes – non zero condition code.

Non Zero Condition Code indicates that the utility had difficulty in trying to do the processing you requested.

The meaning of the non-zero condition code varies with the utility that produced it.



Condition codes – an example.

JOB LOG

```
IEF142I     SAMPLE   STEP1 – STEP WAS EXECUTED – COND CODE 0
IEF373I     STEP/STEP1         /START 94342.1134
IEF374I     STEP/STEP1         /STOP 94342.1134 CPU 0 MIN 00.16 SEC SRB …
…
IEF142I     SAMPLE STEP4 –  STEP WAS EXECUTED – COND CODE 12
…
IEF142I     SAMPLE STEP7 –  STEP WAS EXECUTED – COND CODE 4
```

This example shows part of a job allocation/termination listing containing condition codes. The listing indicates the following:

- STEP1 terminated with condition code 0.
- STEP4 terminated with condition code 12.
- STEP7 terminated with condition code 4.

Code 0 indicates that the utility encountered no errors. Code 4 often indicates a warning condition from which recovery may be possible. Code 12 often indicates an unrecoverable error.

Interpreting utility communications.

Are we on track?

Which of the following statements are true of condition codes?

- A. They are printed in the data set defined on the SYSPRINT statement.
- B. They are produced by the utility as it concludes the step.
- C. They are printed in the job log allocation/termination messages.
- D. They can indicate whether the job concluded successfully.
- E. They can identify default values taken by the utility.

Interpreting utility communications.

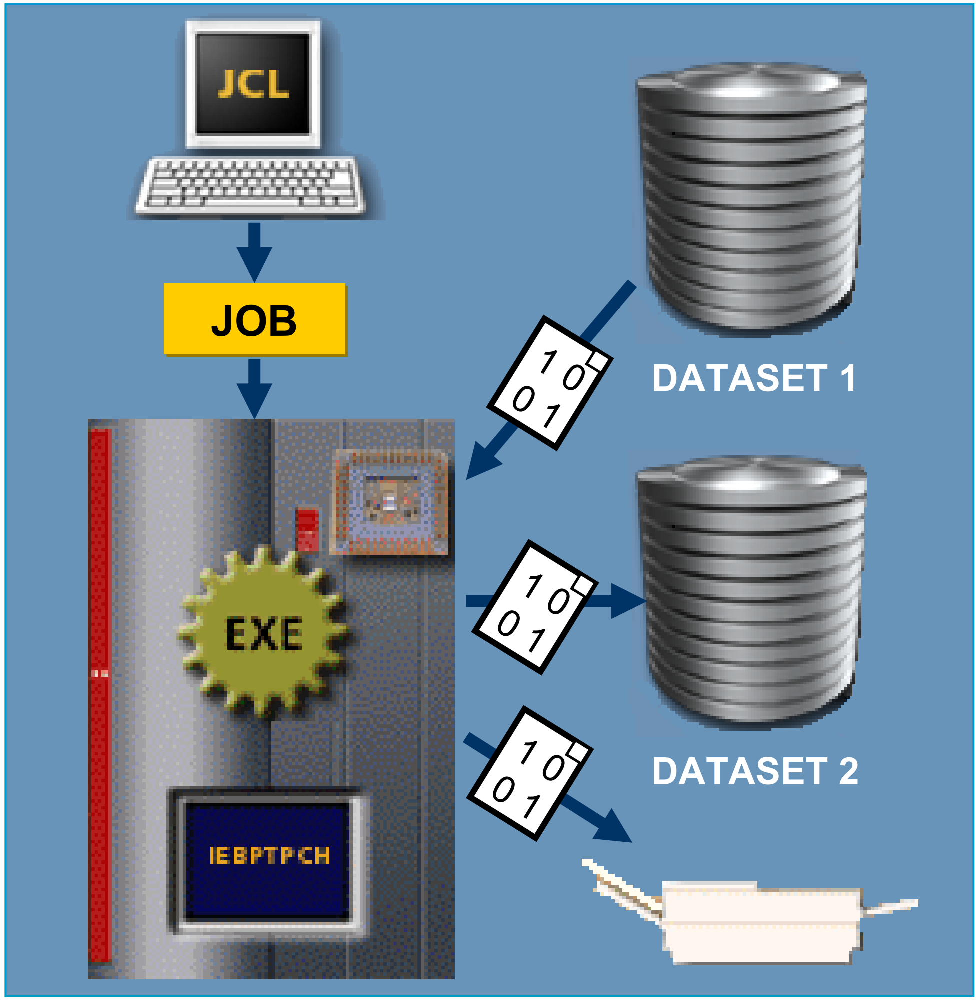
Testing condition codes.

How to test condition codes?

The system tests a condition code when the job is executed if you code a COND parameter on the JOB or EXEC statements.

You can alter your job's processing based on the utility's concluding condition code.

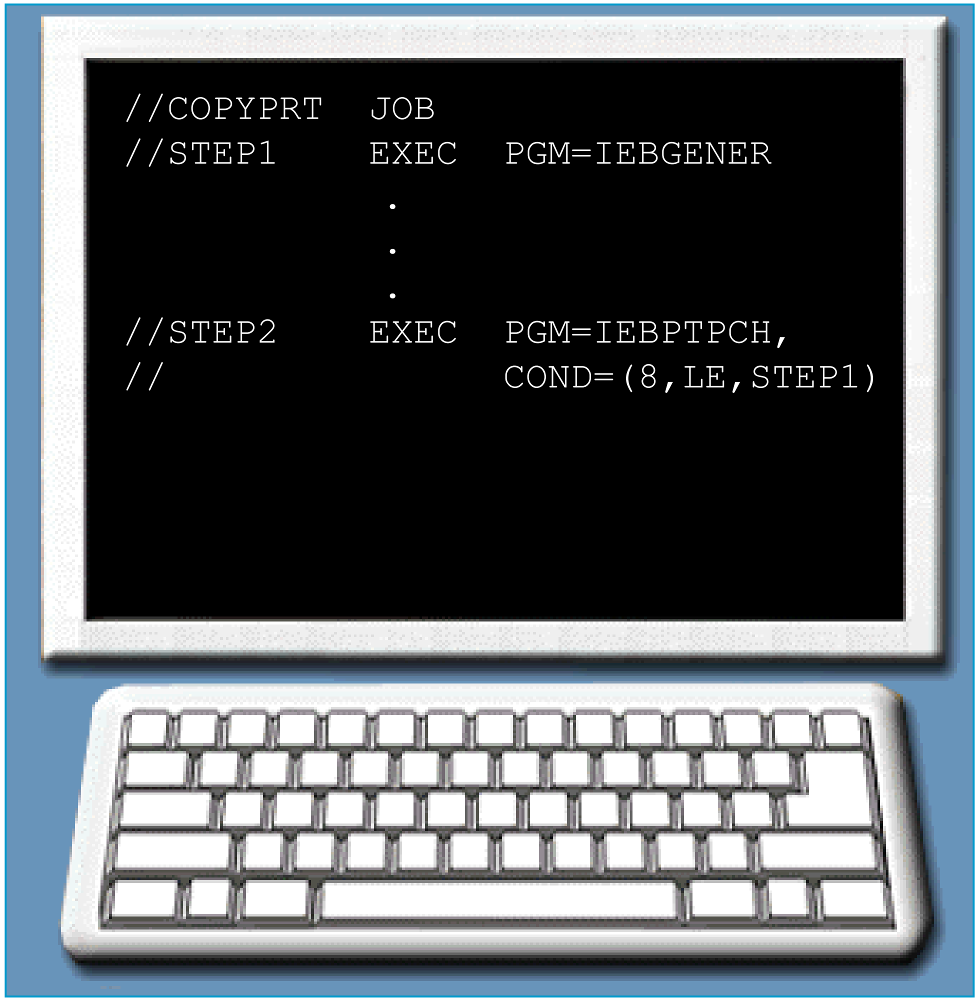
For example, suppose you want to copy a sequential data set to a new sequential data set (using the IEBGENER utility). Then, if the copy is successful, you want to print the new data set (using the IEBPTPCH utility). Otherwise, you do not want to print any data.



Testing condition codes – an example.

Consider IEBGENER to be successful if it concludes with a condition code of 0 or 4; that is, a condition code less than 8.

You would code the JCL as shown on the right. It specifies that STEP2 is only to be executed if STEP1 terminates successfully. That is, STEP2 is executed if STEP1 produces a condition code less than 8.



```
//COPYPRT   JOB
//STEP1     EXEC   PGM=IEBGENER
             .
             .
             .
//STEP2     EXEC   PGM=IEBPTPCH,
//                 COND=(8,LE,STEP1)
```

Continued...

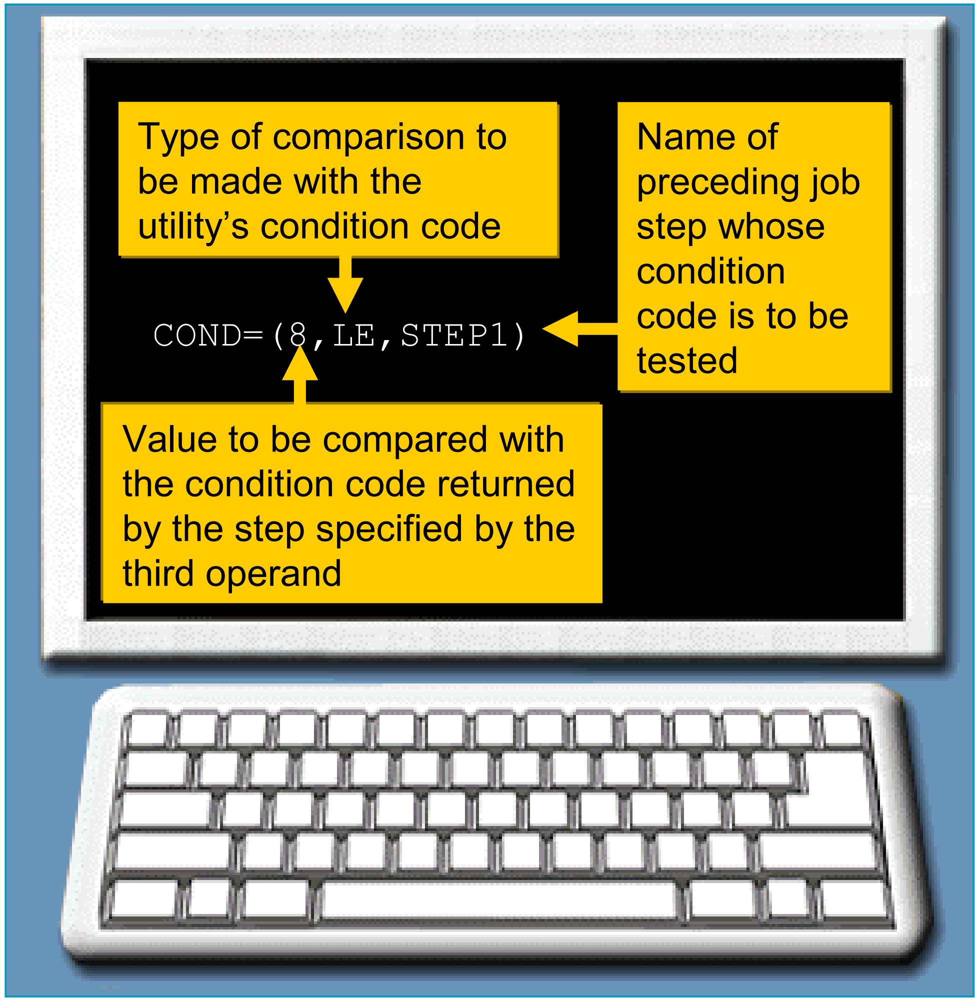
Interpreting utility communications.

Testing condition codes – an example.

The system interprets the COND parameter as follows:

If 8 is less than or equal to (LE) the condition code returned by STEP1, do not execute this step (containing the COND parameter).

Thus, STEP2 will execute only if STEP1 concludes with a condition code less than 8 (0 or 4).


Type of comparison to be made with the utility's condition code
Name of preceding job step whose condition code is to be tested
Value to be compared with the condition code returned by the step specified by the third operand

```
COND=(8,LE,STEP1)
```

Interpreting utility communications.

Are we on track?

Complete the COND parameter in the EXEC statement below for the following situation:

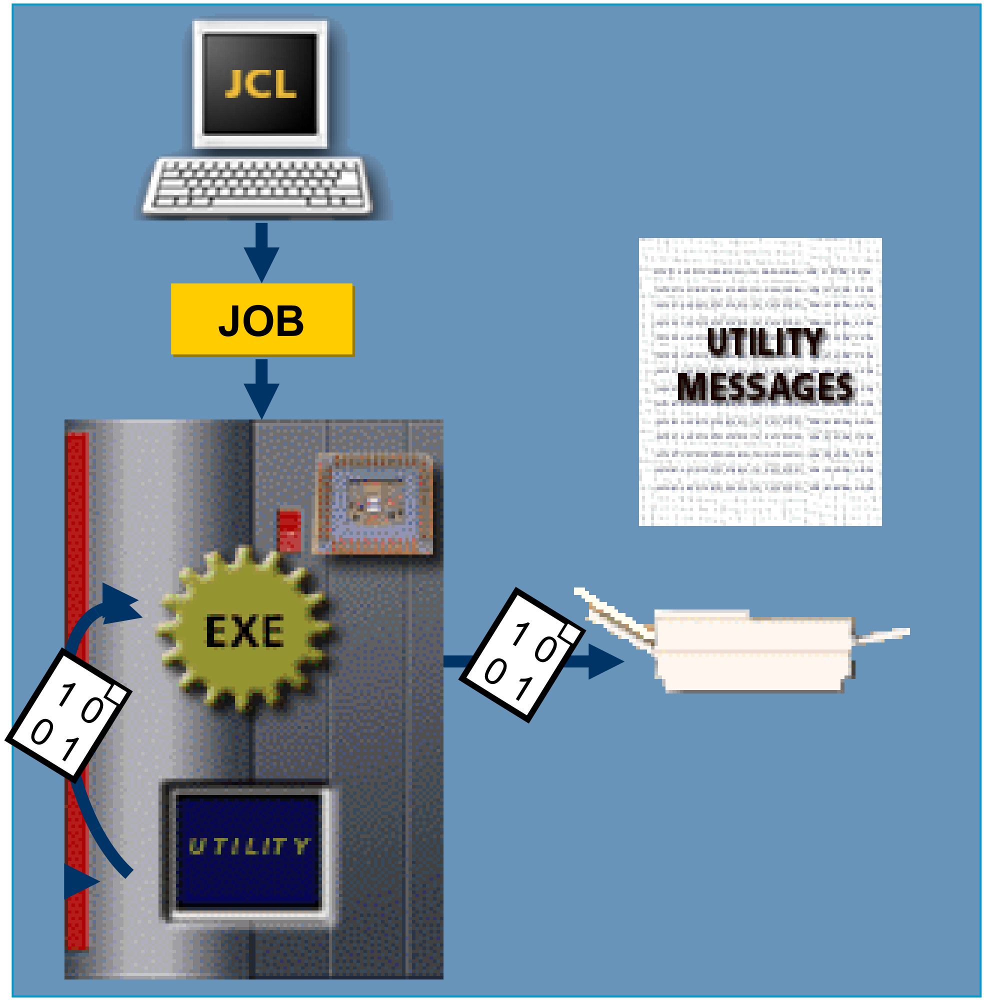
In STEP1 of your job, you want to print the directory of a PDS using IEHLIST. If the printing is successful, you will then add a new member to the directory using IEBUPDTE. (Assume the printing is successful if the system returns a code of 0.)

```
//STEP2        EXEC   PGM=IEBUPDTE,COND=____________
```

Interpreting utility communications.

Utility messages.

Each utility also creates utility messages. The messages are printed in the SYSPRINT data set. SYSPRINT output also includes the submitted control statements.



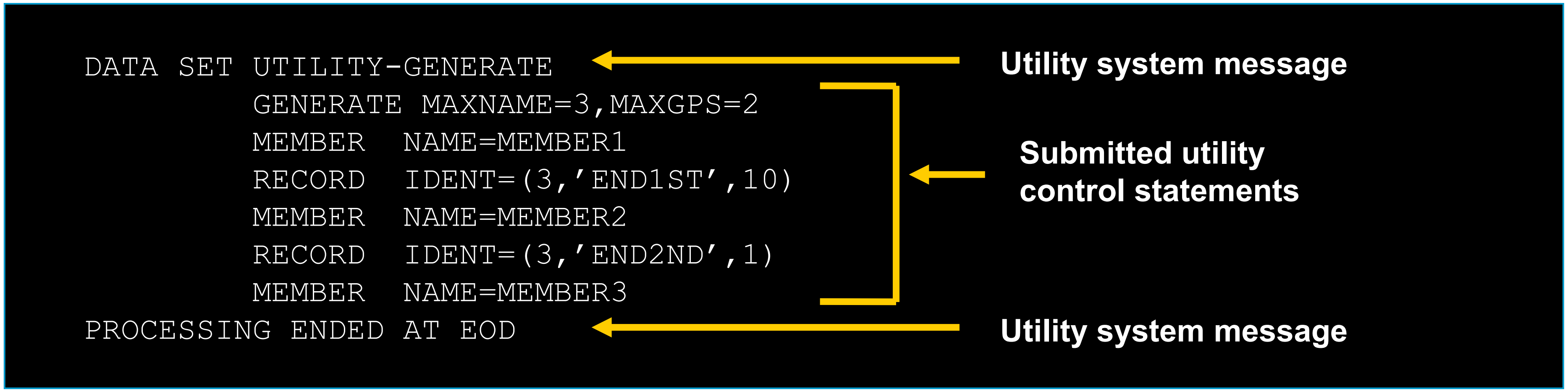
Some of the utility messages are informational and fairly self-explanatory. These utility messages usually do not have a message number associated with them. If the utility produces only informational types of messages, it continues its processing.

Informational utility messages can identify:

- Assumptions made by the utility.
- Default values taken by the utility.

Interpreting utility communications.

Utility messages – an example.



```
DATA SET UTILITY-GENERATE
         GENERATE MAXNAME=3,MAXGPS=2
         MEMBER   NAME=MEMBER1
         RECORD   IDENT=(3,'END1ST',10)
         MEMBER   NAME=MEMBER2
         RECORD   IDENT=(3,'END2ND',1)
         MEMBER   NAME=MEMBER3
PROCESSING ENDED AT EOD
```

Here is an example of SYSPRINT output after the IEBGENER utility successfully completed the task. The output indicates the following:

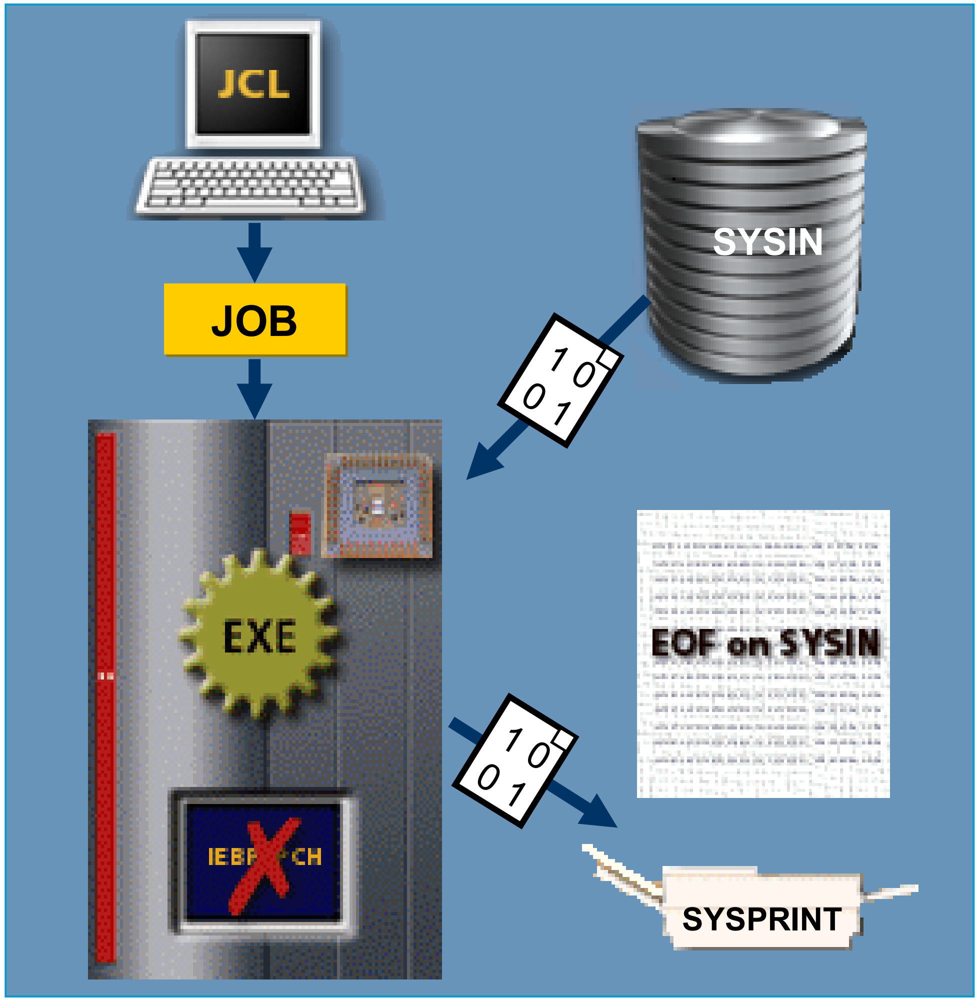
- Utility GENERATE (PGM=IEBGENER) was executed.
- The utility terminated normally. “PROCESSING ENDED AT EOD” (end-of-data) indicates the utility terminated after encountering end-of-file (EOF) on the input data set defined by the //SYSUT1 DD statement. The main indication that the processing completed normally is that there are no error messages printed.

Utility error messages.

Utility error messages can also be included in SYSPRINT output, which indicate that the utility encountered problems. The job may terminate, depending on the severity of the error.

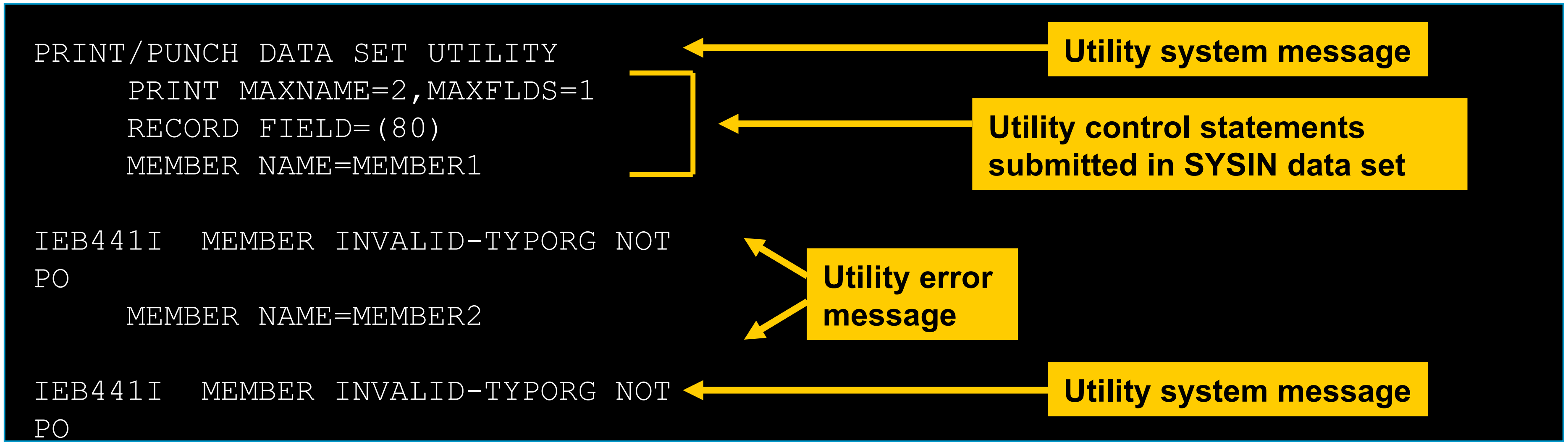
Error and warning messages display a message number, which enables you to look up the numbered message in the Utility Message Manual to find more information about the condition detected.

This will help in determining the source of the error and the correction required to fix it.



Interpreting utility communications.

Utility error messages – an example.



```
PRINT/PUNCH DATA SET UTILITY
     PRINT MAXNAME=2,MAXFLDS=1
     RECORD FIELD=(80)
     MEMBER NAME=MEMBER1

IEB441I  MEMBER INVALID-TYPORG NOT
PO
     MEMBER NAME=MEMBER2

IEB441I  MEMBER INVALID-TYPORG NOT
PO
```

Here is an example of SYSPRINT output when the IEBPTPCH utility encountered an error in the JCL and utility control statements. The output indicates the following:

- The utility PRINT/PUNCH (PGM=IEBPTCH) was executed.
- The utility did not perform the required task as indicated by the error messages (IEB441I). In addition, the utility indicates that the EOF was reached on the control data set (SYSIN) while the utility was searching for additional utility control statements.

Are we on track?

Which of the following JCL DD statements would produce utility messages in the output?

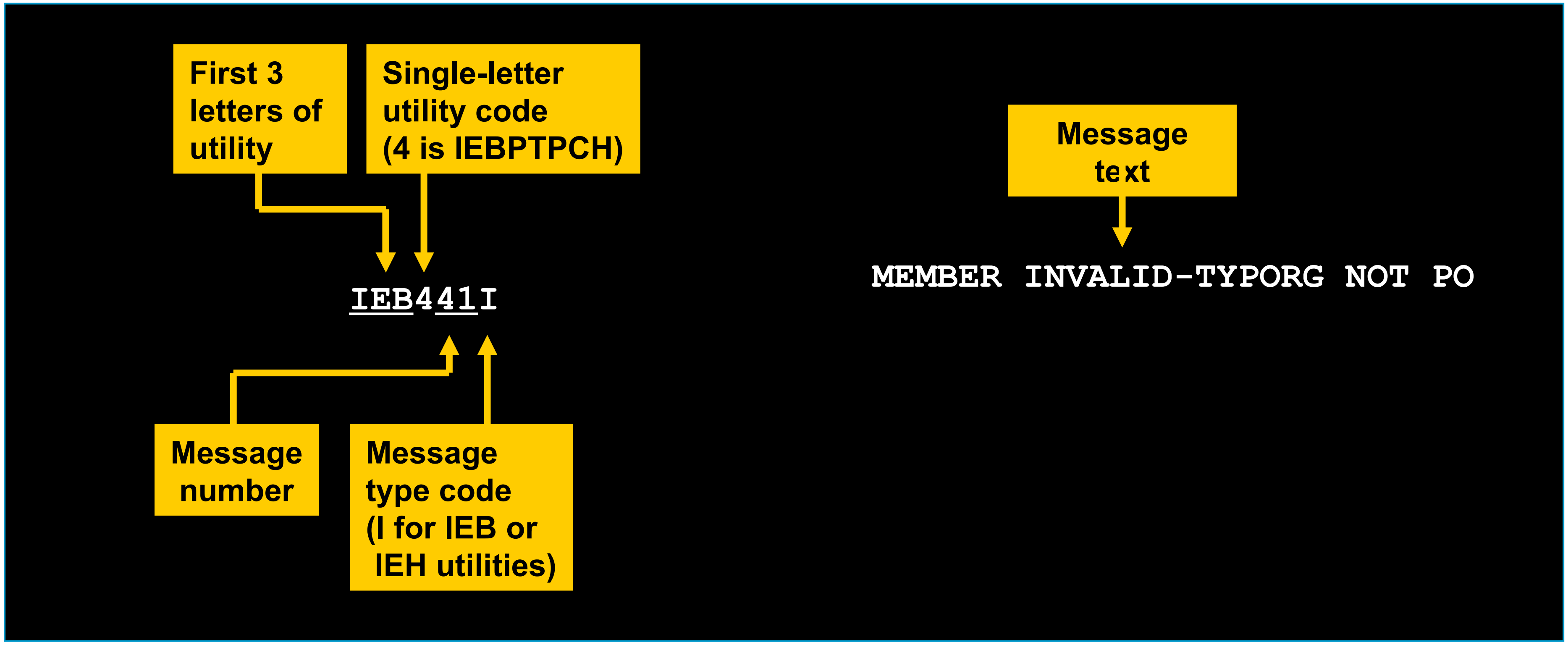
A. //SYSPRINT DD SYSOUT=C

B. //JOBNAME JOB MSGLEVEL=(1,0)

C. //SYSPRINT DD DUMMY

Interpreting utility communications.

Utility error message – general format.


First 3 letters of utility
Single-letter utility code (4 is IEBPTPCH)
Message text
IEB441I
MEMBER INVALID-TYPORG NOT PO
Message number
Message type code (I for IEB or IEH utilities)

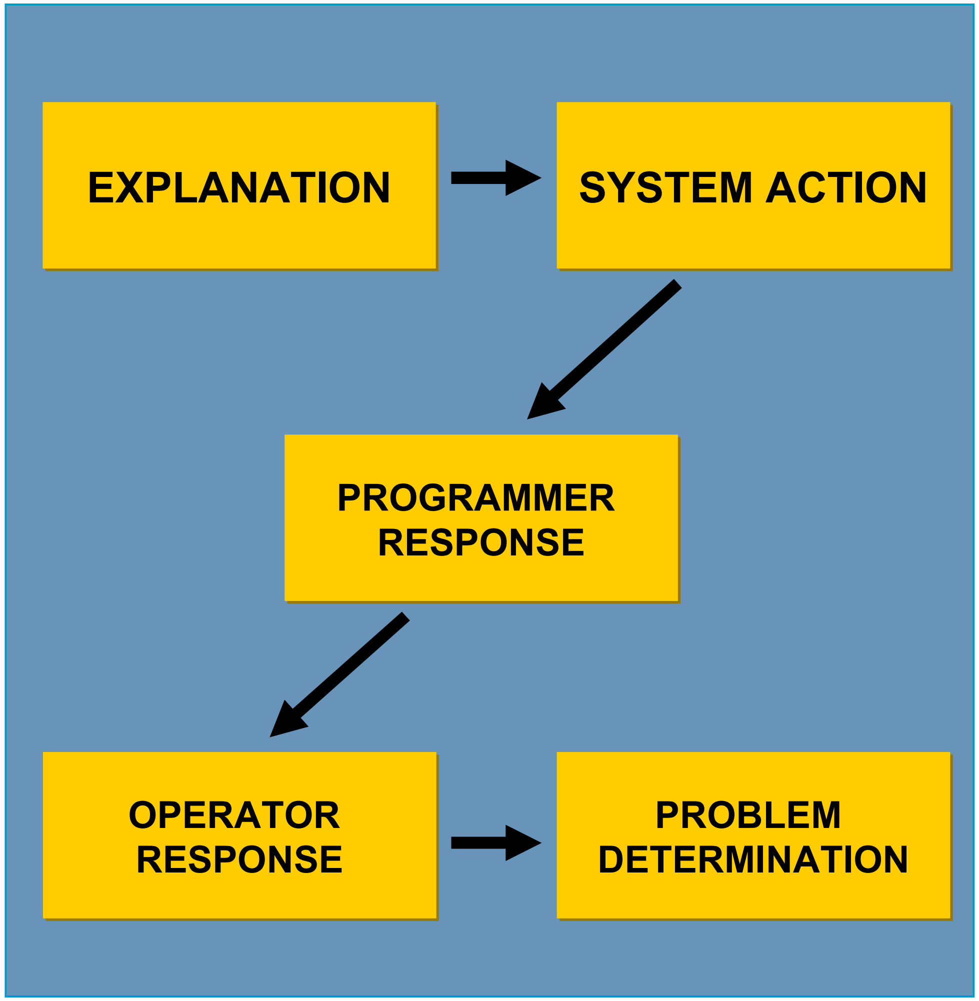
The general format for utility error messages is shown above.

Interpreting utility communications.

Utility messages manual.

Refer to your Utility Messages Manual for a detailed explanation of the warning and error messages that can be issued for each utility:

- Explanation: The cause of the problem.
- System Action: What the utility will do next.
- Programmer Response: Possible corrective measures to be taken by the programmer.
- Operator Response:Possible corrective measures to be taken by the computer operator.
- Problem Determination: Actions to be taken to solve the problem.

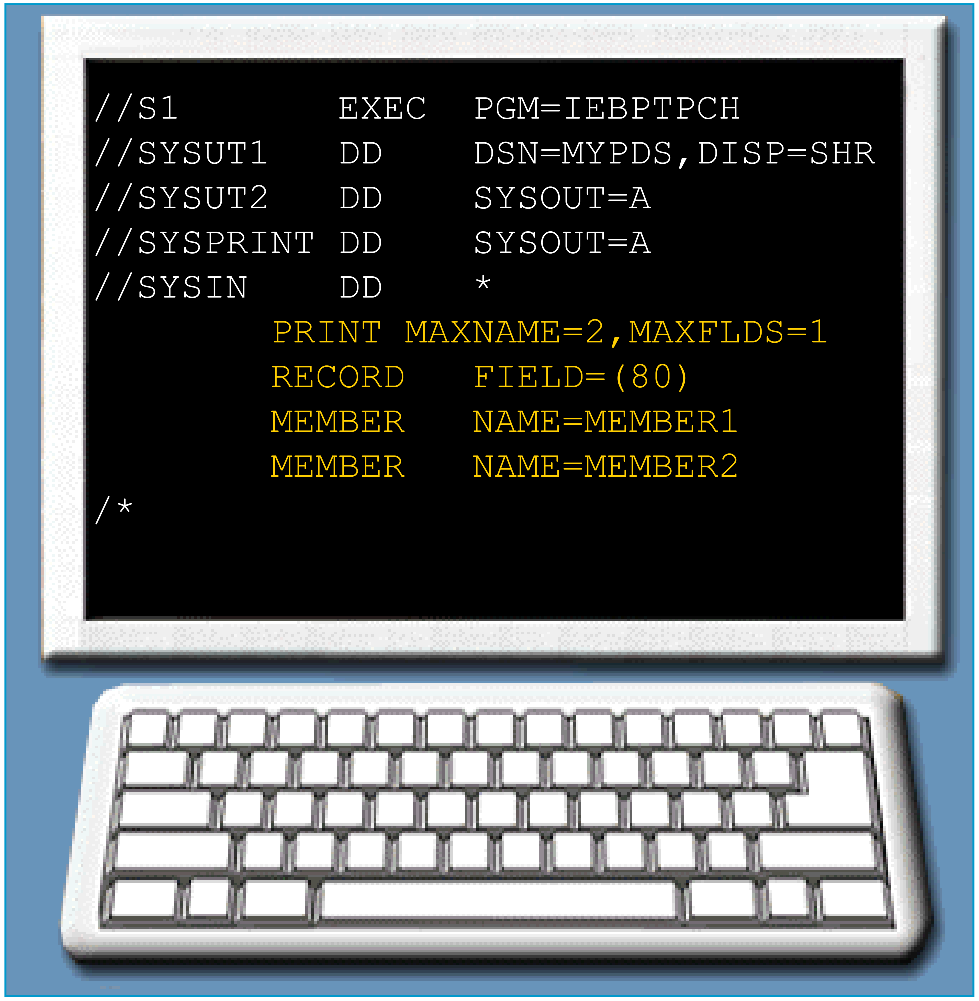


Summary example.

Interpreting utility messages.

The image shows a sample job stream, illustrating the process of interpreting a utility error message and correcting the problem.

The job stream will print MEMBER1 and MEMBER2 of a cataloged partitioned data set named MYPDS, using the utility IEBPTPCH.



```
//S1       EXEC   PGM=IEBPTPCH
//SYSUT1   DD     DSN=MYPDS,DISP=SHR
//SYSUT2   DD     SYSOUT=A
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
        PRINT MAXNAME=2,MAXFLDS=1
        RECORD    FIELD=(80)
        MEMBER    NAME=MEMBER1
        MEMBER    NAME=MEMBER2
/*
```

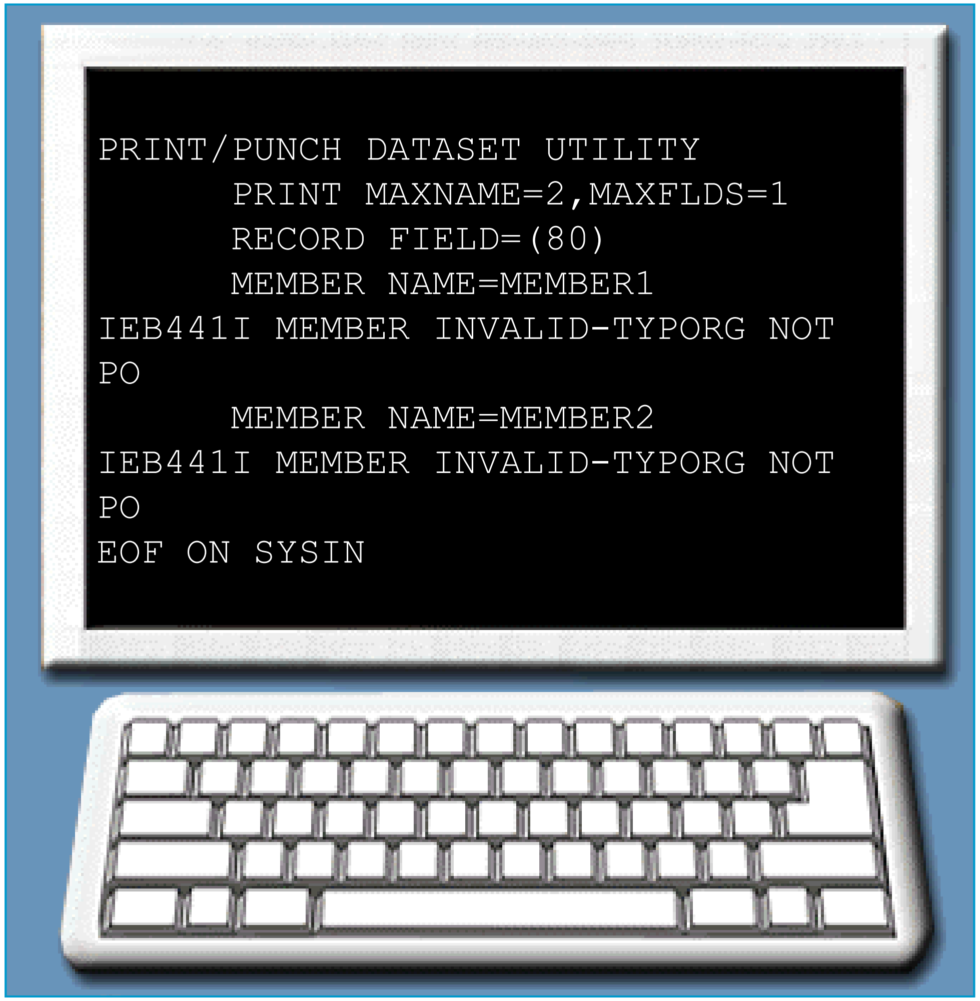
Continued...

Summary example.

Interpreting utility messages.

The job does not complete successfully. The contents of the SYSPRINT output are shown on the right.

The error message is IEB441I MEMBER INVALID.



```
PRINT/PUNCH DATASET UTILITY
      PRINT MAXNAME=2,MAXFLDS=1
      RECORD FIELD=(80)
      MEMBER NAME=MEMBER1
IEB441I MEMBER INVALID-TYPORG NOT
PO
      MEMBER NAME=MEMBER2
IEB441I MEMBER INVALID-TYPORG NOT
PO
EOF ON SYSIN
```

Continued...

Summary example.

Interpreting utility messages.

```
IEB441I MEMBER INVALID: TYPORG NOT PO
```

Explanation: The MEMBER statement preceding this message is incorrect since physical sequential (PS) organization was specified.
That is, TYPORG=PO must be specified on the PRINT or PUNCH utility control statement.

Source: DFSMSdfp

System Action: The program is ended at the end of the control statement scan. The return code is 12.

Application Programmer Response: Probable user error. If SYSUT1 specifies a physical sequential data set, remove the MEMBER statement. If SYSUT1 specifies a partitioned data set, specify TYPORG=PO on the PRINT or PUNCH statement.

The above shows the explanation given by the Utility System Messages Manual for message IEB441I.

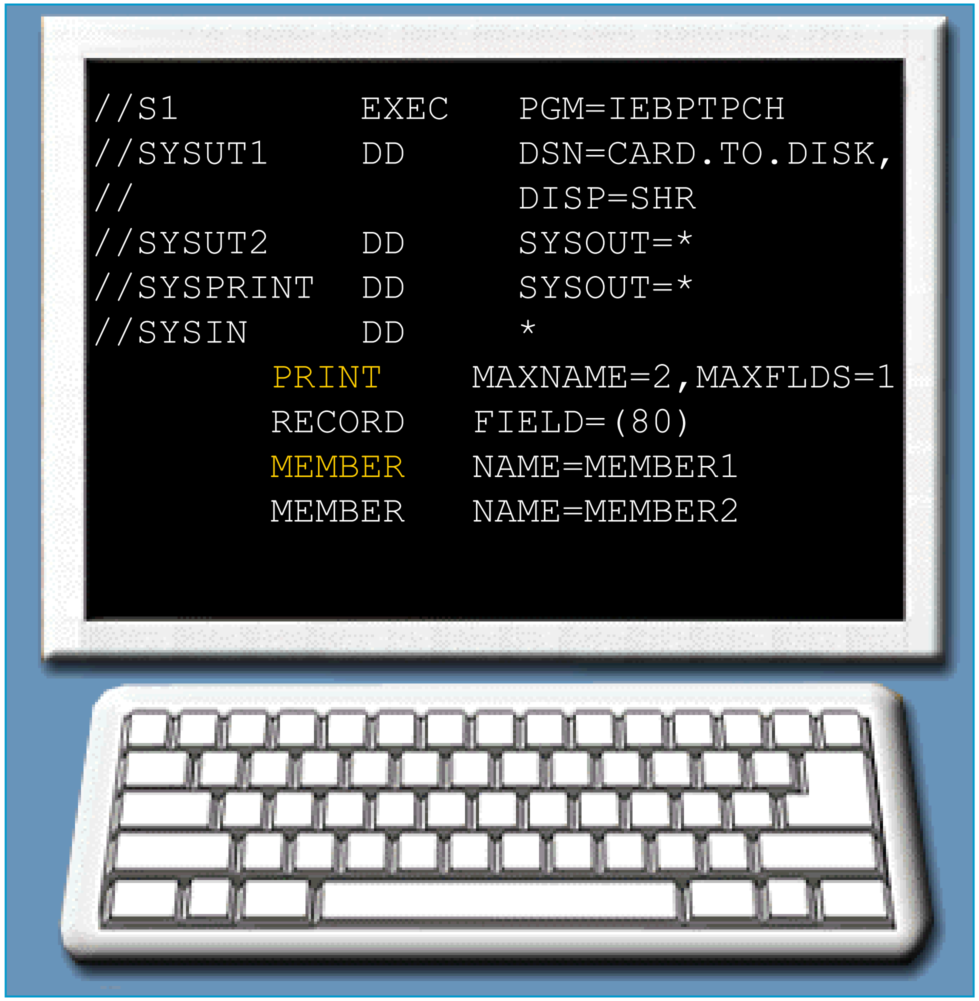
Summary example.

Interpreting utility messages – error condition.

The error condition is not caused by any single statement, but rather by two control statements with conflicting information:

1. The MEMBER utility control statement in the SYSIN data set indicates you are processing members of a partitioned data set.

2. For a partitioned data set, you must specify TYPORG=PO on the PRINT control statement.



```
//S1          EXEC    PGM=IEBPTPCH
//SYSUT1      DD      DSN=CARD.TO.DISK,
//                    DISP=SHR
//SYSUT2      DD      SYSOUT=*
//SYSPRINT    DD      SYSOUT=*
//SYSIN       DD      *
        PRINT       MAXNAME=2,MAXFLDS=1
        RECORD      FIELD=(80)
        MEMBER      NAME=MEMBER1
        MEMBER      NAME=MEMBER2
```

Summary example.

Are we on track?

Review the PRINT control statement that was submitted to the utility:

```
PRINT MAXNAME=2,MAXFLDS=1
```

Complete the PRINT utility control statement to correct the error.

PRINT MAXNAME=2,MAXFLDS=1,__________

Glossary.

IEBGENER Utility
A data set utility program that is designed to copy records from a sequential data set.

IEBPTPCH Utility
A standard IBM utility program that is designed to print or punch data sets.

TYPRUN=SCAN
A JOB statement parameter that suppresses execution of the job. It is often used for checking JCL syntax errors.

Operands
Keyword or positional statements in the operand field of a JCL statement.

Data Control Block
A parameter on a DD statement that describes the attributes of a data set, such as block size and record format.

Summary example.

Unit summary.

Now that you have completed this unit, you should be able to:

- Use your Utilities Manual to identify utility programs available to accomplish a task.
- Identify the JCL statements needed to communicate with selected utilities.
- Specify the purpose of utility control statements.
- Identify utility control statements that have been coded correctly according to the syntax rules.
- Interpret informational and error messages produced by utilities.
- Correct control statements that were coded incorrectly.

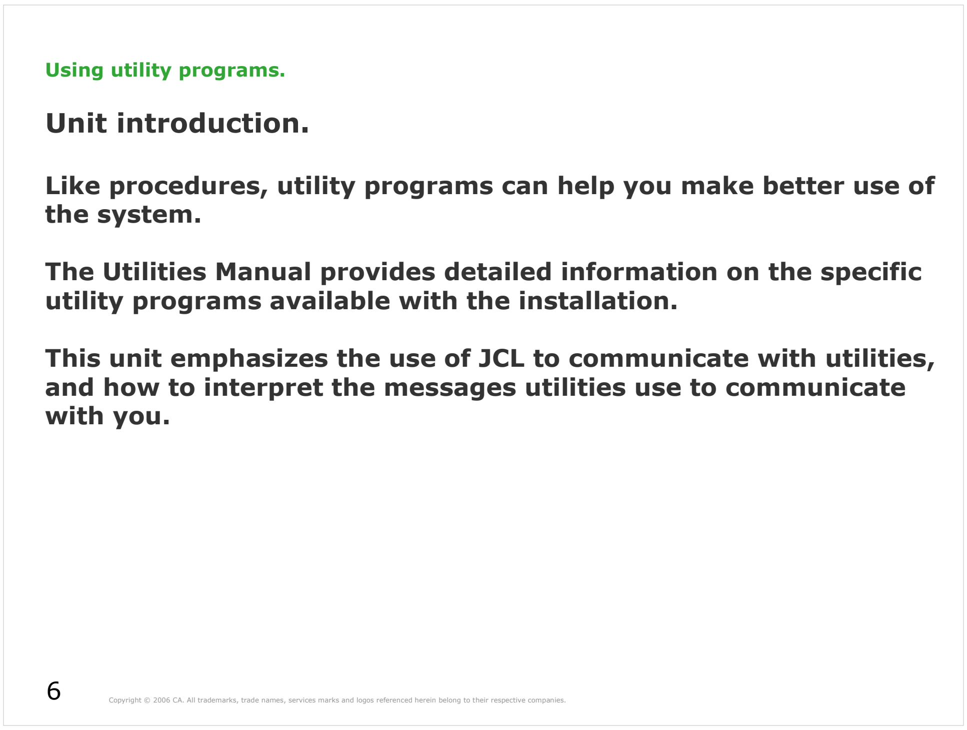
JCL

Chapter c3
Using utility programs

Job Control Language

Chapter c3

Using utility programs



Using utility programs.

Unit introduction.

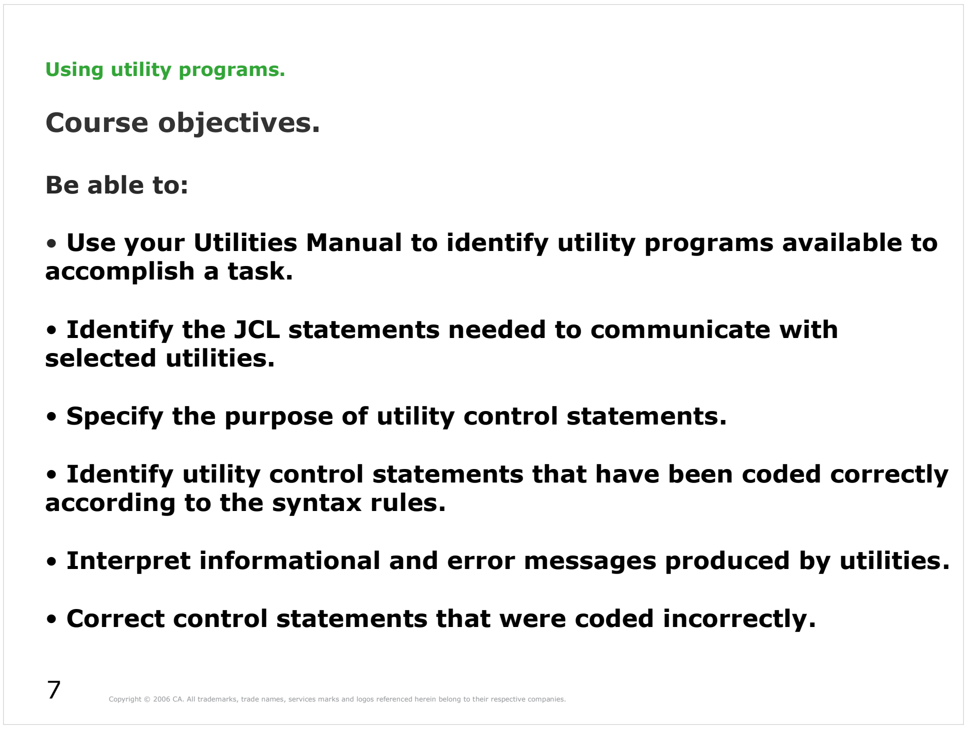
Like procedures, utility programs can help you make better use of the system.

The Utilities Manual provides detailed information on the specific utility programs available with the installation.

This unit emphasizes the use of JCL to communicate with utilities, and how to interpret the messages utilities use to communicate with you.

6

See the „z/OS DFSMSdfp Utilities“ book.



Course objectives.

Be able to:

- **Use your Utilities Manual to identify utility programs available to accomplish a task.**

- **Identify the JCL statements needed to communicate with selected utilities.**

- **Specify the purpose of utility control statements.**

- **Identify utility control statements that have been coded correctly according to the syntax rules.**

- **Interpret informational and error messages produced by utilities.**

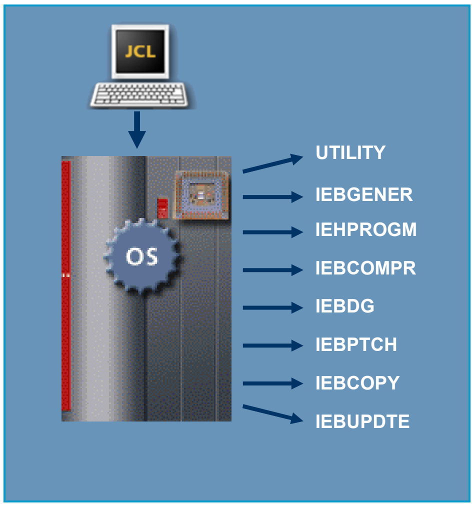
- **Correct control statements that were coded incorrectly.**

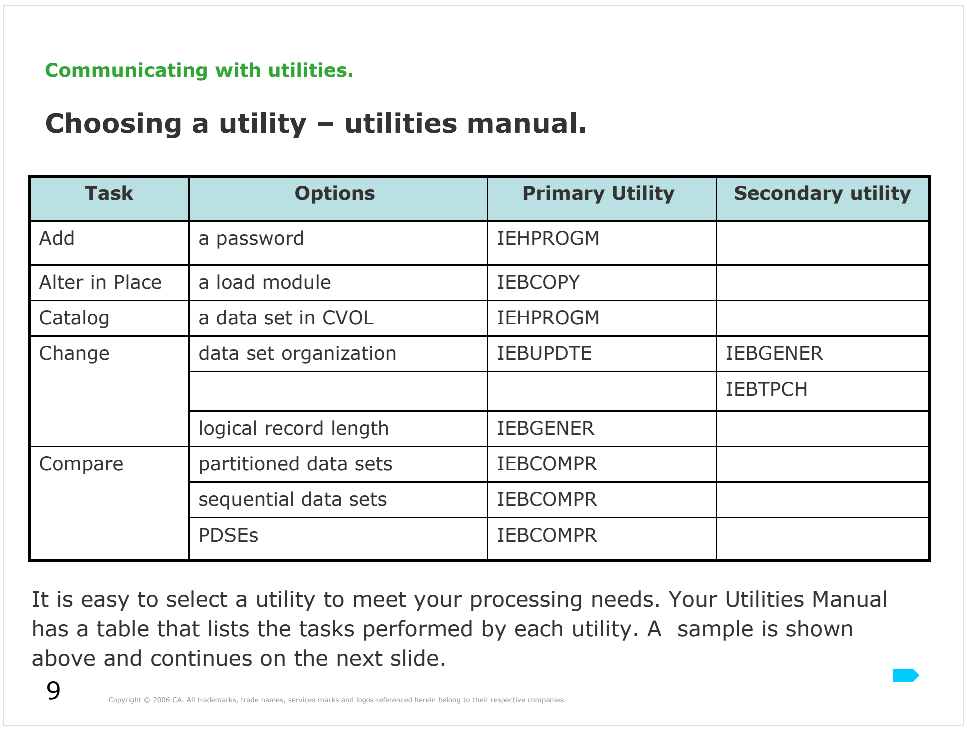
Choosing a utility.

What are utility programs?

Utility programs are general purpose programs that are a part of your OS. They are designed to help you reorganize, compare, or change data at the data set or record level.

Utilities have been in use for many years. Today, some of the functions that utilities have provided may be better performed with applications such as ISPF/PDF. However, utilities are still useful to perform functions in a way that will work in all MVS installations.




Communicating with utilities.

Choosing a utility – utilities manual.

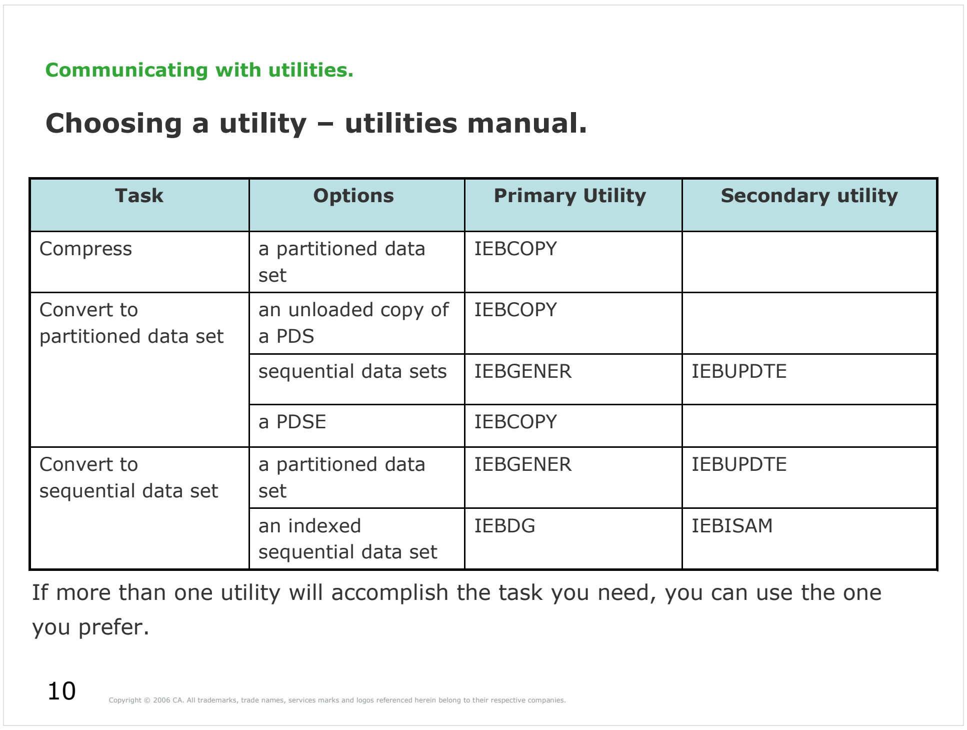
Task	Options	Primary Utility	Secondary utility
Add	a password	IEHPROGM	
Alter in Place	a load module	IEBCOPY	
Catalog	a data set in CVOL	IEHPROGM	
Change	data set organization	IEBUPDTE	IEBGENER
			IEBTPCH
	logical record length	IEBGENER	
Compare	partitioned data sets	IEBCOMPR	
	sequential data sets	IEBCOMPR	
	PDSEs	IEBCOMPR	

It is easy to select a utility to meet your processing needs. Your Utilities Manual has a table that lists the tasks performed by each utility. A sample is shown above and continues on the next slide.

9

Detailed information on how to use each utility is found in individual chapters, which are sequenced alphabetically by utility name.

See Chapter 1, „Guide to Utility program functions“.



Communicating with utilities.

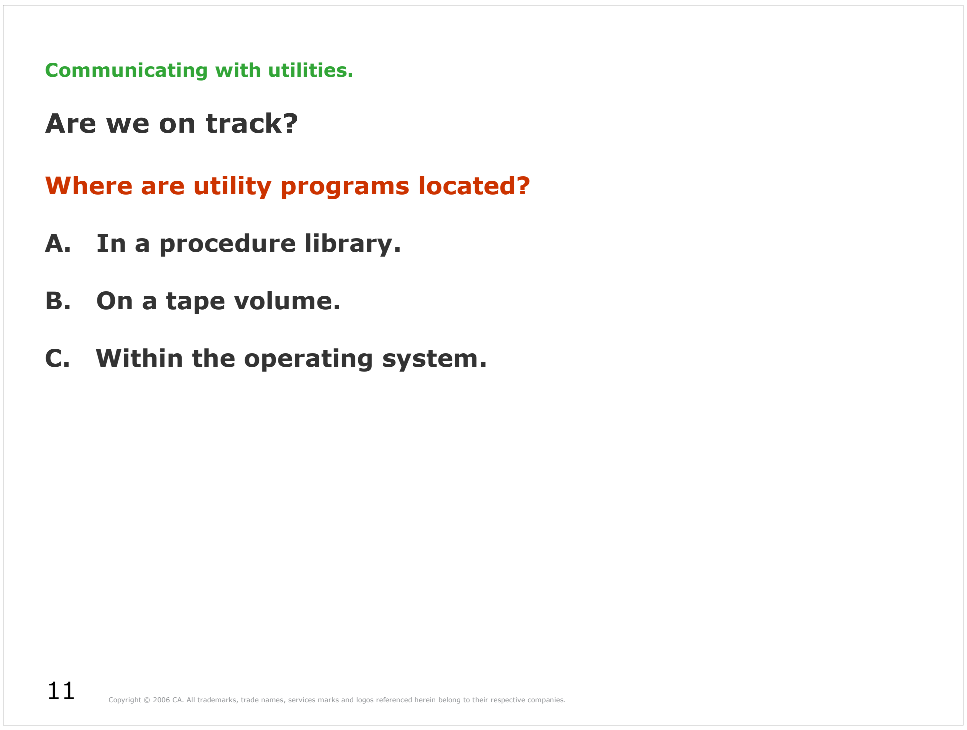
Choosing a utility – utilities manual.

Task	Options	Primary Utility	Secondary utility
Compress	a partitioned data set	IEBCOPY	
Convert to partitioned data set	an unloaded copy of a PDS	IEBCOPY	
	sequential data sets	IEBGENER	IEBUPDTE
	a PDSE	IEBCOPY	
Convert to sequential data set	a partitioned data set	IEBGENER	IEBUPDTE
	an indexed sequential data set	IEBDG	IEBISAM

If more than one utility will accomplish the task you need, you can use the one you prefer.

10

In general, choose the utility that you are most familiar with (and have used before), or requires the least amount of coding (JCL and control statements).



Communicating with utilities.

Are we on track?

Where are utility programs located?

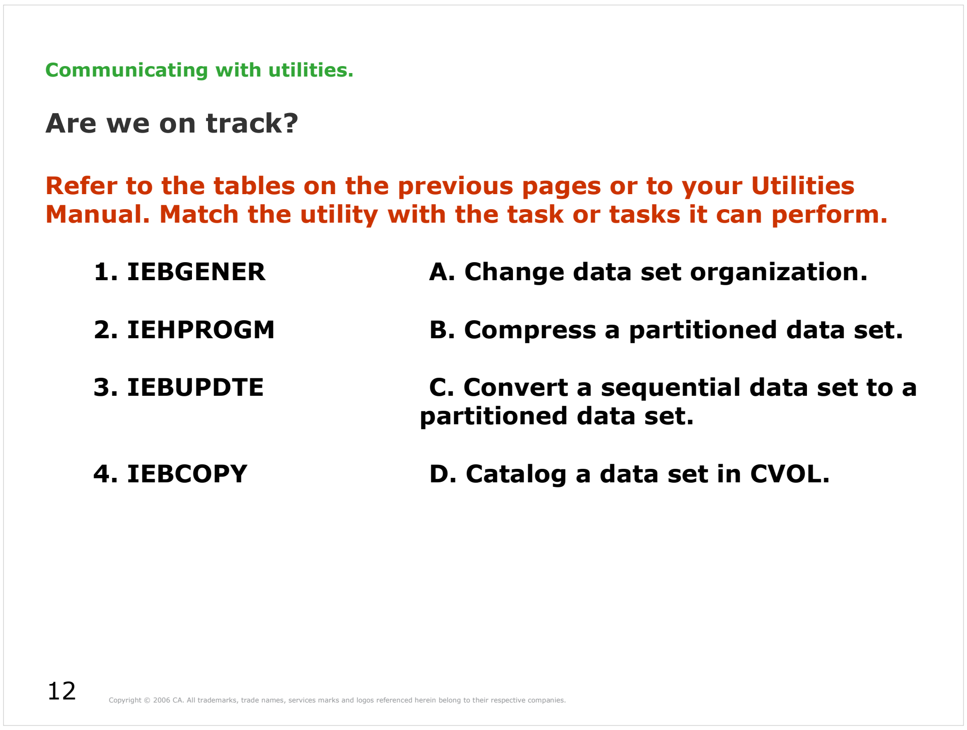
A. In a procedure library.

B. On a tape volume.

C. Within the operating system.

11

The correct answer is C.



Communicating with utilities.

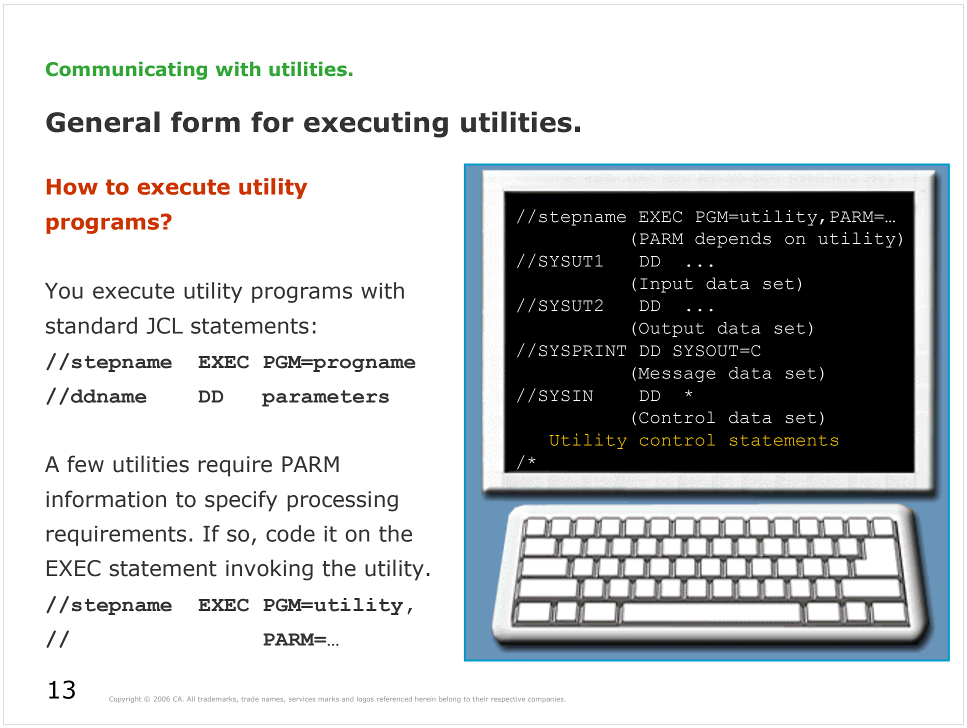
Are we on track?

Refer to the tables on the previous pages or to your Utilities Manual. Match the utility with the task or tasks it can perform.

1. IEBGENER	**A. Change data set organization.**
2. IEHPROGM	**B. Compress a partitioned data set.**
3. IEBUPDTE	**C. Convert a sequential data set to a partitioned data set.**
4. IEBCOPY	**D. Catalog a data set in CVOL.**

12

The correct answer is 1-C, 2-D, 3-A, 4-B.



Communicating with utilities.

General form for executing utilities.

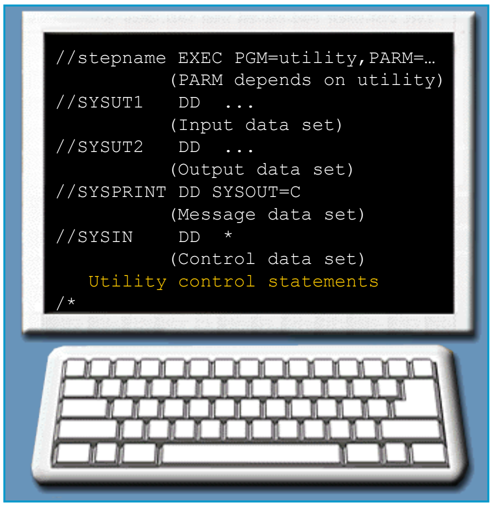
How to execute utility programs?

You execute utility programs with standard JCL statements:

```
//stepname   EXEC  PGM=progname
//ddname     DD    parameters
```

A few utilities require PARM information to specify processing requirements. If so, code it on the EXEC statement invoking the utility.

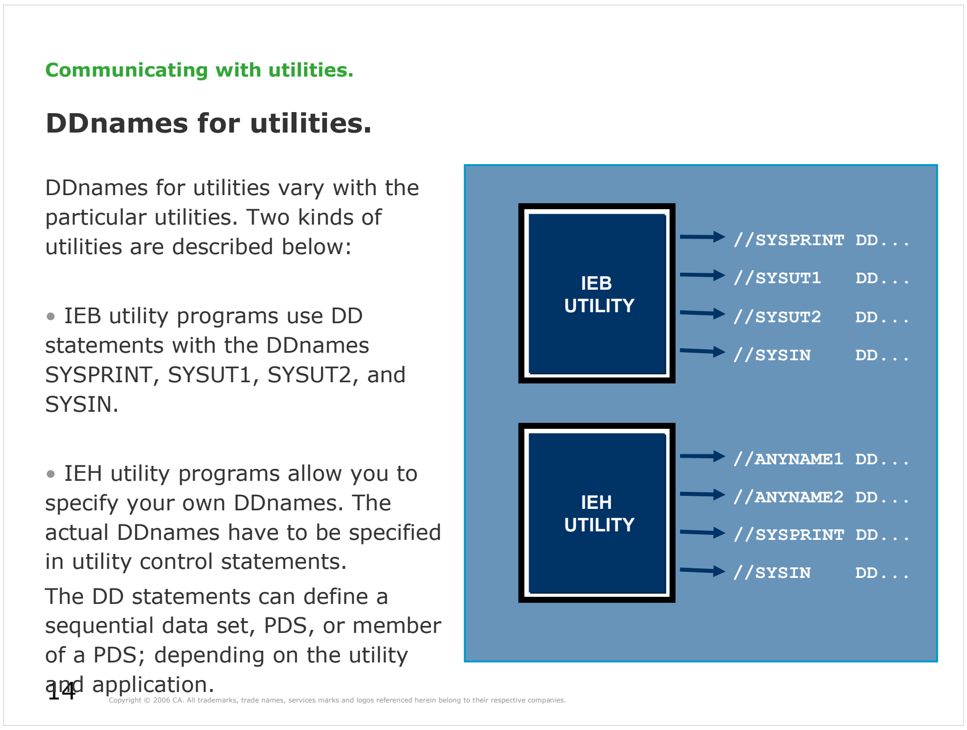
```
//stepname   EXEC  PGM=utility,
//                 PARM=…
```



```
//stepname EXEC PGM=utility,PARM=…
          (PARM depends on utility)
//SYSUT1   DD  ...
          (Input data set)
//SYSUT2   DD  ...
          (Output data set)
//SYSPRINT DD SYSOUT=C
          (Message data set)
//SYSIN    DD  *
          (Control data set)
   Utility control statements
/*
```

13

The job stream needed for an IEB utility is shown on the right.



Communicating with utilities.

DDnames for utilities.

DDnames for utilities vary with the particular utilities. Two kinds of utilities are described below:

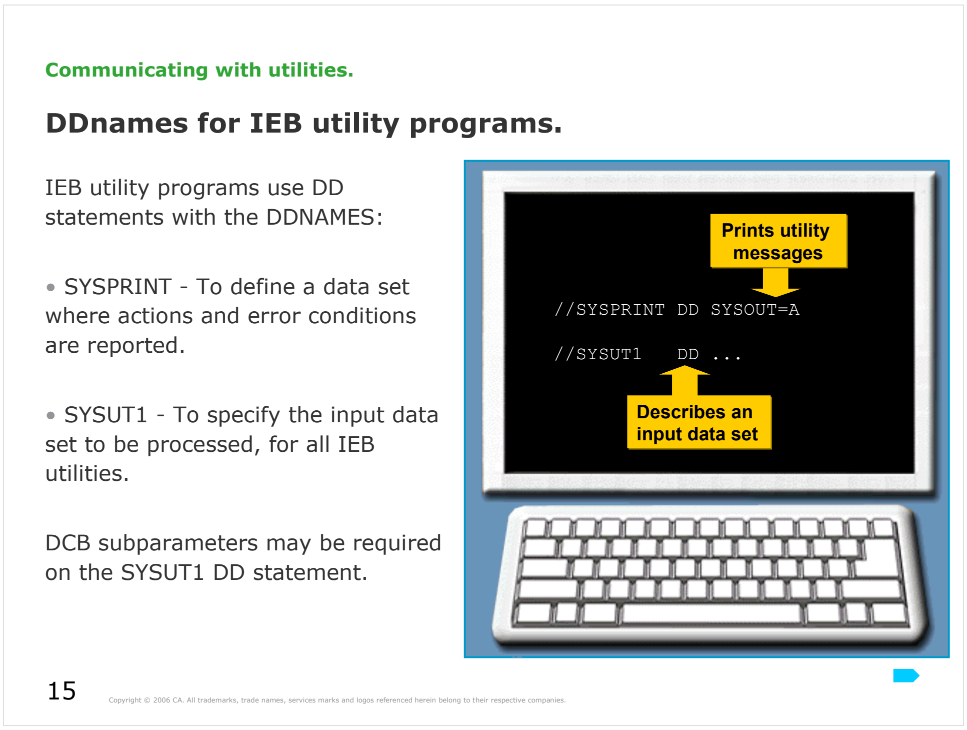
- IEB utility programs use DD statements with the DDnames SYSPRINT, SYSUT1, SYSUT2, and SYSIN.

- IEH utility programs allow you to specify your own DDnames. The actual DDnames have to be specified in utility control statements.

The DD statements can define a sequential data set, PDS, or member of a PDS; depending on the utility and application.

```
//SYSPRINT DD...
//SYSUT1   DD...
//SYSUT2   DD...
//SYSIN    DD...

//ANYNAME1 DD...
//ANYNAME2 DD...
//SYSPRINT DD...
//SYSIN    DD...
```

The Utilities Manual indicates the DDnames as “anyname1” and “anyname2”. You must substitute valid and unique DDnames for “anyname1” and “anyname2”.



Communicating with utilities.

DDnames for IEB utility programs.

IEB utility programs use DD statements with the DDNAMES:

- SYSPRINT - To define a data set where actions and error conditions are reported.
- SYSUT1 - To specify the input data set to be processed, for all IEB utilities.

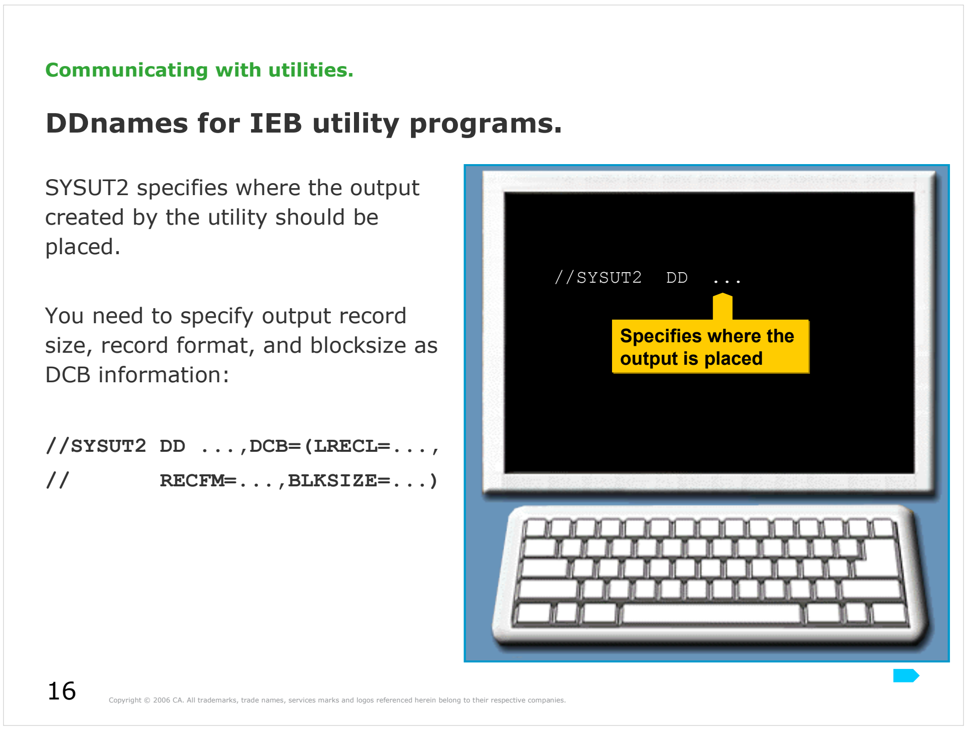
DCB subparameters may be required on the SYSUT1 DD statement.

Prints utility messages
Describes an input data set

```
//SYSPRINT DD SYSOUT=A

//SYSUT1   DD ...
```

15

It is strongly recommended that SYSOUT be specified in the SYSPRINT DD statement.
For IEH utilities you can specify any valid DDname.



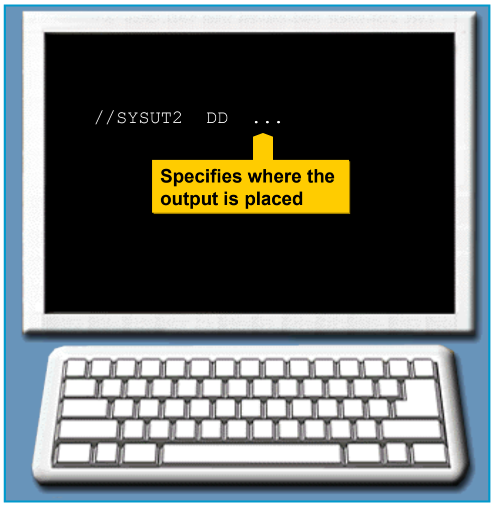
Communicating with utilities.

DDnames for IEB utility programs.

SYSUT2 specifies where the output created by the utility should be placed.

You need to specify output record size, record format, and blocksize as DCB information:

```
//SYSUT2 DD ...,DCB=(LRECL=...,
//         RECFM=...,BLKSIZE=...)
```



```
//SYSUT2  DD  ...
```

16

If this information is the same as the input (SYSUT1) data set's information, you do not need to code it on the SYSUT2 DD statement. The utility uses the same information that it finds for the input data set.

For IEH utilities you can specify any valid DDname.

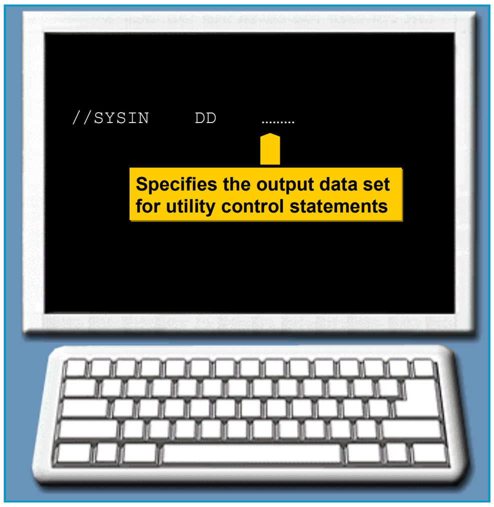
Communicating with utilities.

DDnames for IEB utility programs.

SYSIN defines the control data set, in which the utility control statements you code are placed. Usually, this data set is in the job stream.

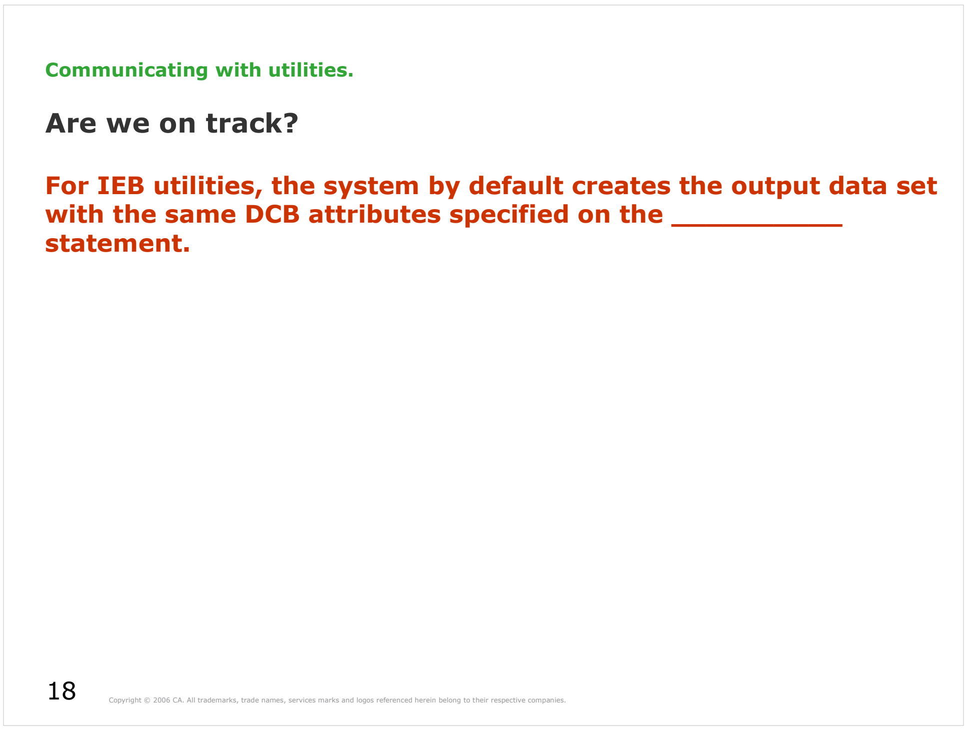
If you do not need any control statements, you should code:

```
//SYSIN     DD      DUMMY
```



```
//SYSIN     DD      .........
```

Specifies the output data set for utility control statements



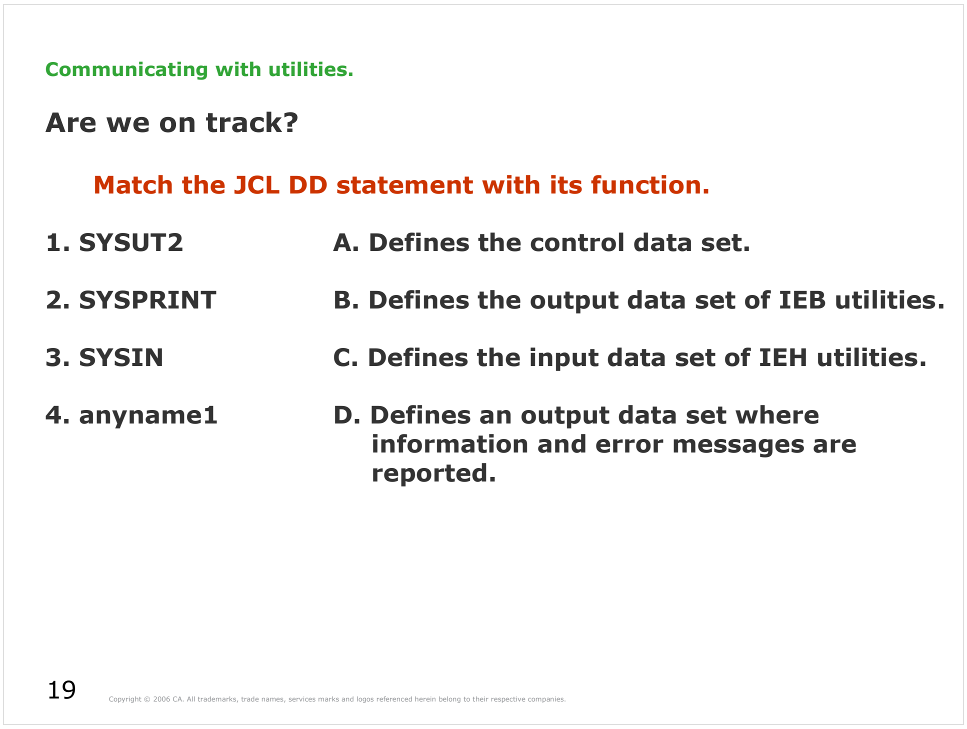
Communicating with utilities.

Are we on track?

For IEB utilities, the system by default creates the output data set with the same DCB attributes specified on the ___________ statement.

18

The correct answer is SYSUT1.



Communicating with utilities.

Are we on track?

Match the JCL DD statement with its function.

1. SYSUT2	**A. Defines the control data set.**
2. SYSPRINT	**B. Defines the output data set of IEB utilities.**
3. SYSIN	**C. Defines the input data set of IEH utilities.**
4. anyname1	**D. Defines an output data set where information and error messages are reported.**

19

The correct answer is 1-B, 2-D, 3-A, 4-C.

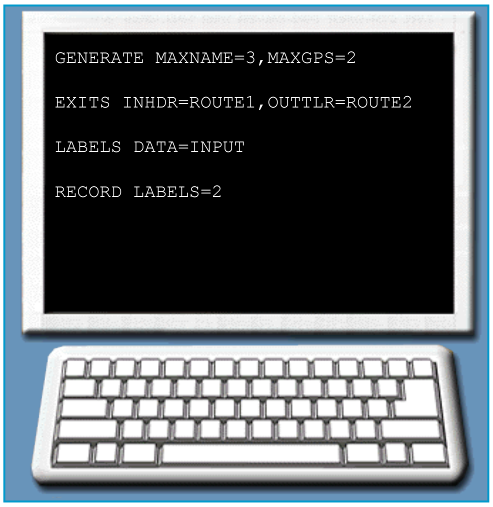
Communicating with utilities.

Utility control statements.

Why code a utility control statement?

Utility control statements are coded to specify to the utility the task you want to perform and, in some cases, the data set to be processed. Each utility has a list of available control statements.

Examples of control statements used with IEBGENER are shown on the right.

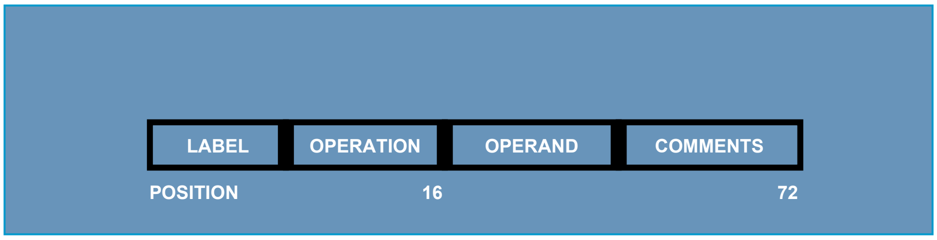


```
GENERATE MAXNAME=3,MAXGPS=2

EXITS INHDR=ROUTE1,OUTTLR=ROUTE2

LABELS DATA=INPUT

RECORD LABELS=2
```

Utility control statements are not JCL statements. The syntax of the control statements for each utility is described in detail in your Utilities Manual.

Communicating with utilities.

Utility control statements – general format.

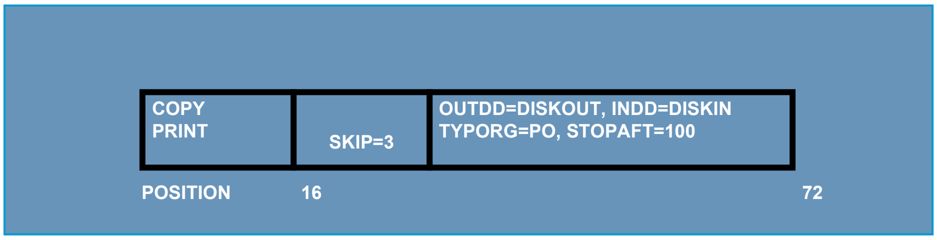


The control statements used by all of the utilities (with the exception of IEBUPDTE) have the general format shown above indicating the standard coding positions, where:

- LABEL symbolically identifies the control statement. LABEL is optional in most cases.
- OPERATION identifies the type of control statement.
- The OPERAND is made up of one or more keyword parameters, separated by commas.

Communicating with utilities.

Utility control statements – standard coding positions.



The general form for standard coding positions are:

- Control statements are coded as in-stream data in columns 2 through 71.
- To continue a control statement, code a nonblank character in column 72.
- Then following standard coding procedures, you continue the statement in column 16 of the following line.

Communicating with utilities.

Notational conventions to code a special DD statements.

In the Utilities Manual, certain symbols called notational conventions indicate whether control statement labels, operands, or sub-operands are necessary or optional.

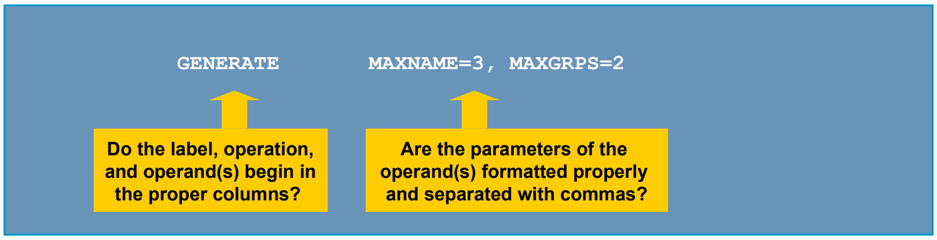
For example, brackets [ ] are sometimes used to indicate that entry is optional: [label]

The notational conventions to code a special DD statement are as follows:

[ ]	**Brackets enclose an optional entry.**
|	**An OR sign (a vertical bar) separates alternative entries.**
{ }	**Braces enclose alternative entries. You can only use one of the entries.**
"	**Quotation marks indicate that a space must be left before the next parameter.**

Communicating with utilities.

Utility control statements – syntax.

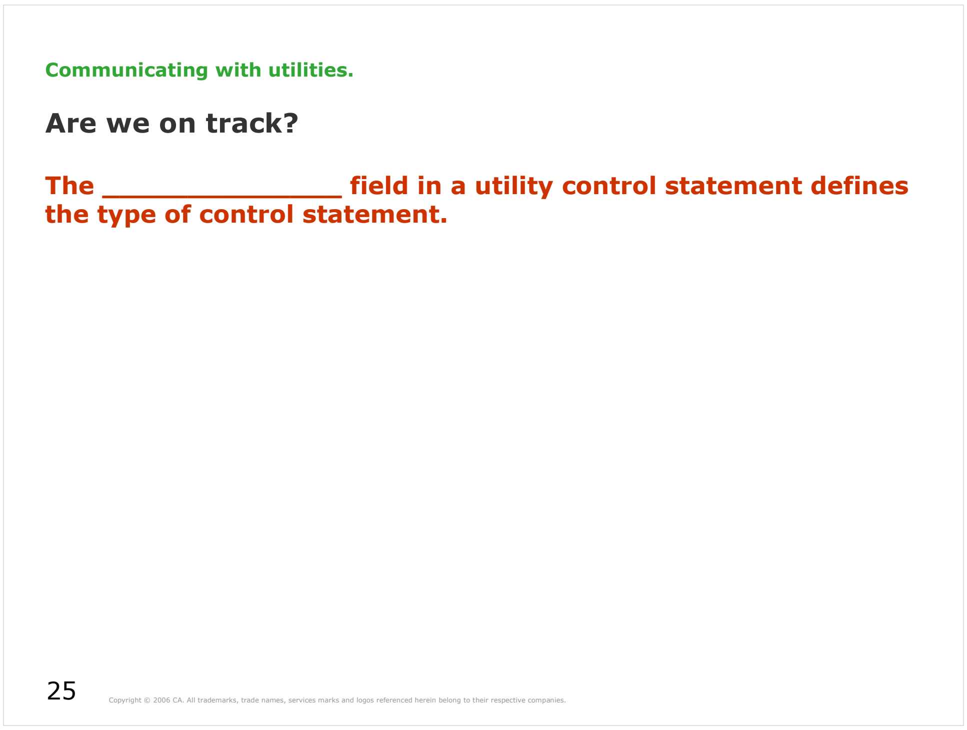


```
GENERATE     MAXNAME=3, MAXGRPS=2
```

Do the label, operation, and operand(s) begin in the proper columns?
Are the parameters of the operand(s) formatted properly and separated with commas?

At execution, all of the utilities verify that the control statements you supply have valid syntax and content. If there are syntax errors, you should consider the following:

- Do the label, operation, and operand(s) begin in the proper columns?
- Are the continuation statements coded in the proper format?
- Are the parameters of the operand(s) formatted properly and separated with commas?

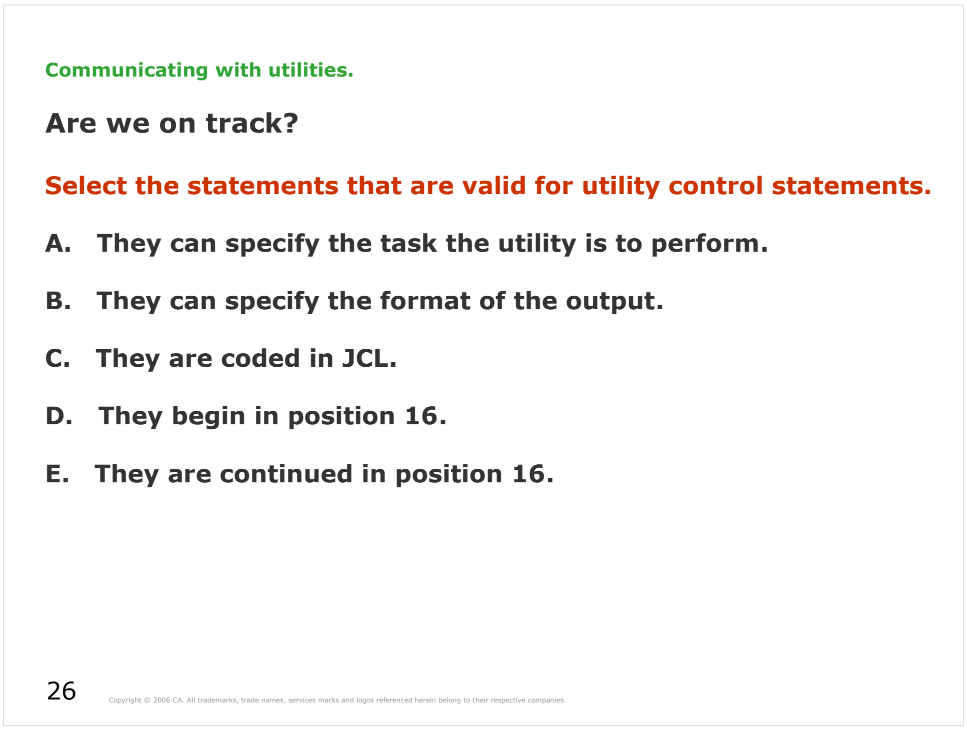

Communicating with utilities.

Are we on track?

The _________________ field in a utility control statement defines the type of control statement.

25

The correct answer is operation.



Communicating with utilities.

Are we on track?

Select the statements that are valid for utility control statements.

A. They can specify the task the utility is to perform.

B. They can specify the format of the output.

C. They are coded in JCL.

D. They begin in position 16.

E. They are continued in position 16.

26

The correct answer is A., B., and E.

Glossary.

PARM
A parameter on the EXEC statement that passes control information (such as DEBUG) to the job step.

IEB Utility Programs
System utility programs that are used to list or change information related to data sets & volumes.

IEH Utility Programs
Data set utility programs that are used to reorganize, change, or compare data at the data set or record level.

LABEL
A DD statement parameter that contains information on a non-temporary data set, like volume identification.

Interpreting utility communications.

Kinds of communications.

How do utility programs communicate?

Utility programs communicate with you through condition code settings and utility messages.

What do the messages indicate?

These messages indicate if the utility:

- Understood the request for processing.
- Completed the requested processing successfully.

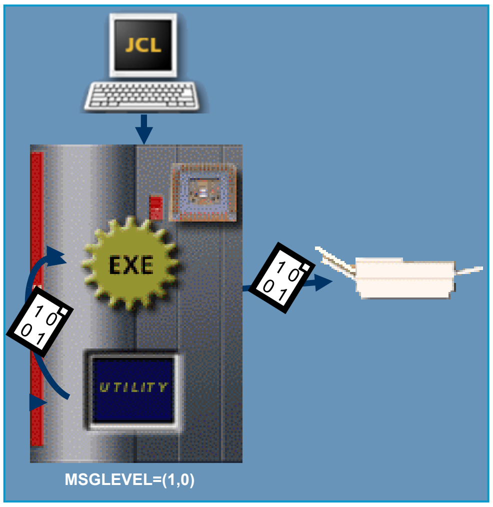
Interpreting utility communications.

Condition codes.

What are Condition codes?

Condition codes are produced by the utility as it concludes. They indicate whether the job was successfully completed.

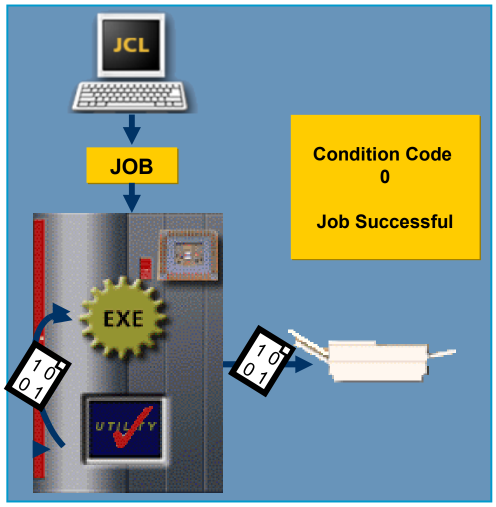
Condition codes are printed in the job log's allocation/termination listing. You print the job log by coding MSGLEVEL=(1,0) on the JOB statement.



Kinds of condition codes – zero condition code.

Zero Condition Code means that the utility detected no errors in the control statement information.

However, this does not necessarily mean that the utility did what you wanted it to do. (It may have assumed inappropriate default values for control statement parameters you did not specify.)



Interpreting utility communications.

Condition codes – sample.

Utility	0	4	8	12	16
IEBGENER	Successful completion.	Warning. Probable completion	Processing ended at user's request.	Unrecoverable error. Job step terminated.	Job step terminated.
IEBEDIT	Successful completion.	Error condition. Recovery may be possible.	Unrecoverable error.	Not used.	Not used.

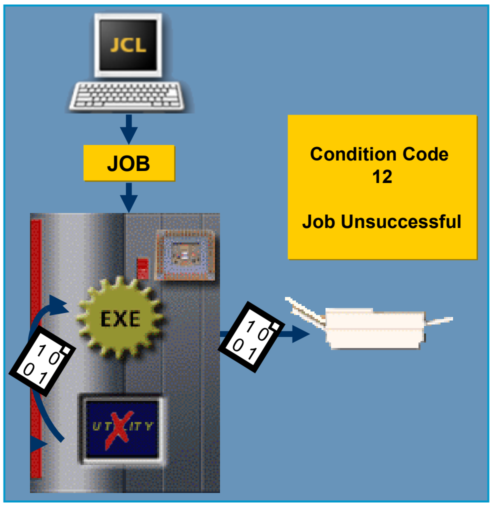
The table above shows sample condition codes created by the IEBGENER and IEBEDIT utilities.

Interpreting utility communications.

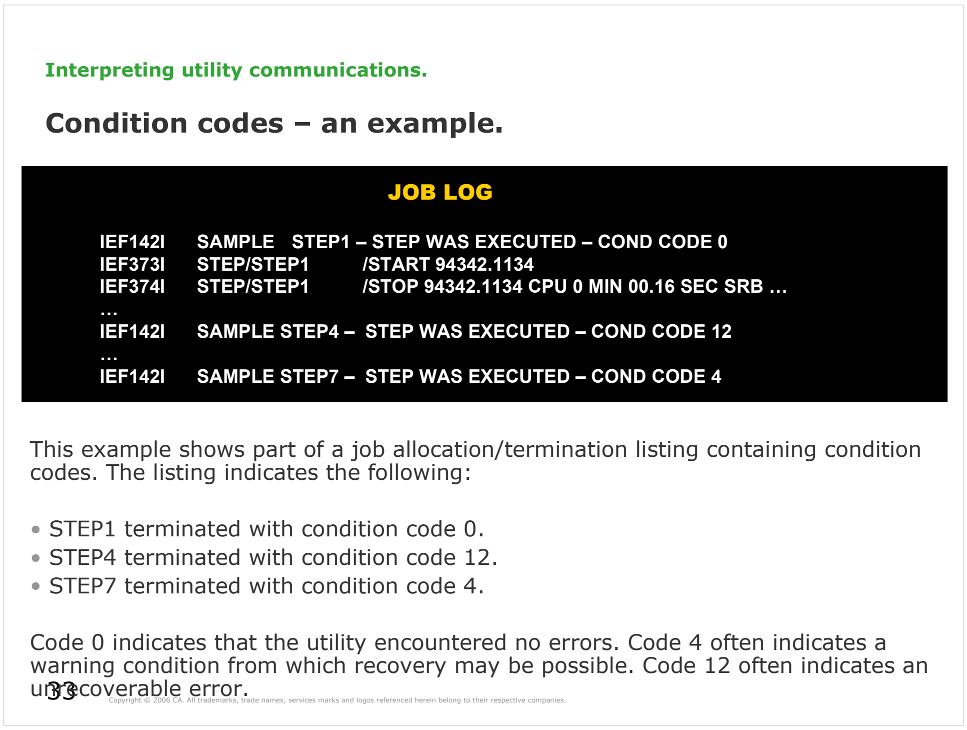
Kinds of condition codes – non zero condition code.

Non Zero Condition Code indicates that the utility had difficulty in trying to do the processing you requested.

The meaning of the non-zero condition code varies with the utility that produced it.



32



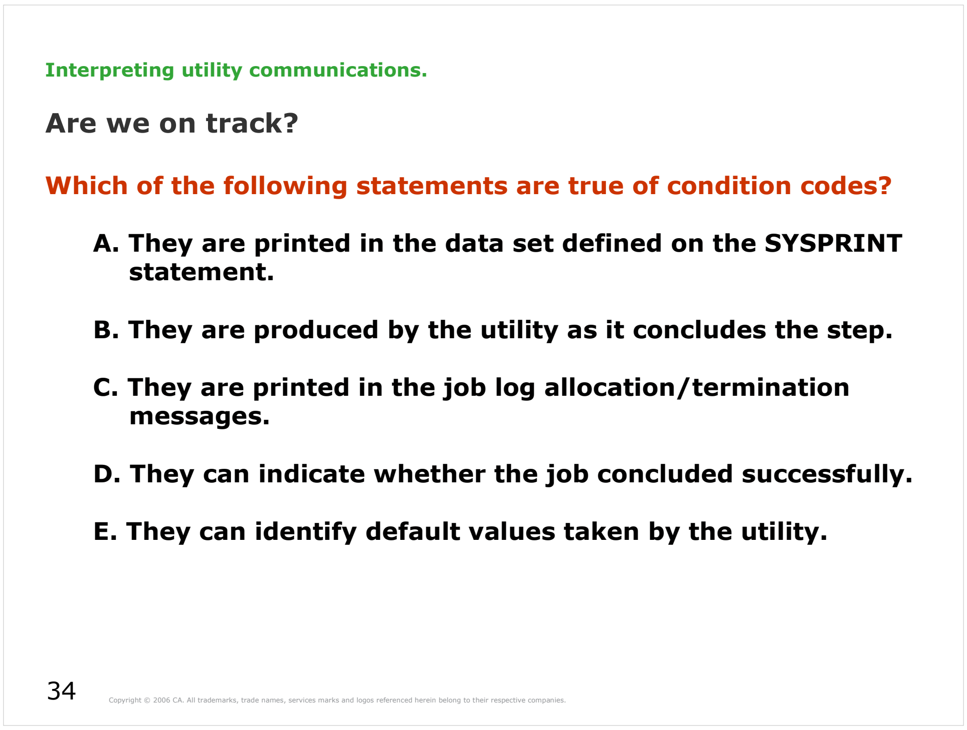
Condition codes – an example.

```
                          JOB LOG

IEF142I   SAMPLE   STEP1 – STEP WAS EXECUTED – COND CODE 0
IEF373I   STEP/STEP1        /START 94342.1134
IEF374I   STEP/STEP1        /STOP 94342.1134 CPU 0 MIN 00.16 SEC SRB …
…
IEF142I   SAMPLE STEP4 –  STEP WAS EXECUTED – COND CODE 12
…
IEF142I   SAMPLE STEP7 –  STEP WAS EXECUTED – COND CODE 4
```

This example shows part of a job allocation/termination listing containing condition codes. The listing indicates the following:

- STEP1 terminated with condition code 0.
- STEP4 terminated with condition code 12.
- STEP7 terminated with condition code 4.

Code 0 indicates that the utility encountered no errors. Code 4 often indicates a warning condition from which recovery may be possible. Code 12 often indicates an unrecoverable error.



Interpreting utility communications.

Are we on track?

Which of the following statements are true of condition codes?

A. They are printed in the data set defined on the SYSPRINT statement.

B. They are produced by the utility as it concludes the step.

C. They are printed in the job log allocation/termination messages.

D. They can indicate whether the job concluded successfully.

E. They can identify default values taken by the utility.

34

The correct answer is B., C., and D.

Interpreting utility communications.

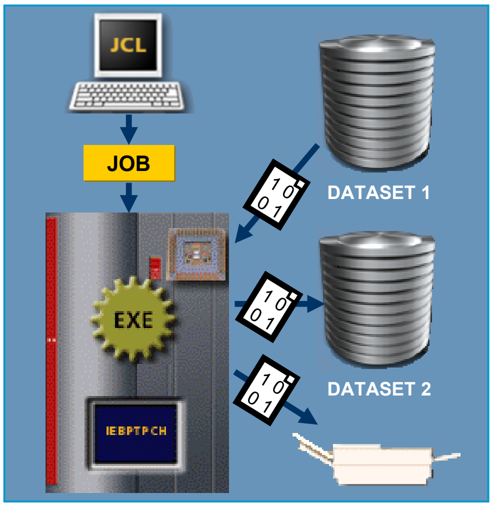
Testing condition codes.

How to test condition codes?

The system tests a condition code when the job is executed if you code a COND parameter on the JOB or EXEC statements.

You can alter your job's processing based on the utility's concluding condition code.

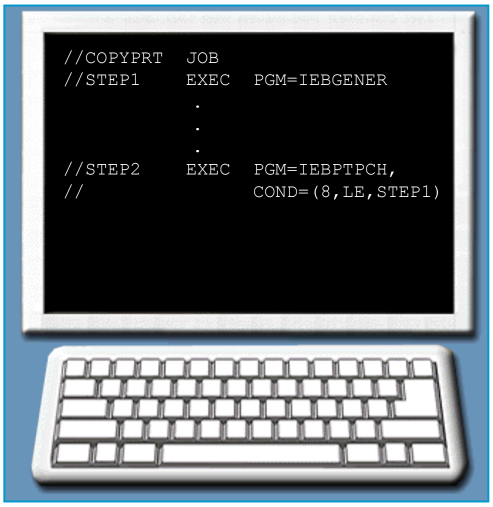
For example, suppose you want to copy a sequential data set to a new sequential data set (using the IEBGENER utility). Then, if the copy is successful, you want to print the new data set (using the IEBPTPCH utility). Otherwise, you do not want to print any data.



Testing condition codes – an example.

Consider IEBGENER to be successful if it concludes with a condition code of 0 or 4; that is, a condition code less than 8.

You would code the JCL as shown on the right. It specifies that STEP2 is only to be executed if STEP1 terminates successfully. That is, STEP2 is executed if STEP1 produces a condition code less than 8.



```
//COPYPRT   JOB
//STEP1     EXEC   PGM=IEBGENER
             .
             .
             .
//STEP2     EXEC   PGM=IEBPTPCH,
//                 COND=(8,LE,STEP1)
```

Continued…

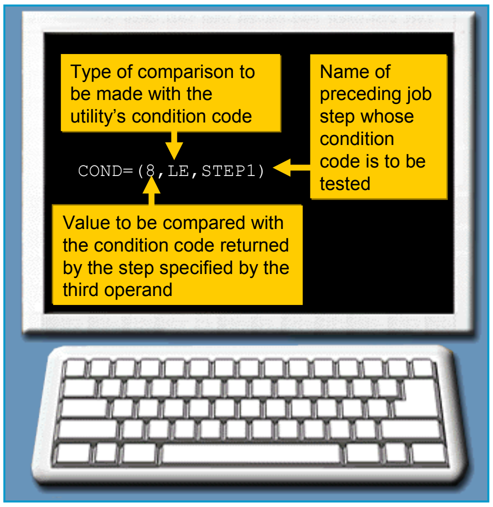
Interpreting utility communications.

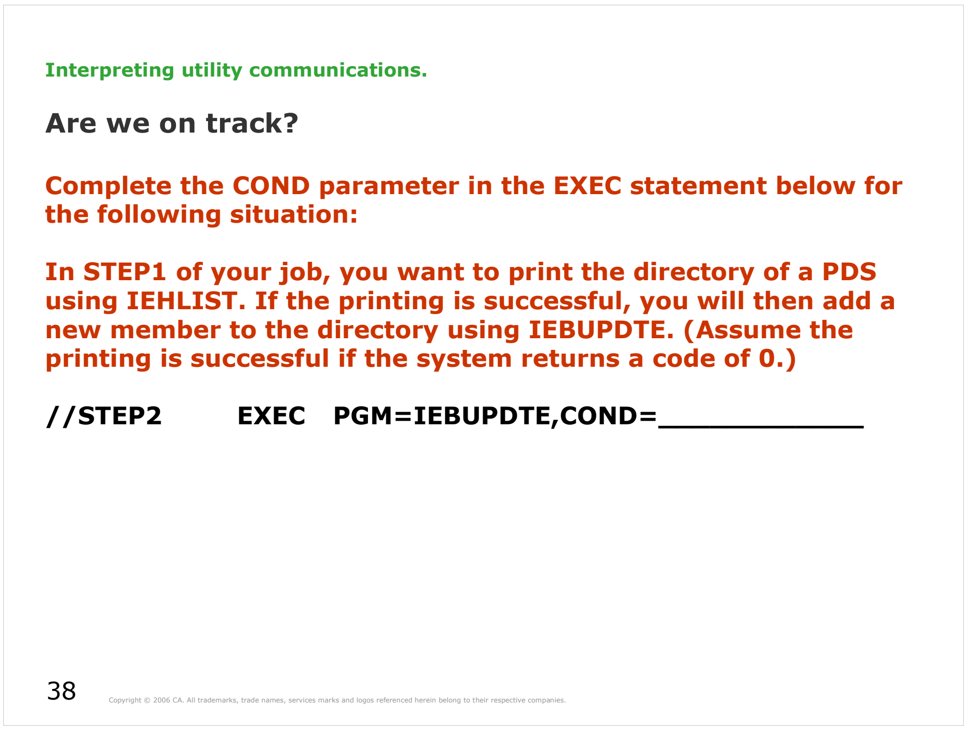
Testing condition codes – an example.

The system interprets the COND parameter as follows:

If 8 is less than or equal to (LE) the condition code returned by STEP1, do not execute this step (containing the COND parameter).

Thus, STEP2 will execute only if STEP1 concludes with a condition code less than 8 (0 or 4).


Type of comparison to be made with the utility's condition code
Name of preceding job step whose condition code is to be tested
COND=(8,LE,STEP1)
Value to be compared with the condition code returned by the step specified by the third operand



Interpreting utility communications.

Are we on track?

Complete the COND parameter in the EXEC statement below for the following situation:

In STEP1 of your job, you want to print the directory of a PDS using IEHLIST. If the printing is successful, you will then add a new member to the directory using IEBUPDTE. (Assume the printing is successful if the system returns a code of 0.)

```
//STEP2        EXEC   PGM=IEBUPDTE,COND=____________
```

38

The correct answer is (4,LE,STEP1)

Interpreting utility communications.

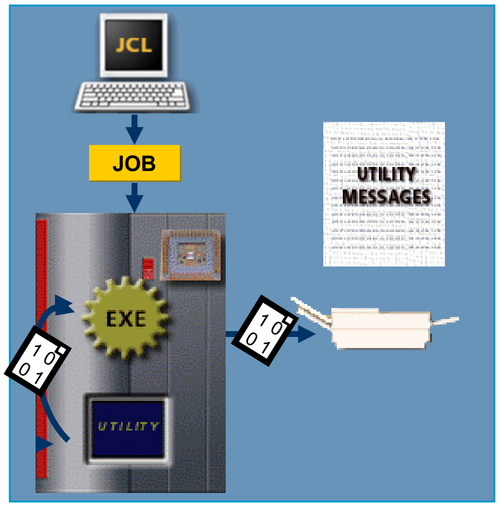
Utility messages.

Each utility also creates utility messages. The messages are printed in the SYSPRINT data set. SYSPRINT output also includes the submitted control statements.

Some of the utility messages are informational and fairly self-explanatory. These utility messages usually do not have a message number associated with them. If the utility produces only informational types of messages, it continues its processing.

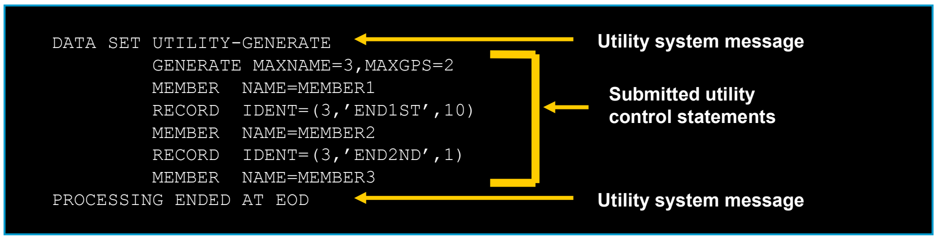
Informational utility messages can identify:

- Assumptions made by the utility.
- Default values taken by the utility.



Interpreting utility communications.

Utility messages – an example.



```
DATA SET UTILITY-GENERATE
         GENERATE MAXNAME=3,MAXGPS=2
         MEMBER   NAME=MEMBER1
         RECORD   IDENT=(3,'END1ST',10)
         MEMBER   NAME=MEMBER2
         RECORD   IDENT=(3,'END2ND',1)
         MEMBER   NAME=MEMBER3
PROCESSING ENDED AT EOD
```

Here is an example of SYSPRINT output after the IEBGENER utility successfully completed the task. The output indicates the following:

- Utility GENERATE (PGM=IEBGENER) was executed.
- The utility terminated normally. "PROCESSING ENDED AT EOD" (end-of-data) indicates the utility terminated after encountering end-of-file (EOF) on the input data set defined by the //SYSUT1 DD statement. The main indication that the processing completed normally is that there are no error messages printed.

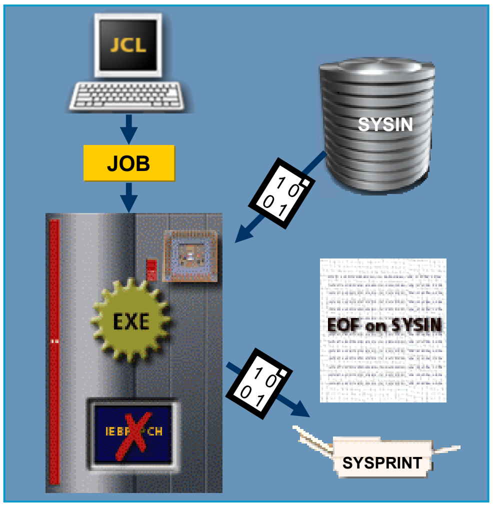
Interpreting utility communications.

Utility error messages.

Utility error messages can also be included in SYSPRINT output, which indicate that the utility encountered problems. The job may terminate, depending on the severity of the error.

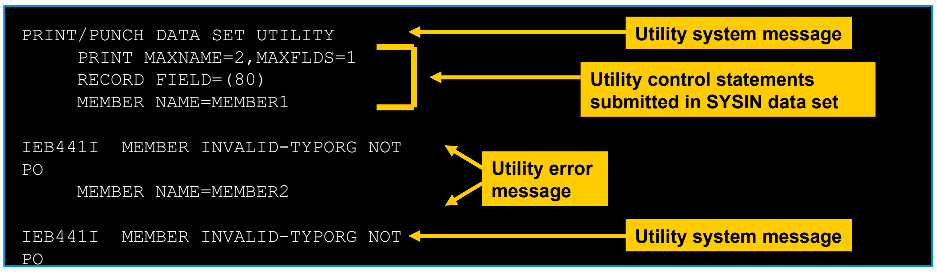
Error and warning messages display a message number, which enables you to look up the numbered message in the Utility Message Manual to find more information about the condition detected.

This will help in determining the source of the error and the correction required to fix it.



Interpreting utility communications.

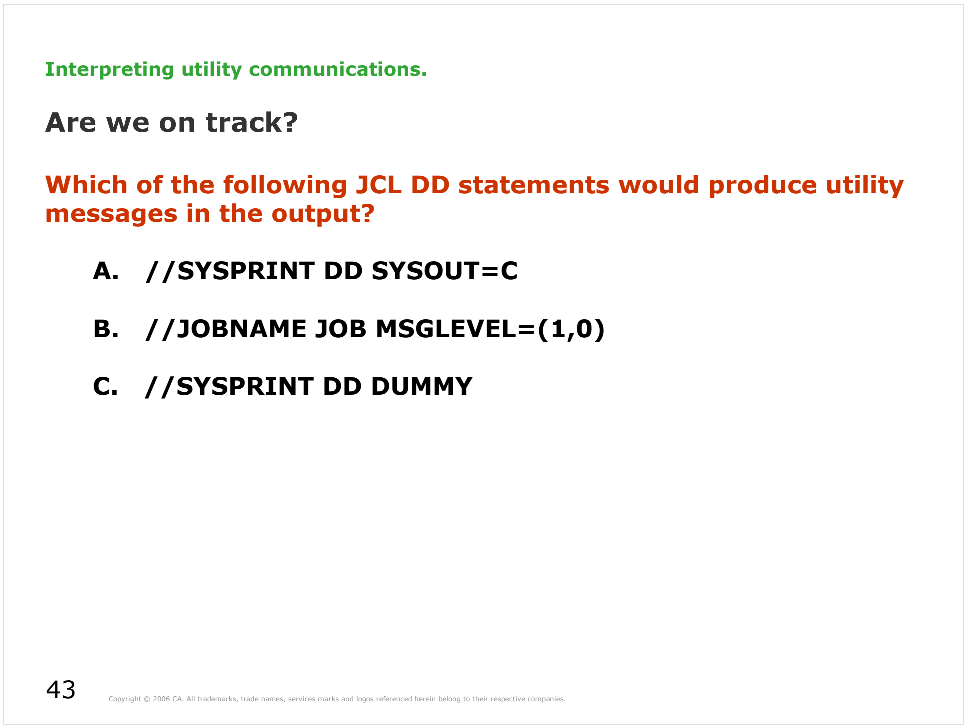
Utility error messages – an example.



```
PRINT/PUNCH DATA SET UTILITY
     PRINT MAXNAME=2,MAXFLDS=1
     RECORD FIELD=(80)
     MEMBER NAME=MEMBER1

IEB441I  MEMBER INVALID-TYPORG NOT
PO
     MEMBER NAME=MEMBER2

IEB441I  MEMBER INVALID-TYPORG NOT
PO
```

Here is an example of SYSPRINT output when the IEBPTPCH utility encountered an error in the JCL and utility control statements. The output indicates the following:

- The utility PRINT/PUNCH (PGM=IEBPTCH) was executed.
- The utility did not perform the required task as indicated by the error messages (IEB441I). In addition, the utility indicates that the EOF was reached on the control data set (SYSIN) while the utility was searching for additional utility control statements.



Interpreting utility communications.

Are we on track?

Which of the following JCL DD statements would produce utility messages in the output?

A. //SYSPRINT DD SYSOUT=C

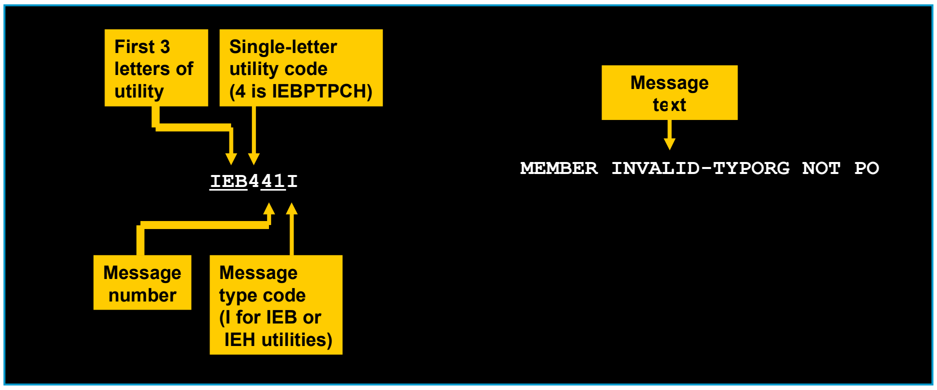
B. //JOBNAME JOB MSGLEVEL=(1,0)

C. //SYSPRINT DD DUMMY

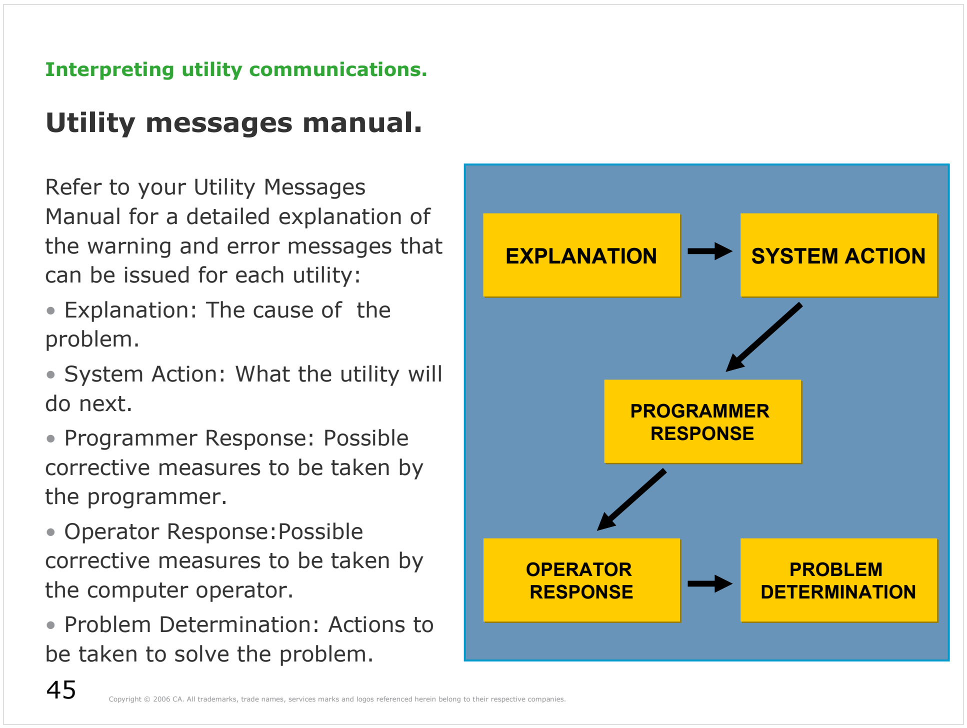
43

The correct answer is A.

Utility error message – general format.


First 3 letters of utility
Single-letter utility code (4 is IEBPTPCH)
Message text
IEB441I
MEMBER INVALID-TYPORG NOT PO
Message number
Message type code (I for IEB or IEH utilities)

The general format for utility error messages is shown above.

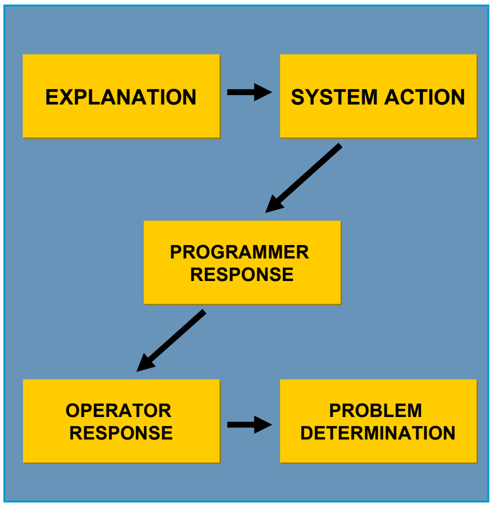


Interpreting utility communications.

Utility messages manual.

Refer to your Utility Messages Manual for a detailed explanation of the warning and error messages that can be issued for each utility:

- Explanation: The cause of the problem.
- System Action: What the utility will do next.
- Programmer Response: Possible corrective measures to be taken by the programmer.
- Operator Response:Possible corrective measures to be taken by the computer operator.
- Problem Determination: Actions to be taken to solve the problem.



45

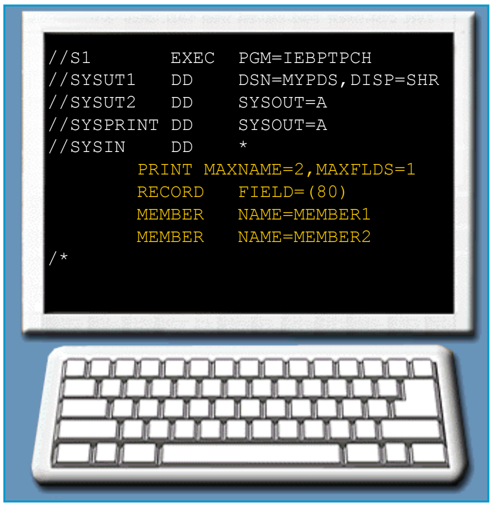
See „MVS System Messages Volume 7 (IEB - IEE)“ or QuickRef application: qw.

Summary example.

Interpreting utility messages.

The image shows a sample job stream, illustrating the process of interpreting a utility error message and correcting the problem.

The job stream will print MEMBER1 and MEMBER2 of a cataloged partitioned data set named MYPDS, using the utility IEBPTPCH.



```
//S1         EXEC   PGM=IEBPTPCH
//SYSUT1     DD     DSN=MYPDS,DISP=SHR
//SYSUT2     DD     SYSOUT=A
//SYSPRINT DD     SYSOUT=A
//SYSIN      DD     *
        PRINT MAXNAME=2,MAXFLDS=1
        RECORD    FIELD=(80)
        MEMBER    NAME=MEMBER1
        MEMBER    NAME=MEMBER2
/*
```

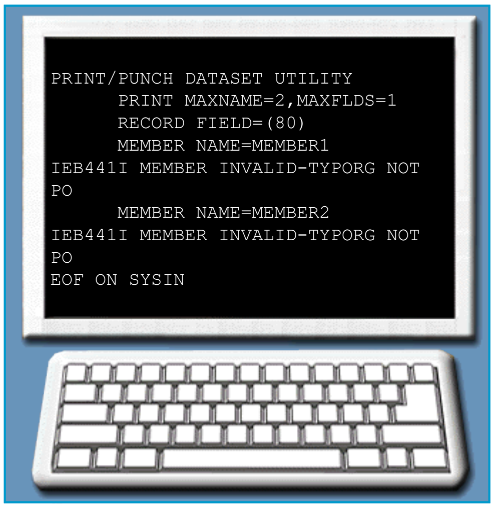
Continued…

Summary example.

Interpreting utility messages.

The job does not complete successfully. The contents of the SYSPRINT output are shown on the right.

The error message is IEB441I MEMBER INVALID.



```
PRINT/PUNCH DATASET UTILITY
      PRINT MAXNAME=2,MAXFLDS=1
      RECORD FIELD=(80)
      MEMBER NAME=MEMBER1
IEB441I MEMBER INVALID-TYPORG NOT
PO
      MEMBER NAME=MEMBER2
IEB441I MEMBER INVALID-TYPORG NOT
PO
EOF ON SYSIN
```

Continued…

Summary example.

Interpreting utility messages.

```
IEB441I MEMBER INVALID: TYPORG NOT PO
```

Explanation: The MEMBER statement preceding this message is incorrect since physical sequential (PS) organization was specified.
That is, TYPORG=PO must be specified on the PRINT or PUNCH utility control statement.

Source: DFSMSdfp

System Action: The program is ended at the end of the control statement scan. The return code is 12.

Application Programmer Response: Probable user error. If SYSUT1 specifies a physical sequential data set, remove the MEMBER statement. If SYSUT1 specifies a partitioned data set, specify TYPORG=PO on the PRINT or PUNCH statement.

The above shows the explanation given by the Utility System Messages Manual for message IEB441I.

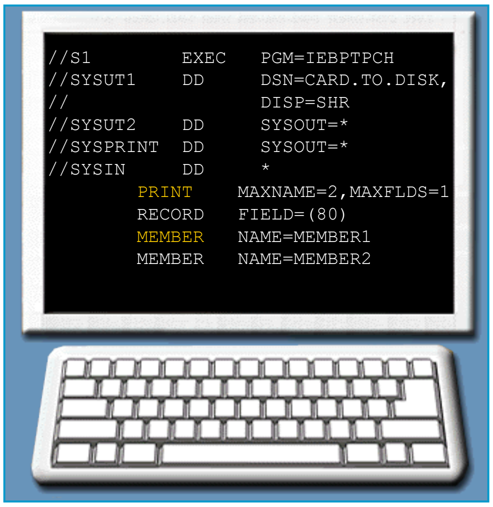
Summary example.

Interpreting utility messages – error condition.

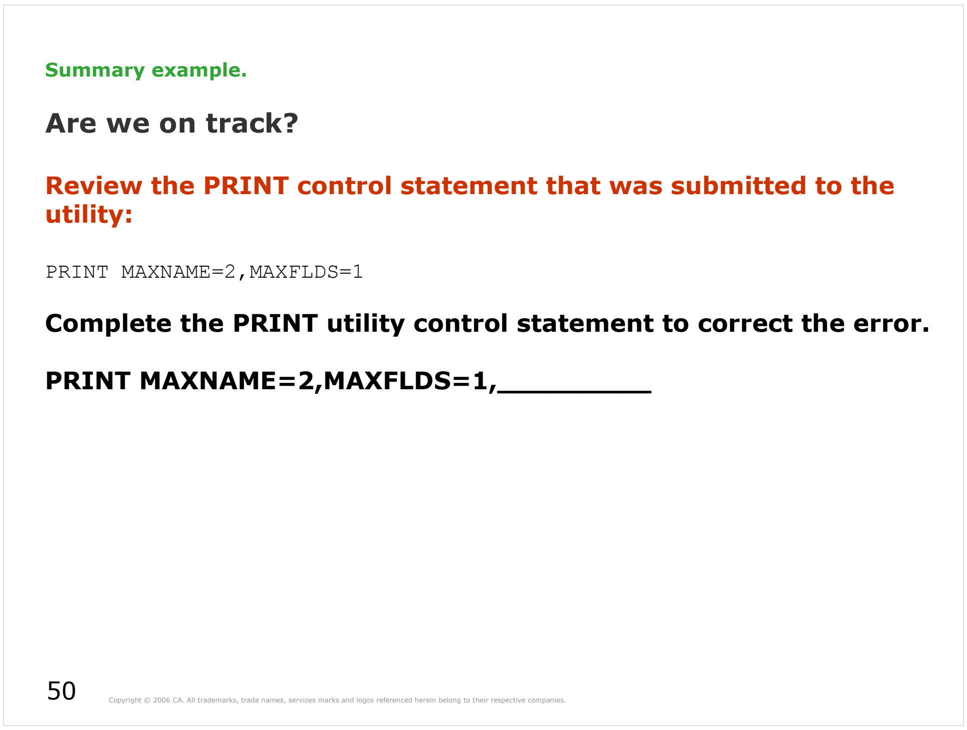
The error condition is not caused by any single statement, but rather by two control statements with conflicting information:

1. The MEMBER utility control statement in the SYSIN data set indicates you are processing members of a partitioned data set.

2. For a partitioned data set, you must specify TYPORG=PO on the PRINT control statement.



```
//S1          EXEC    PGM=IEBPTPCH
//SYSUT1      DD      DSN=CARD.TO.DISK,
//                    DISP=SHR
//SYSUT2      DD      SYSOUT=*
//SYSPRINT    DD      SYSOUT=*
//SYSIN       DD      *
        PRINT     MAXNAME=2,MAXFLDS=1
        RECORD    FIELD=(80)
        MEMBER    NAME=MEMBER1
        MEMBER    NAME=MEMBER2
```



Summary example.

Are we on track?

Review the PRINT control statement that was submitted to the utility:

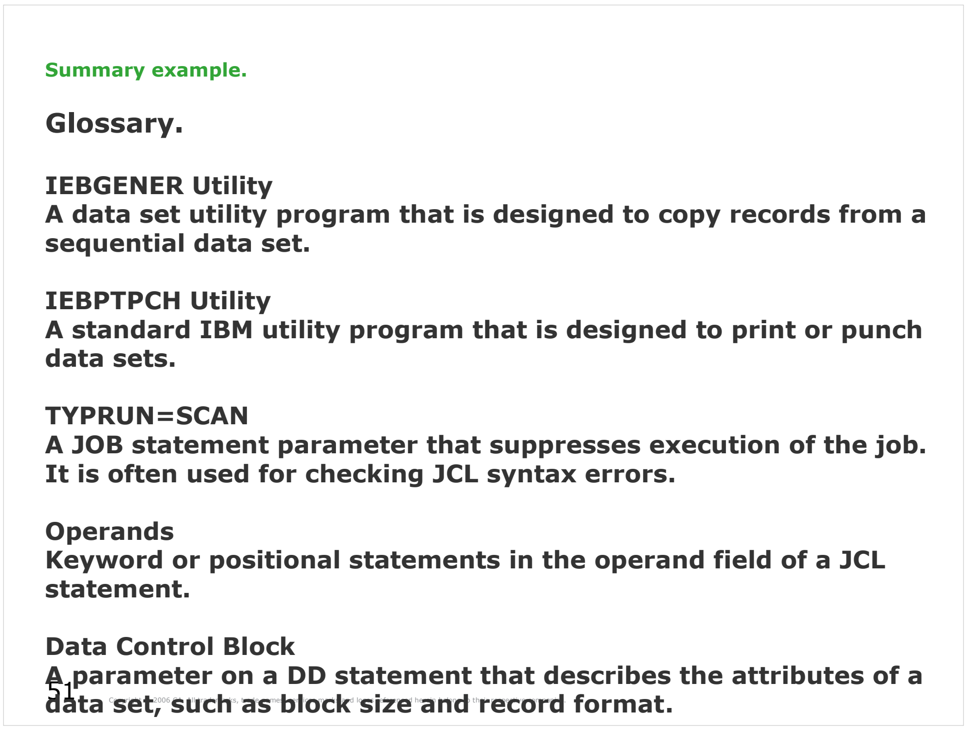
```
PRINT MAXNAME=2,MAXFLDS=1
```

Complete the PRINT utility control statement to correct the error.

PRINT MAXNAME=2,MAXFLDS=1,__________

50

The correct answer is TYPORG=PO



Summary example.

Glossary.

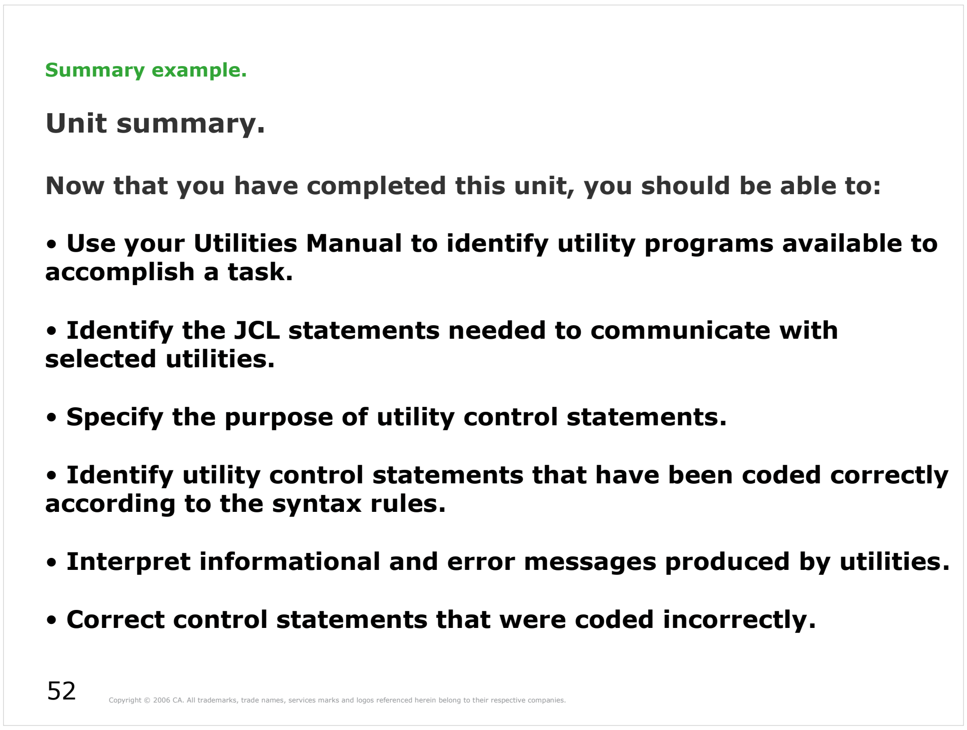
IEBGENER Utility
A data set utility program that is designed to copy records from a sequential data set.

IEBPTPCH Utility
A standard IBM utility program that is designed to print or punch data sets.

TYPRUN=SCAN
A JOB statement parameter that suppresses execution of the job. It is often used for checking JCL syntax errors.

Operands
Keyword or positional statements in the operand field of a JCL statement.

Data Control Block
A parameter on a DD statement that describes the attributes of a data set, such as block size and record format.



Summary example.

Unit summary.

Now that you have completed this unit, you should be able to:

- **Use your Utilities Manual to identify utility programs available to accomplish a task.**
- **Identify the JCL statements needed to communicate with selected utilities.**
- **Specify the purpose of utility control statements.**
- **Identify utility control statements that have been coded correctly according to the syntax rules.**
- **Interpret informational and error messages produced by utilities.**
- **Correct control statements that were coded incorrectly.**

52