JCL

Chapter c4
Sample utility application

Job Control Language

Sample utility application.

Chapter c4

Sample utility application

Unit introduction.

This unit analyzes examples of typical processing tasks that can be accomplished with utilities.

Although available utilities, required JCL, and control statements will vary with different installations, this unit provides a sampling of the tasks that utilities can perform.

The material also introduces a process that can be applied in choosing and executing any utility for processing needs.

Course objectives.

Be able to:

- Determine the job stream.
- Print sequential data sets.
- Maintain source libraries.
- List PDS directories.

Determining the job stream.

Utility programs can be used in a variety of situations to provide the processing required.

How to determine job streams?

An effective method for determining the job stream that will accomplish the necessary processing is as follows:

1. Use the Utilities Manual table to find the utility that may be appropriate to accomplish the task.

2. Once an appropriate utility or utilities is selected, read the chapters of the Utilities Manual that describe the utility in detail.

3. Look for an example that approximates the processing requirements in the Utilities Manual. Modify the example (as needed) to accomplish the task.

Determining the job stream.

In some cases, more than one utility will provide the processing required. Select any one of the utilities that will do the job. The table below provides a list of utilities that may be used to process a job/task.

JOB	UTILITIES
Change data set organization	IEBGENER, IEBUPDTE, IEBPTPCH
Convert to sequential data set	IEBGENER, IEBUPDTE
Copy a partitioned data set	IEBCOPY
Copy a sequential data set	IEBGENER, IEBUPDTE, IEBPTPCH
Create a sequential output data set	IEBDG, IEBGENER, IEBPTPCH
Edit and copy a sequential data set	IEBGENER, IEBUPDTE, IEBPTPCH
Merge partitioned data sets	IEBCOPY
Replace selected members of a PDSE	IEBCOPY, IEBUPDTE

Determining the job stream.

Are we on track?

Put the following steps in order to reflect the suggested method of selecting and executing a utility program:

A. Find an appropriate example application.

B. Look up a utility that will perform the task.

C. Read the chapter that describes the utility.

D. Modify the code in the example as needed.

Print sequential data sets.

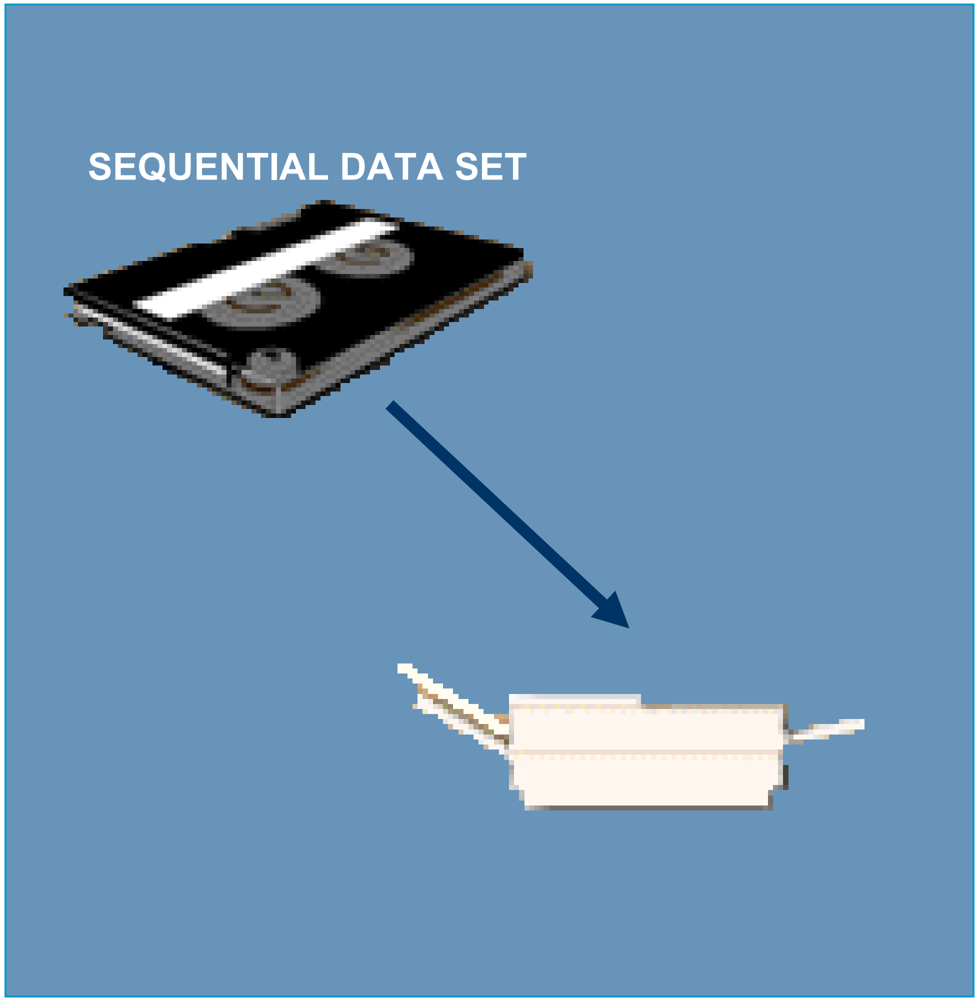
Selecting a utility.

How to select a utility?

Assume you need to list the contents of a data set in order to determine what it contains. The data set resides on a tape volume, as shown on the right. In this case, consider the following:

1. The need to print the contents of a data set.

2. The data set resides on a tape volume, which implies a sequential data set.

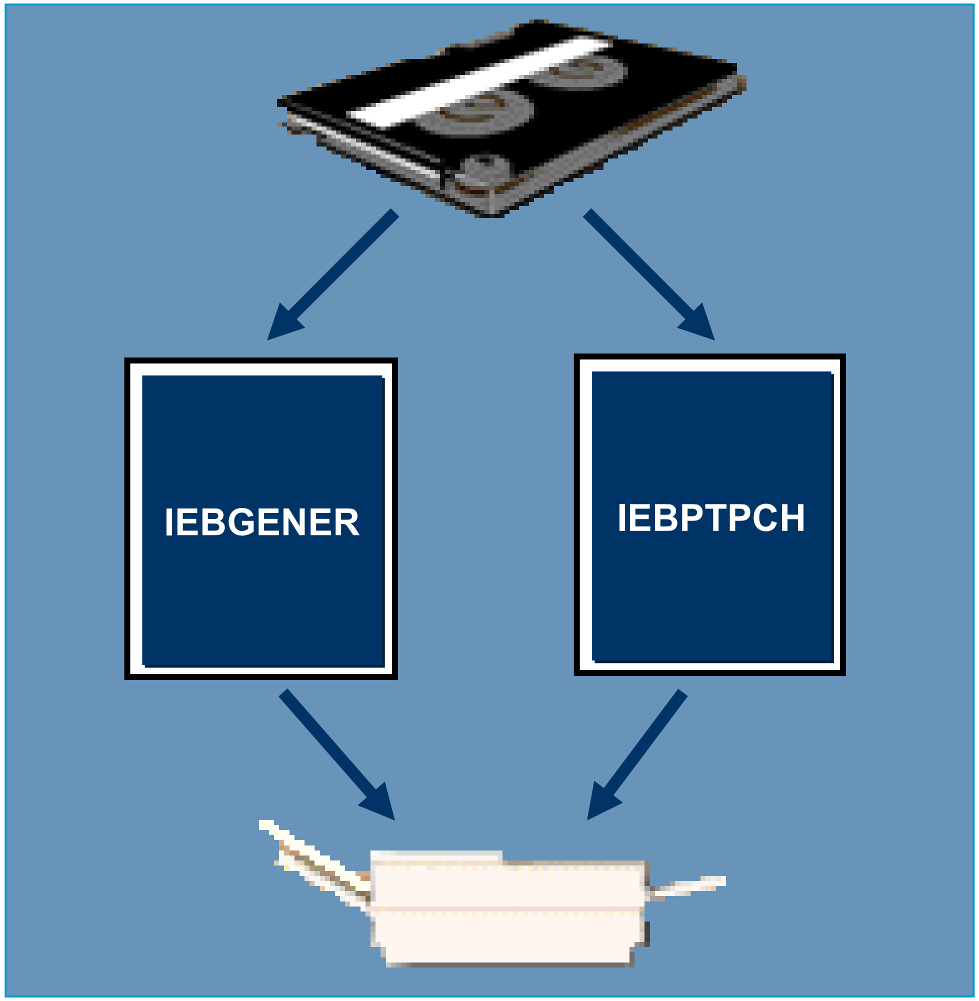


Print sequential data sets.

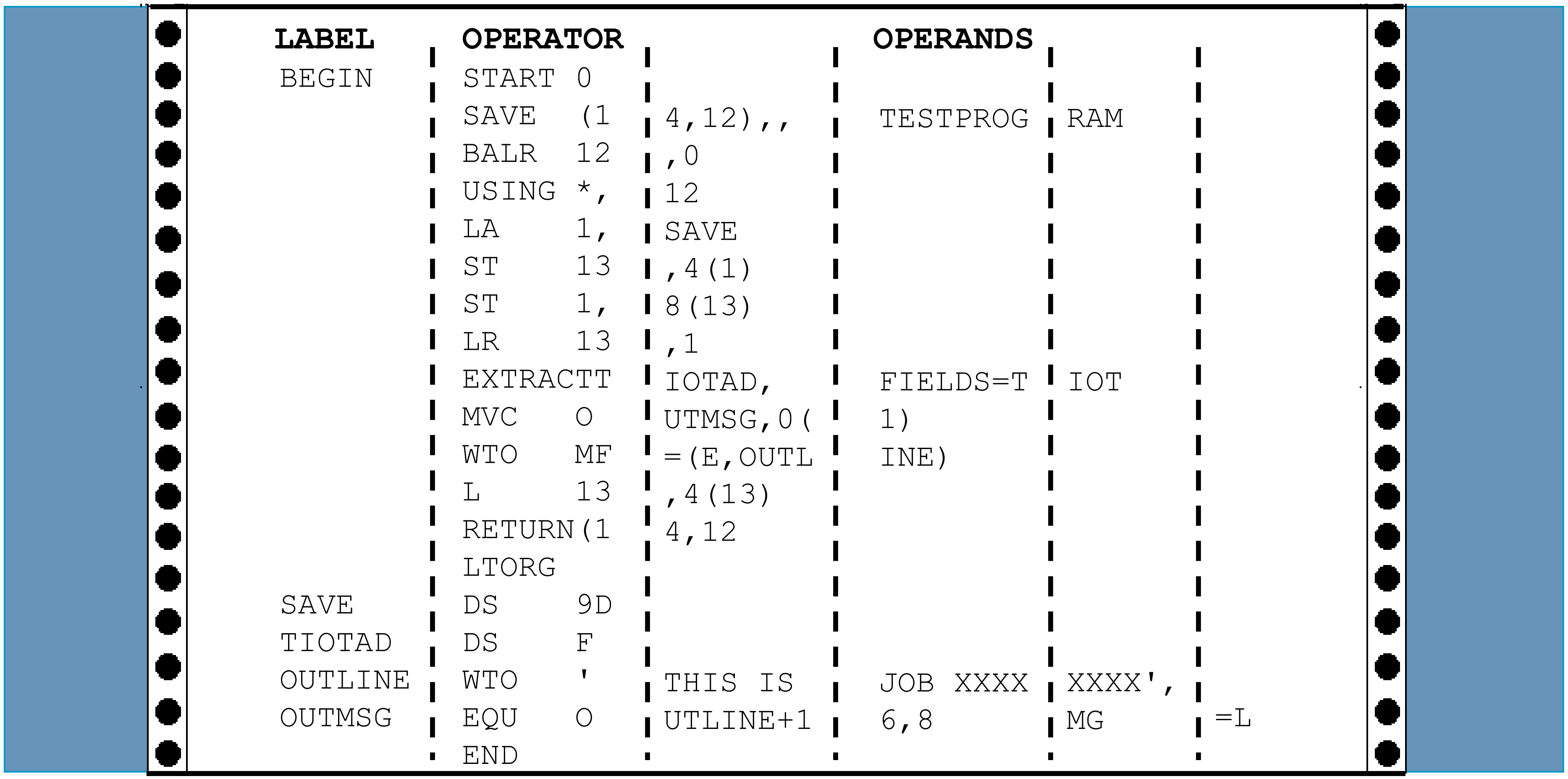
Kinds of utilities.

The utilities available to print data sets are as follows:

- IEBGENER – useful when you want to copy, edit, or print the contents of a sequential data set.
- IEBUPDTE – can be used to create and modify PDS members.
- IEBPTPCH – can also be used to print the contents of a data set.



IEBPTPCH output.



```
LABEL      OPERATOR               OPERANDS
BEGIN    | START 0  |          |          |          |
         | SAVE  (1 | 4,12),,  | TESTPROG | RAM      |
         | BALR  12 | ,0       |          |          |
         | USING *, | 12       |          |          |
         | LA    1, | SAVE     |          |          |
         | ST    13 | ,4(1)    |          |          |
         | ST    1, | 8(13)    |          |          |
         | LR    13 | ,1       |          |          |
         | EXTRACTT | IOTAD,   | FIELDS=T | IOT      |
         | MVC   O  | UTMSG,0( | 1)       |          |
         | WTO   MF | =(E,OUTL | INE)     |          |
         | L     13 | ,4(13)   |          |          |
         | RETURN(1 | 4,12     |          |          |
         | LTORG    |          |          |          |
SAVE     | DS    9D |          |          |          |
TIOTAD   | DS    F  |          |          |          |
OUTLINE  | WTO   '  | THIS IS  | JOB XXXX | XXXX',   |
OUTMSG   | EQU   O  | UTLINE+1 | 6,8      | MG       | =L
         | END      |          |          |          |
```

IEBPTCH has a unique output format. The data set in the example would print as shown above.

Print sequential data sets.

Are we on track?

The ________________ utility is useful when you want to copy, edit, or print the contents of a sequential data set.

Print sequential data sets.

JCL statements for IEBGENER.

```
//PRINT     JOB
//STEP1     EXEC    PGM=IEBGENER
//SYSPRINT DD       SYSOUT=A
//SYSIN     DD      DUMMY
//SYSUT1    DD      DDNAME=MYDATA,DISP=OLD,UNIT=TAPE,VOL=SER=112233
//SYSUT2    DD      SYSOUT=A
```

The JCL statements required for IEBGENER, as shown above, are:

JOB – It identifies the job to the operating system.

EXEC – It specifies that the IEBGENER should be executed.

SYSPRINT – It indicates that the messages generated by IEBGENER are to be directed to a printer.

SYSIN – It is necessary for IEBGENER. However, this example does not require control statements, so the SYSIN statement specifies DUMMY.

SYSUT1 – It identifies the input data set.

SYSUT2 – It specifies the output data set of the PRINT operation.

Are we on track?

Complete the following SYSIN statement to indicate that no utility control statements are required to process a job using a utility:

```
//SYSIN   DD ________________
```

Print sequential data sets.

Are we on track?

For this programming exercise, assume you want to copy an in-stream input, sequential data set to a tape volume. Using IEBGENER complete the JCL statements on the screen as follows:

In line 1, execute IEBGENER.
In line 2, create an output tape data set named TSTTAPE.
In line 4, specify that the output records are 80 characters, blocked 40 records per block. The record format is FB.
In line 5, create utility messages in SYSOUT class A.

```
   //jobname    JOB
1. //STEPNAME  EXEC _____________
2. //SYSUT2      DD   DSN=__________
3. //                 VOL=SER=123456,UNIT=TAPE,
4. //                  DCB=(RECFM=FB,___________
5. //SYSPRINT   DD   _____________
6. //SYSIN       DD   DUMMY
7. //SYSUT1      DD   *
      in-stream data
   /*
```

Maintaining source libraries.

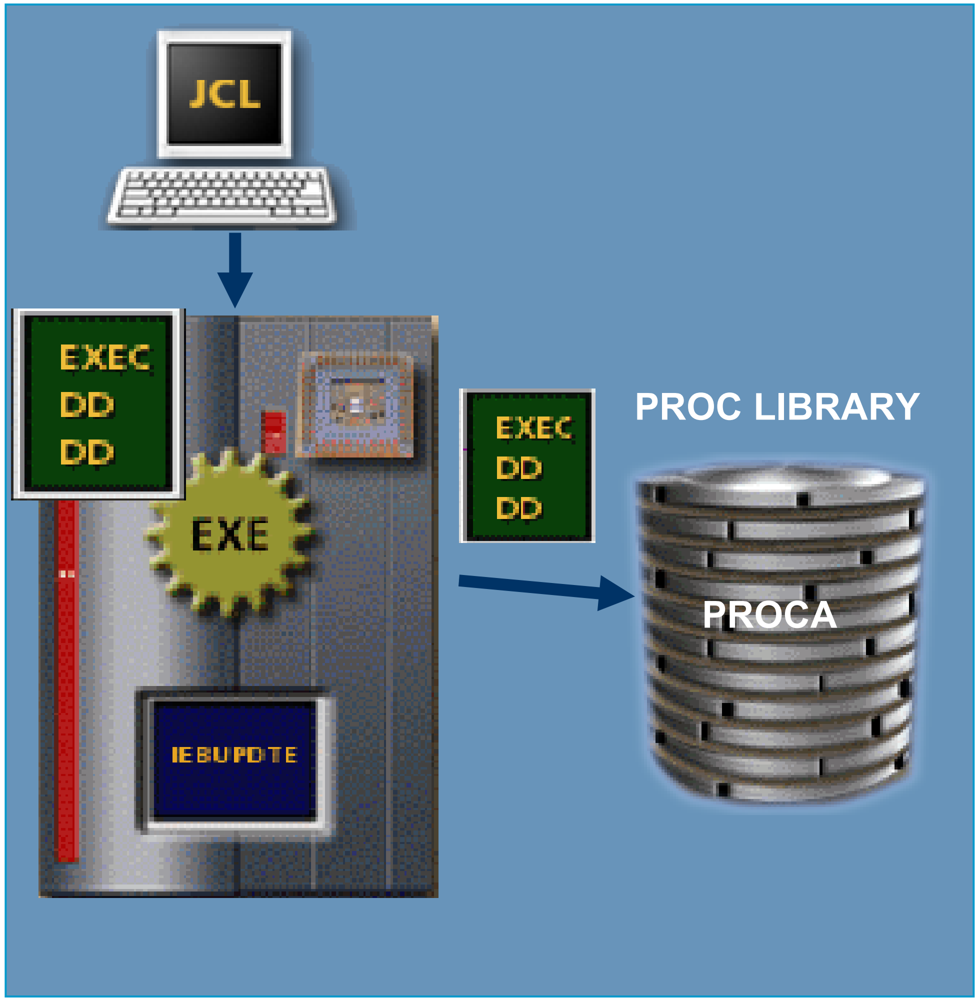
IEBUPDTE program.

The IEBUPDTE (Update Data Set) program can be used to create or modify sequential data sets or PDSs or PDSEs.

Why use IEBUPDTE?

IEBUPDTE is used primarily for updating procedure, source and macro libraries.

For the next example, assume a procedure requires to be stored in a procedure library and assign a sequence number to each created record.

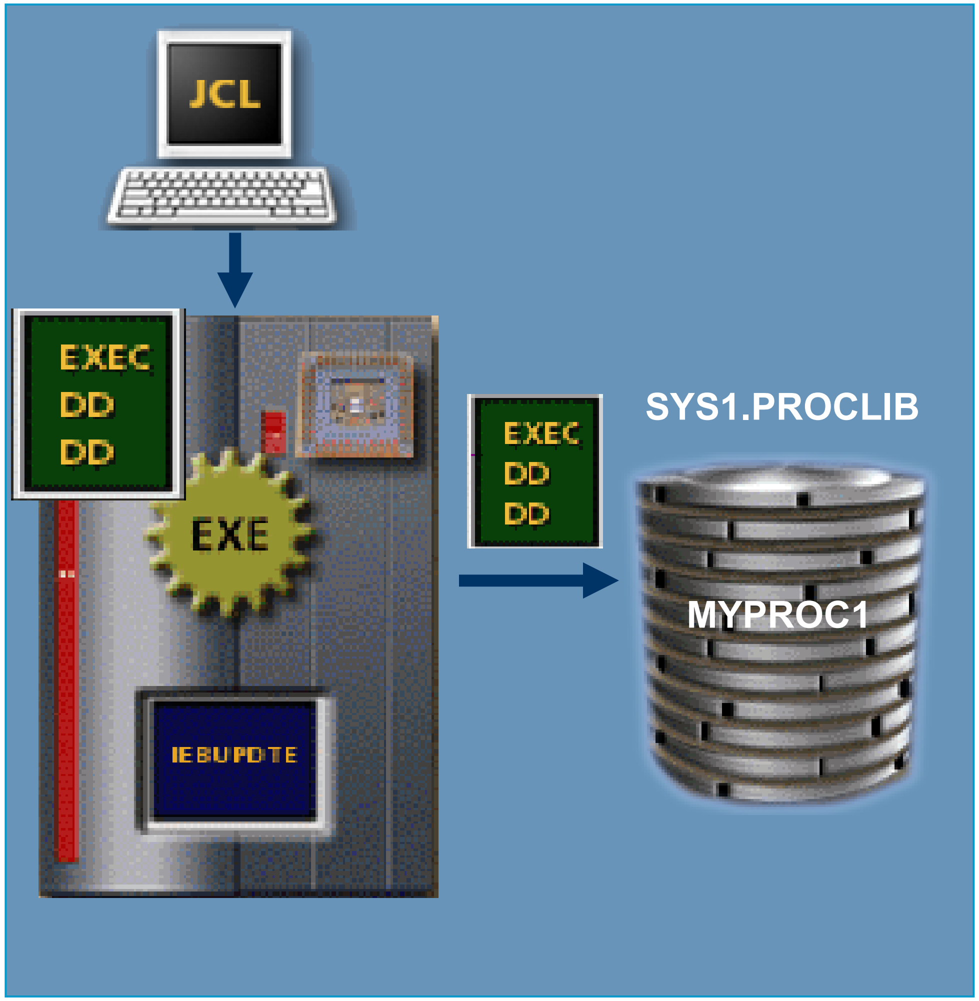


Maintaining source libraries.

IEBUPDTE program – an example.

Assume the processing requirements are as follows:

- The procedure will be entered in the job stream.
- The output data set is a procedure library named SYS1.PROCLIB.
- The new member will be named MYPROC1.
- A unique sequence number should be given to each record of the output member. The starting sequence number is to be 10 and the numbers should be incremented by 10.
- Each record should be listed as it is added to the PDS.

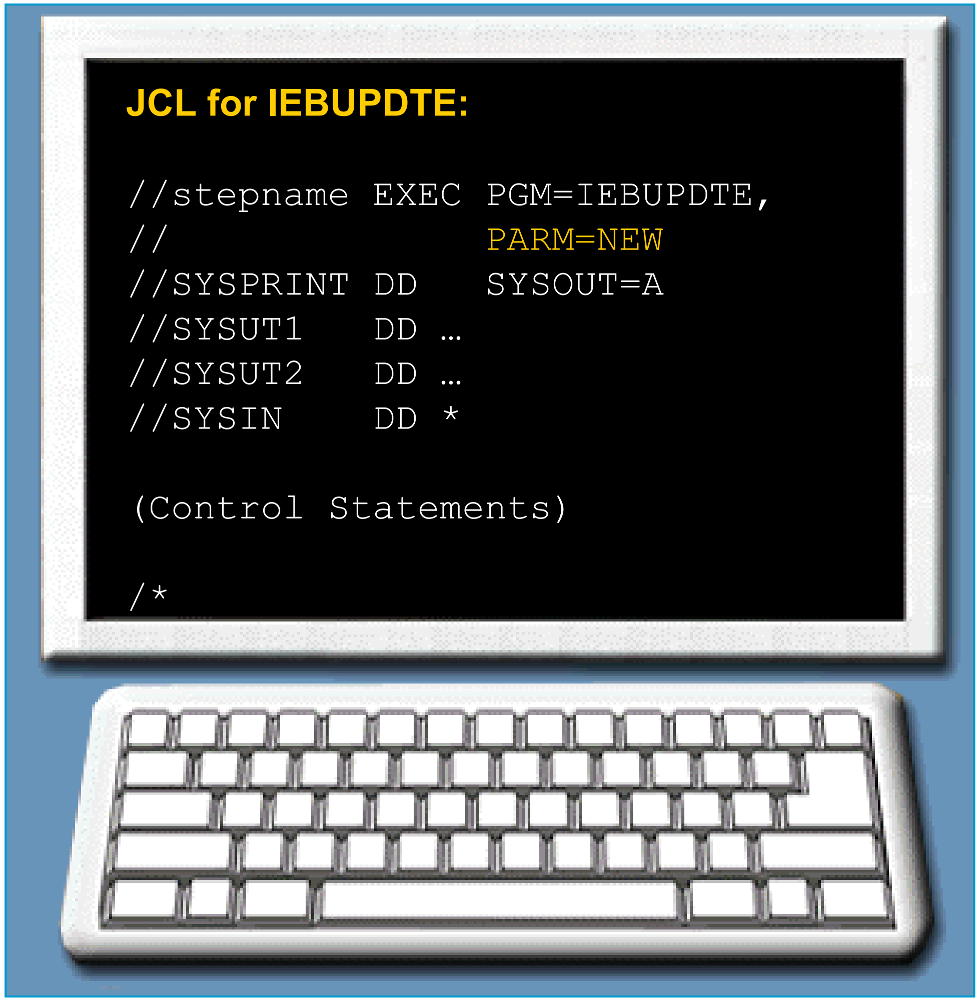


General form for IEBUPDTE.

The example on the right shows the general form for IEBUPDTE.

When adding a new member to an existing PDS, specify PARM=NEW on the EXEC statement invoking IEBUPDTE.

PARM=MOD would indicate that the output data set existed and has to be modified. MOD is the default if PARM is omitted.



JCL for IEBUPDTE:

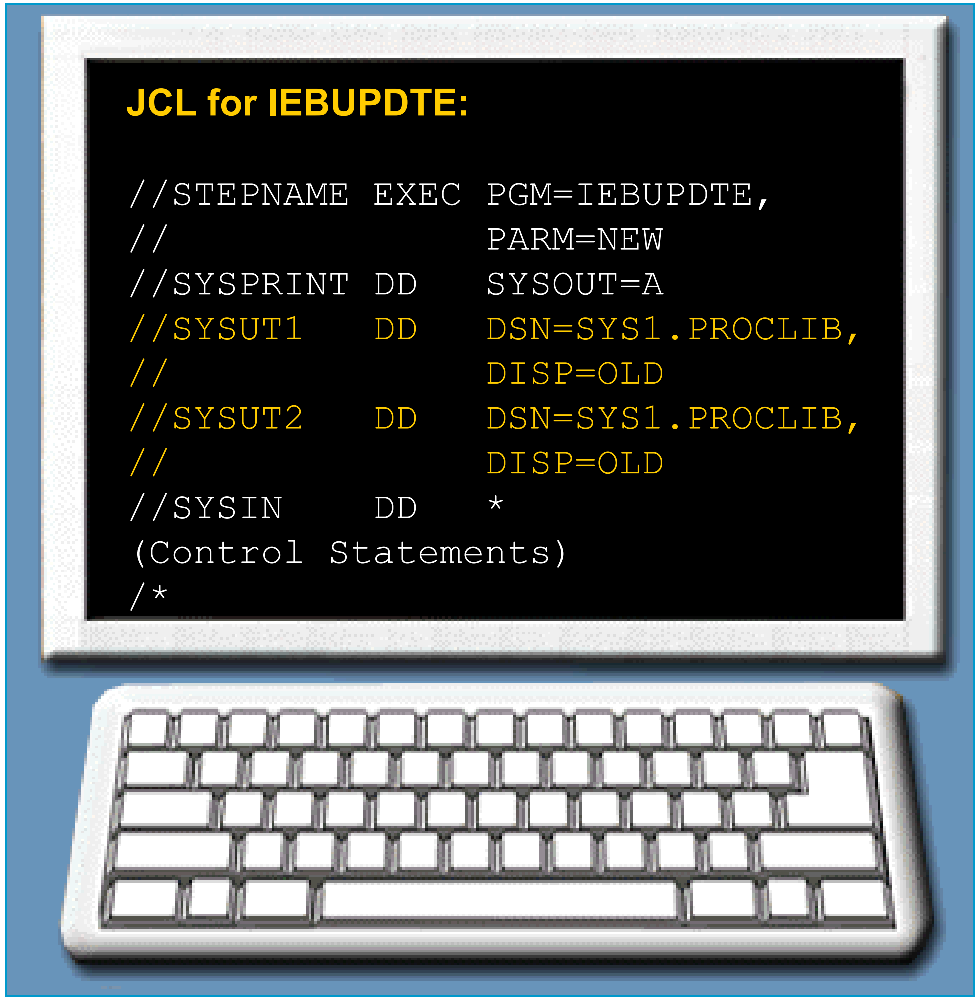
```
//stepname EXEC PGM=IEBUPDTE,
//              PARM=NEW
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD …
//SYSUT2   DD …
//SYSIN    DD *

(Control Statements)

/*
```

Updating a source library.

The JCL on the screen is used to update an existing library.

The SYSUT1 DD statement defines the PDS to be updated, and the SYSUT2 DD statement defines the PDS after it is updated.

The data set defined by these two DD statements is the same. Only its content is changed by the utility.



JCL for IEBUPDTE:

```
//STEPNAME EXEC PGM=IEBUPDTE,
//              PARM=NEW
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSN=SYS1.PROCLIB,
//              DISP=OLD
//SYSUT2   DD   DSN=SYS1.PROCLIB,
//              DISP=OLD
//SYSIN    DD   *
(Control Statements)
/*
```

Are we on track?

Code an EXEC statement below to:

Execute IEBUPDTE

Specify that an existing data set is to be altered:

//JOBSTEP EXEC ________________

Maintaining source libraries.

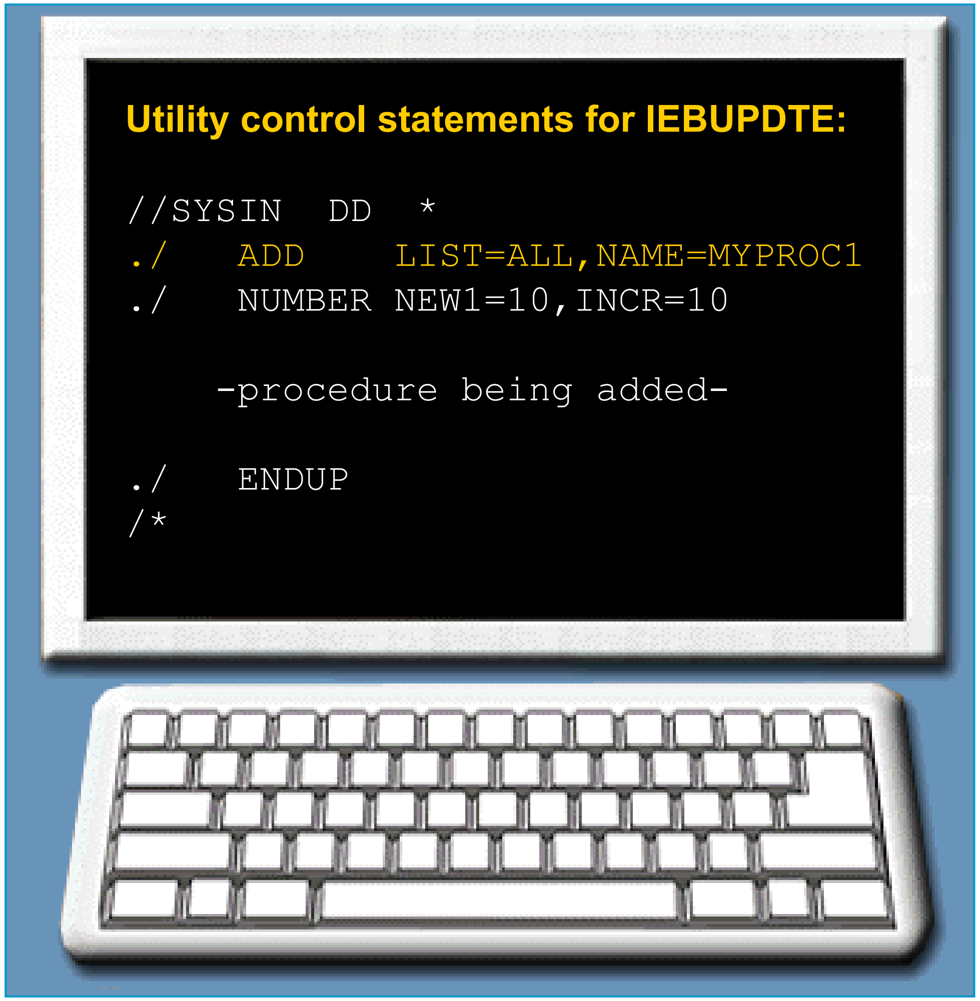
IEBUPDTE uses special utility control statements with the ./ characters in positions 1 and 2. These two characters distinguish the control statements from the input program that is also in the job stream.

The following control statements are required:

./ ADD indicates that a new member is to be added to the PDS.

LIST=ALL is an option to list the entire member on SYSPRINT.

NAME=names the newly created member is a procedure MYPROC1.



Utility control statements for IEBUPDTE:

```
//SYSIN   DD   *
./    ADD     LIST=ALL,NAME=MYPROC1
./    NUMBER NEW1=10,INCR=10

    -procedure being added-

./    ENDUP
/*
```

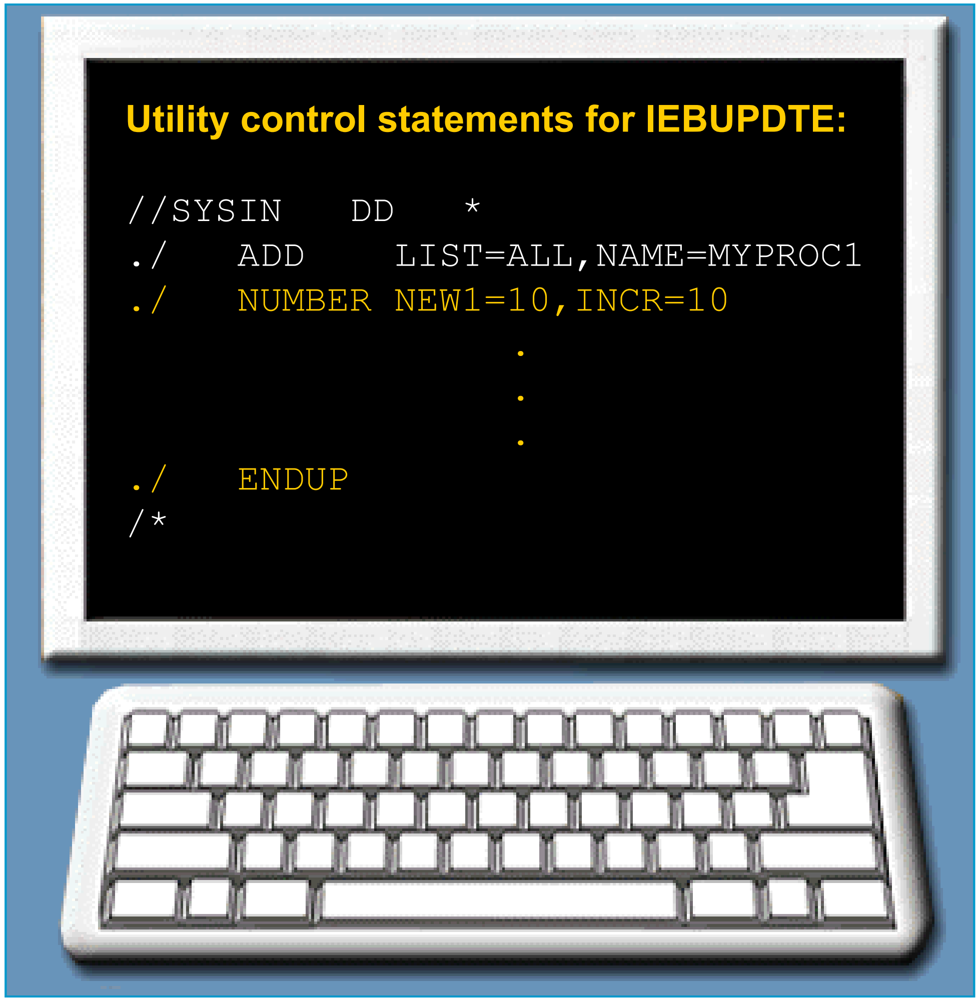
Maintainig source libraries.

./   NUMBER indicates that the new procedure is to be assigned sequence numbers.

NEW1= specifies the first sequence number (in this case, 10).

INCR=10 indicates that the records will be assigned sequence numbers in increments of 10 (10, 20,etc).

./   ENDUP marks the end of the member being added.



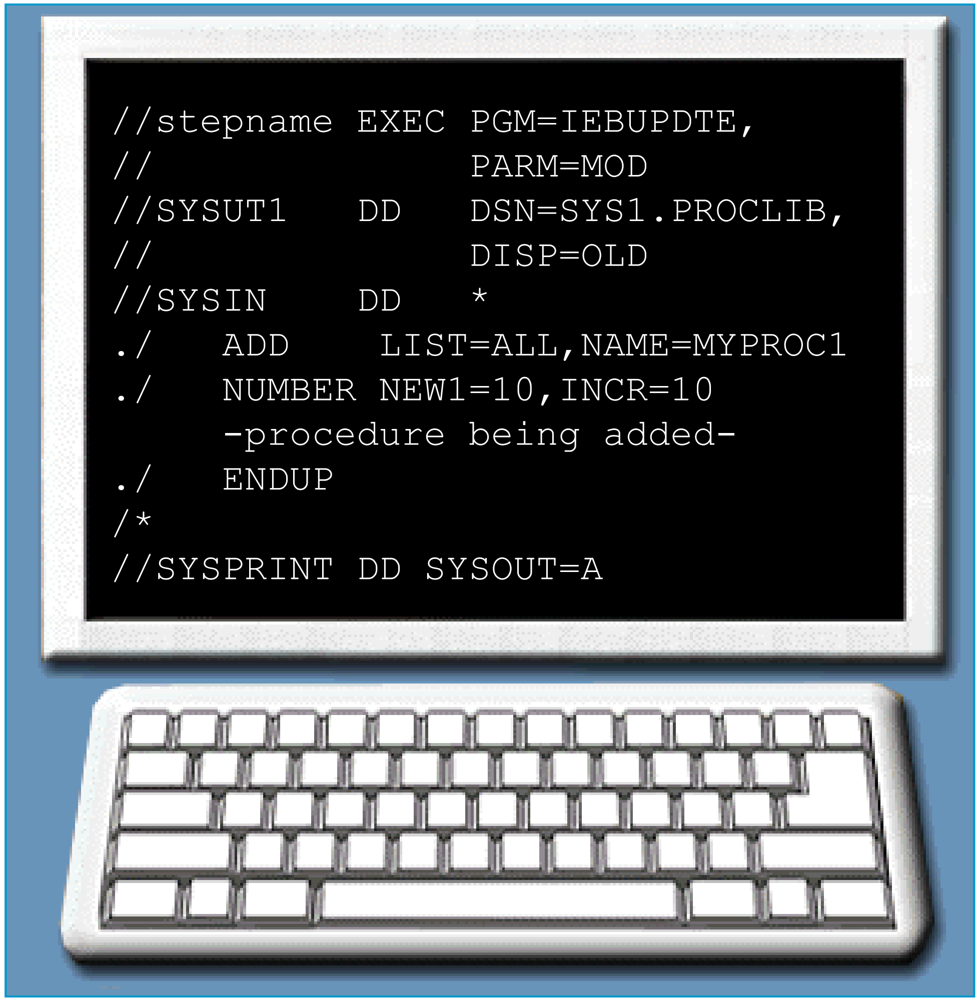
Utility control statements for IEBUPDTE:

```
//SYSIN    DD    *
./    ADD     LIST=ALL,NAME=MYPROC1
./    NUMBER NEW1=10,INCR=10
                 .
                 .
                 .
./    ENDUP
/*
```

Maintaining source libraries.

The example on the right shows the job stream needed for these processing requirements.

Notice that the characters ./ in the control statements begin in column 1.



```
//stepname EXEC PGM=IEBUPDTE,
//              PARM=MOD
//SYSUT1    DD   DSN=SYS1.PROCLIB,
//              DISP=OLD
//SYSIN     DD   *
./    ADD     LIST=ALL,NAME=MYPROC1
./    NUMBER NEW1=10,INCR=10
      -procedure being added-
./    ENDUP
/*
//SYSPRINT DD SYSOUT=A
```

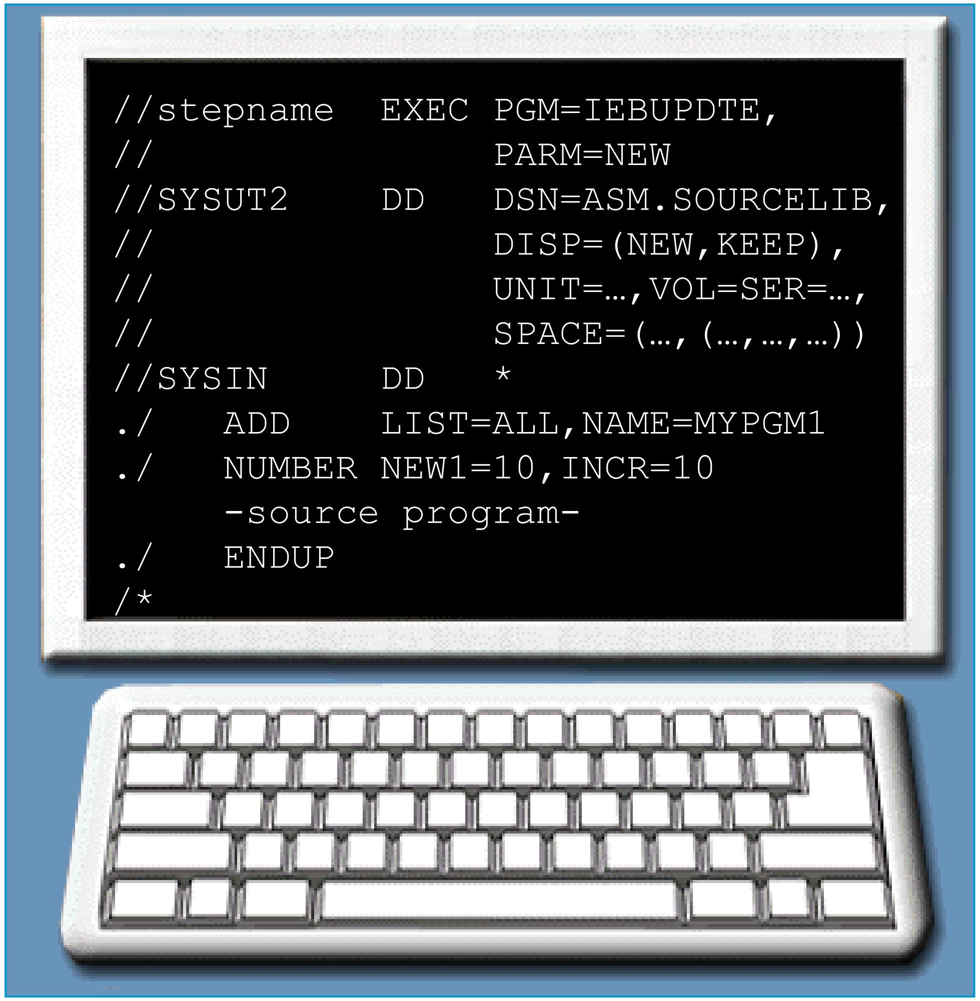
Maintaining source libraries.

In the previous example, a new member was added to an existing PDS.

If you changed the requirements to add a new member to a new PDS, the job stream would look like the screen on the right.

The output data set is ASM.SOURCLIB and the output member is named MYPGM1. In this case:

- PARM=NEW is on the EXEC statement. This specifies a new master data set.
- When creating a PDS, no //SYSUT1 DD is needed.
- The new partitioned master is defined on the SYSUT2 statement.



```
//stepname   EXEC PGM=IEBUPDTE,
//                PARM=NEW
//SYSUT2     DD   DSN=ASM.SOURCELIB,
//                DISP=(NEW,KEEP),
//                UNIT=…,VOL=SER=…,
//                SPACE=(…,(…,…,…))
//SYSIN      DD   *
./    ADD    LIST=ALL,NAME=MYPGM1
./    NUMBER NEW1=10,INCR=10
      -source program-
./    ENDUP
/*
```

Maintaining source libraries.

IEBUPDTE can be used to add (./ ADD) modify (./ CHANGE), or replace (./ REPL) members in a PDS, depending on the control statements used.

./ ADD precedes and names a member or data set to be added. The general form is as follows:

```
./   ADD          NAME=member, LIST=ALL
```

./ CHANGE is used when deleting, numbering or adding data in a member or data set. It is followed by DELETE, NUMBER or data statements.

./ REPL precedes a member to replace an existing member in the SYSUT2 data set.

Are we on track?

Match the IEBUPDTE utility control statement with its function:

./ ADD	**A. Precedes and names a member or a data set to be added to the SYSUT2 data set.**
./ REPL	**B. Precedes a member to replace an existing member in the SYSUT2 PDS.**
./ NUMBER	**C. Indicates modifications to a member of a PDS.**
./ CHANGE	**D. Indicates that portions of a member are to be numbered.**

Maintaining source libraries.

Are we on track?

Assume you have the following records in a member named PAYROLL in a PDS named MYSOURCE:

```
123      J.SMITH                 DEPT.49       00000010
146      J.DOE                   DEPT.98       00000020
987      H.BROWN                 DEPT.012      00000030
993      J.HORNER                DEPT.23       00000040
```

Complete the data statement so that the IEBUPDTE will alter PAYROLL to include the following. Assume there are no spaces between the fields.

```
987      H.BROWN                 DEPT.911      00000030
```

```
//              EXEC   PGM=IEBUPDTE,PARM=MOD  . . .
//SYSIN         DD     *
./   CHANGE     NAME=PAYROLL,LIST=ALL  987 _______________
./   ENDUP
/*
```

Listing PDS directories.

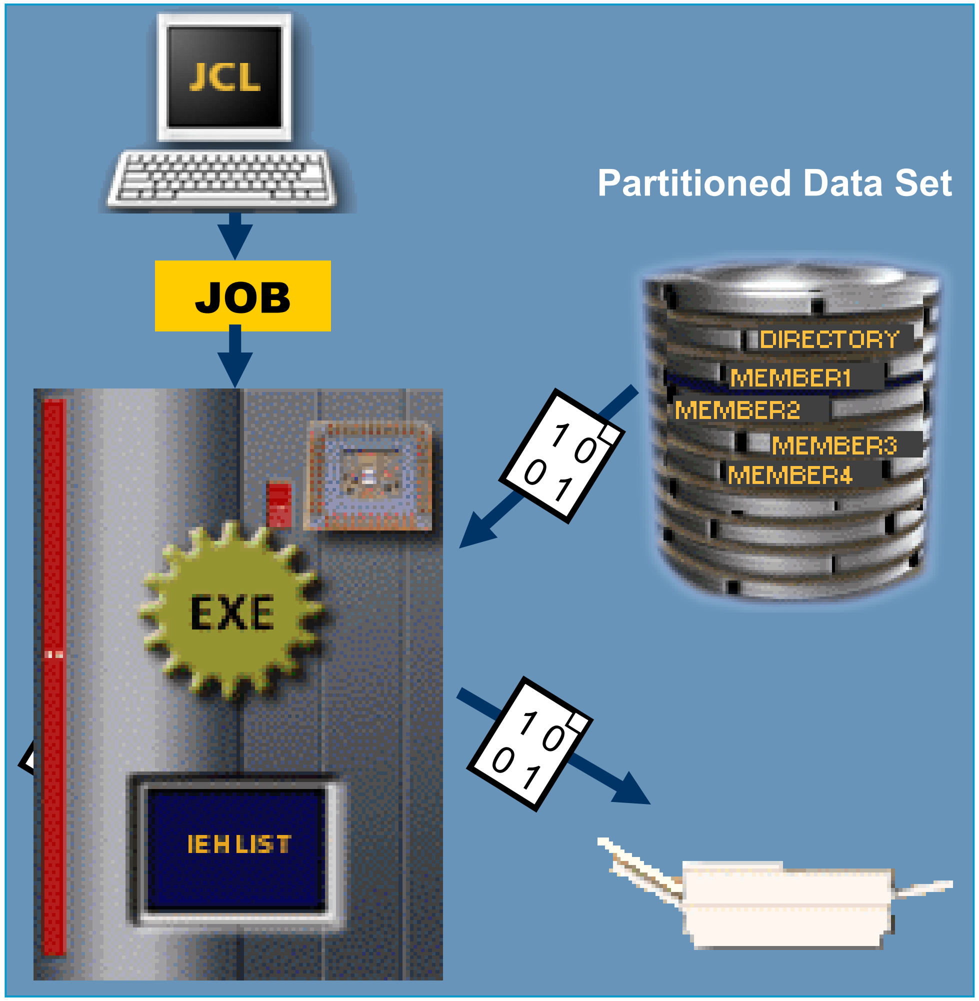
The final utility application concerns printing the contents of a PDS directory.

Why is IEHLIST utility important?

The listing helps determine which member names are already in the PDS.

The IEHLIST (List System Data) utility can:

- List the entries in a PDS directory.
- List the Volume Table of Contents (VTOC) of direct-access volumes.

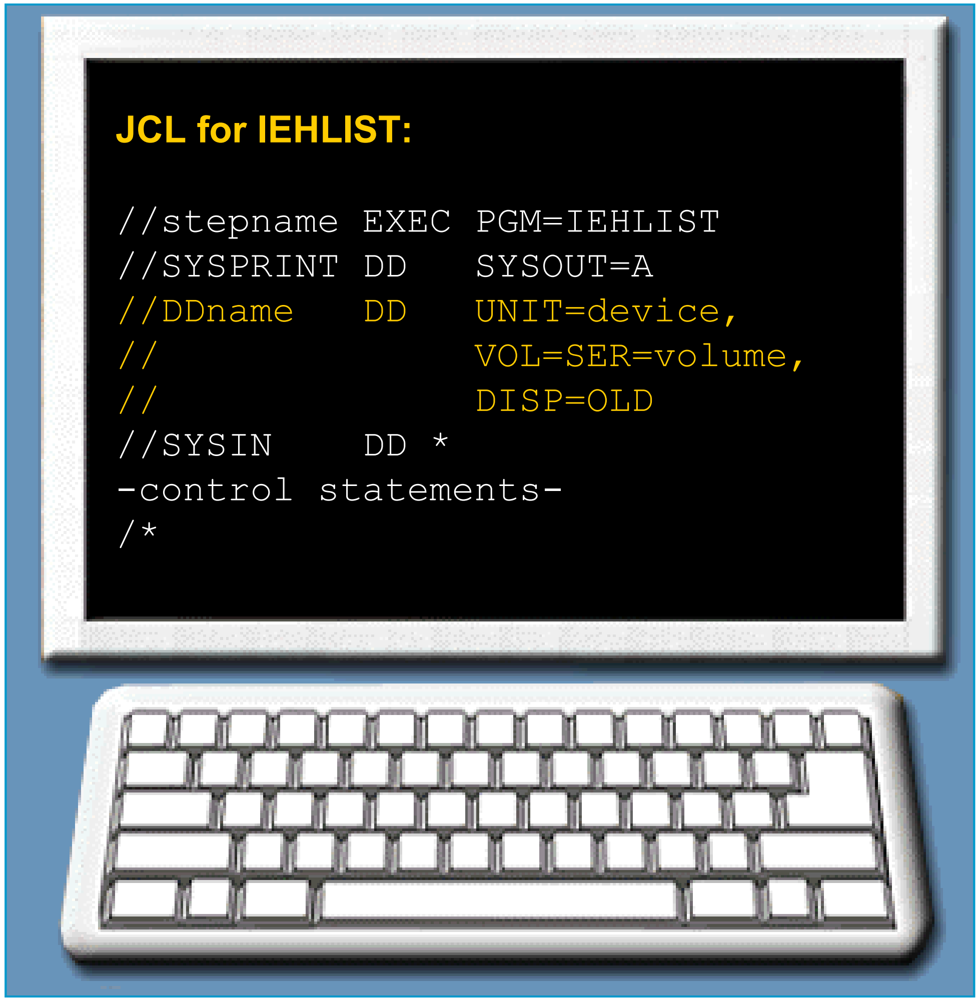


Listing PDS directories.

When using IEHLIST, use the SYSIN and SYSPRINT DD statements.

However, the user can create a DDNAME for the DD statement that defines the volume to be processed.

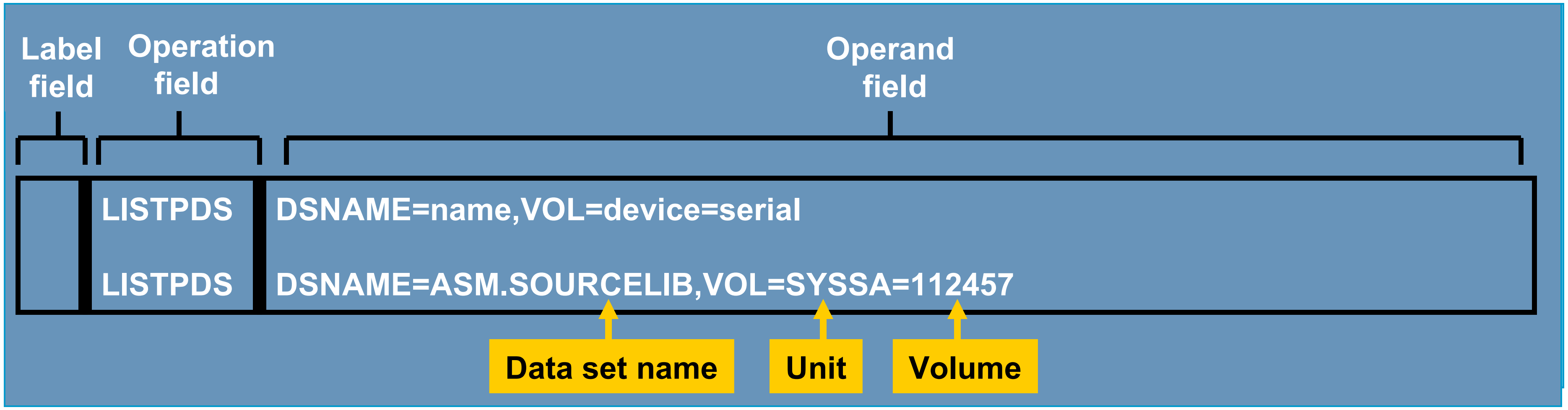
The purpose of this DD statement is to provide enough information for the system to find and allocate the volume.



JCL for IEHLIST:

```
//stepname EXEC PGM=IEHLIST
//SYSPRINT DD   SYSOUT=A
//DDname   DD   UNIT=device,
//              VOL=SER=volume,
//              DISP=OLD
//SYSIN    DD *
-control statements-
/*
```

Listing PDS directories.



```
LISTPDS  DSNAME=name,VOL=device=serial
LISTPDS  DSNAME=ASM.SOURCELIB,VOL=SYSSA=112457
```

For IEHLIST, use a utility control statement to specify the data set's name, rather than coding it on the DD statement. The control statement also indicates the unit type and volume serial number of the data set.

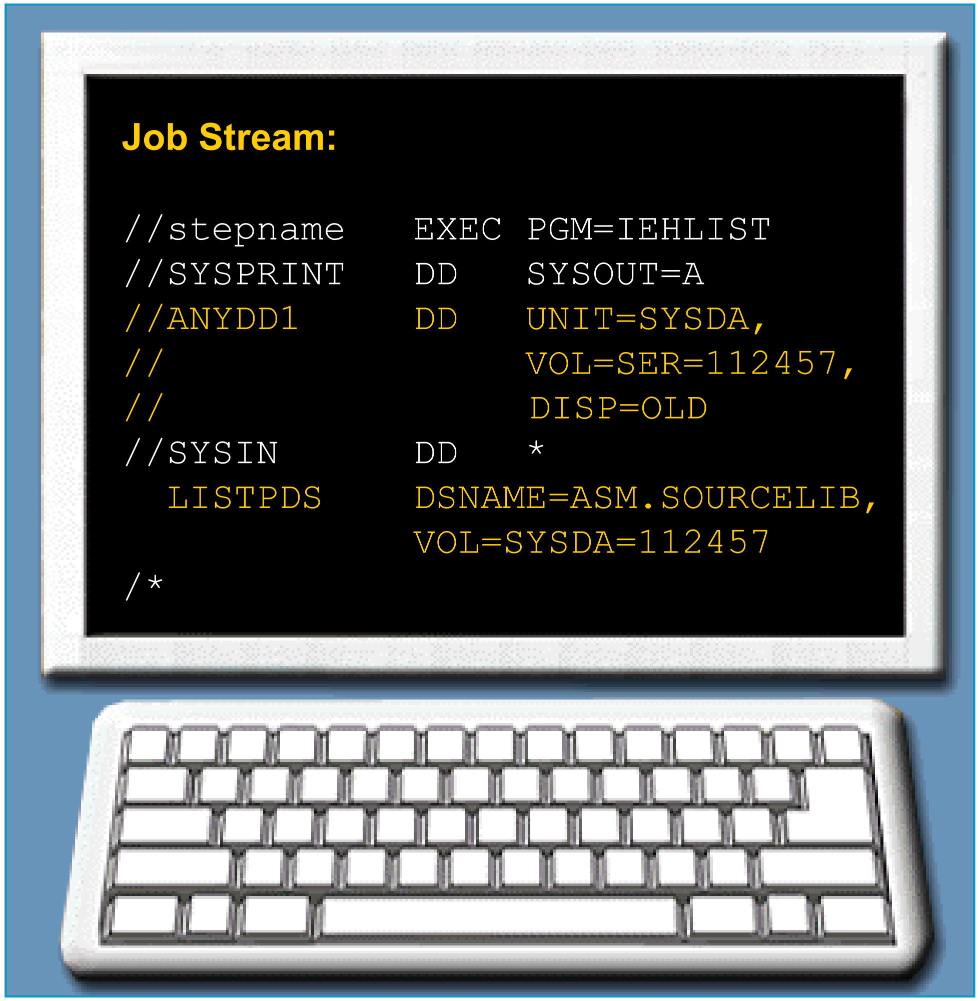
LISTPDS requests a listing of one or more partitioned data sets or PDSEs.

The general form and example are shown above. The user must know the data set's complete name and volume location. The utility cannot refer to JCL or catalog information.

Listing PDS directories – an example.

The complete job stream to list the PDS directory is illustrated on the right.

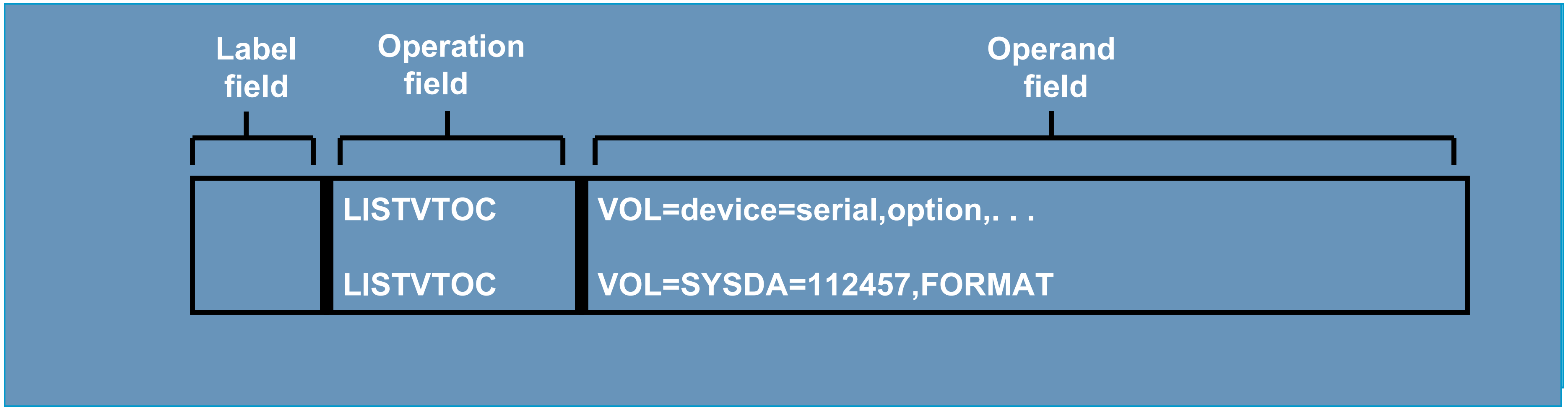
This example uses the ddname ANYDD1. The ANYDD1 DD identifies the directory listing function to be performed, and the name of the data set whose directory needs to be listed.



Job Stream:

```
//stepname    EXEC PGM=IEHLIST
//SYSPRINT    DD   SYSOUT=A
//ANYDD1      DD   UNIT=SYSDA,
//                 VOL=SER=112457,
//                 DISP=OLD
//SYSIN       DD   *
  LISTPDS     DSNAME=ASM.SOURCELIB,
              VOL=SYSDA=112457
/*
```

Listing Volume Table Of Contents – an example.



One other control statement is often used with the IEHLIST utility. LISTVTOC requests a listing of all or part of a volume table of contents.

The example above would list the volume table of contents of DASD volume 112457, using a unit designation of SYSDA. The FORMAT option lists the VTOC in edited form.

Listing PDS directories.

Are we on track?

With the IEHLIST utility, the data set name, volume serial number, and unit type are given on the ___________ control statement.

Listing PDS directories.

Are we on track?

Assume that you want to print the VTOC on direct access volume 987654. You will use a unit designation of DASD.

Code the JCL DD statement required to identify the volume. Use DISP=OLD.

```
//STEPNAME EXEC PGM=IEHLIST
//ANYDD     DD   VOL=SER=987654
//               _______________
//SYSPRINT DD   SYSOUT=A
//SYSIN     DD   *
```

Code the utility control statement required with IEHLIST to request a listing of a VTOC. Specify that the VTOC is to be listed in edited form.

```
  LISTVTOC      _________________
/*
```

Listing PDS directories.

Glossary.

LISTPDS
An IEHLIST control statement that lists the directory of a partitioned data set.

IEBCOPY Library copy program.

IEBCOPY is a data set utility that is used to copy or merge members between one or more PDSs, or PDSEs, in full or in part. You can also use IEBCOPY to create a backup of a PDS into a sequential data set (called an unload data set or PDSU), and to copy members from the backup into a PDS.

You can use IEBCOPY to perform the following tasks:

- Make a copy of a PDS or PDSE.
- Merge PDSs.
- Create a sequential form of a PDS or PDSE for a backup or transport.
- Reload one or more members from a PDSU into a PDS or PDSE.
- Replace members of a PDS or PDSE.
- Rename selected members of a PDS or PDSE when copied.
- Exclude members from a data set to be copied, unloaded, or loaded.
- Compress a PDS in place.
- Upgrade a load module for faster loading by MVS program fetch.
- Copy and reblock load modules.
- Convert a PDS to a PDSE or a PDSE to a PDS.
- and many others...

IEBCOPY Library copy program.

Converting PDSs to PDSEs.

You can use IEBCOPY to convert partitioned data sets to PDSEs.

To convert a PDS to a PDSE, create a PDSE and copy the PDS into the new PDSE. This can be accomplished with one use of IEBCOPY.

IEBCOPY Library copy program.

Copying data sets.

IEBCOPY can be used to totally or partially copy a PDS from one DASD to another. In addition, a data set can be copied to its own volume, provided its data set name is changed. (If the data set name is not changed, IEBCOPY interprets the request as a compress-in-place.)

Members copied into a PDS are not physically reordered; members are copied in the physical order in which they occur in the original data set.

IEBCOPY Library copy program.

Merging data sets.

Merging data sets is done by copying or loading the additional members to an existing PDS. The merge operation (ordering of the directory of the output data set) is automatically performed by IEBCOPY.

Increasing Directory Space for a PDS
IEBCOPY cannot increase the number of directory blocks in a PDS. (A PDSE directory automatically expands as needed.) If you are not sure there will be enough directory blocks in the output PDS you are merging to, then you should expand the output data set directory space before beginning the merge operation.

IEBCOPY Library copy program.

Unloading (backing up) data sets.

IEBCOPY can be used to create a backup copy of a PDS by copying (unloading) it to a sequential data set on DASD, tape, or other device supported by QSAM.

IEBCOPY creates an unload data set when you specify physical sequential organization (DSORG=PS) for the output data set. To create a PDS, specify DSORG=PO and DSNTYPE=PDS or DSNTYPE=LIBRARY.

To unload more than one PDS to the same tape volume in one execution of IEBCOPY, multiple copy operations must be used and multiple sequential data sets must be allocated to successive files on the tape.

The PDSE directory can contain attributes in addition to those traditionally kept in a PDS directory entry.
Some PDSE extended attributes are recorded on an unload data set and will be reloaded when the target is a PDSE.

IEBCOPY Library copy program.

Job control statements.

Statement	Use
JOB	Starts the job.
EXEC	Starts IEBCOPY.
SYSPRINT DD	Defines a sequential data set used for listing control statements and messages.
SYSUT1 or anyname1 DD	Defines a PDS or unload data set for input.
SYSUT2 or anyname2 DD	Defines a PDS or unload data set for output.
SYSUT3 DD	Defines a spill data set on a DASD or VIO device. SYSUT3 is used when there is no space in virtual storage for some input data set directory entries.
SYSUT4 DD	Defines a spill data set on a DASD or VIO device. SYSUT4 is used when there is no space in virtual storage for the output data set directory.
SYSIN DD	Defines the optional control data set.

IEBCOPY Library copy program.

Utility control statements.

Statement	Use
ALTERMOD	Indicates the beginning of an alter-in-place operation for load modules.
COPY	Indicates the beginning of a COPY operation.
COPYGRP	Indicates the beginning of a COPYGRP operation.
COPYMOD	Indicates the beginning of a copy and load module reblock operation.
INDD=	Indicates the beginning of another copy step.
EXCLUDE	Specifies members in the input data set to be excluded from the copy step.
SELECT	Specifies which members in the input data set are to be copied.

IEBCOPY Library copy program.

Example 1 – Copy an entire data set.

In this example, a PDS (DATASET5) is copied from one disk volume to another.

```
//COPY       JOB ...
//JOBSTEP    EXEC PGM=IEBCOPY
//SYSPRINT   DD SYSOUT=A
//SYSUT1     DD DSNAME=DATASET5,UNIT=disk,VOL=SER=111113,DISP=SHR
//SYSUT2     DD DSNAME=DATASET4,UNIT=disk,VOL=SER=111112,
//              DISP=(NEW,KEEP),SPACE=(TRK,(5,1,2))
```

The control statements are discussed below:

• SYSUT1 DD defines a PDS, DATASET5, that contains two members (A and C).

• SYSUT2 DD defines a new PDS, DATASET4, that is to be kept after the copy operation. Five tracks are allocated for the data set; two blocks are allocated for directory entries.

• Because the PDS has only two members, SYSUT3 and SYSUT4 DD are not needed.

IEBCOPY Library copy program.

Example 2 – Merge four data sets.

In this example, members are copied from three input PDSs (DATASET1, DATASET5, and DATASET6) to an existing output PDS (DATASET2). The sequence in which the control statements occur controls the manner and sequence in which PDSs are processed.

```
//COPY       JOB ...
//JOBSTEP    EXEC PGM=IEBCOPY
//SYSPRINT   DD SYSOUT=A
//IN1        DD DSN=DATASET1,UNIT=disk,VOL=SER=111112,DISP=SHR
//IN5        DD DSN=DATASET5,UNIT=disk,VOL=SER=111114,DISP=OLD
//IN6        DD DSN=DATASET6,UNIT=disk,VOL=SER=111117,DISP=(OLD,DELETE)
//OUT2       DD DSN=DATASET2,UNIT=disk,VOL=SER=111115,DISP=(OLD,KEEP)
//SYSUT3     DD UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN      DD *
  COPYOPER   COPY INDD=IN1
                  INDD=IN5
                  INDD=IN6
                  OUTDD=OUT2
/*
```

IEBCOPY Library copy program.

Example 2 – Merge four data sets.

The control statements are discussed below:

- IN1 DD defines a PDS (DATASET1). This data set contains three members (A, B, and F).
- IN5 DD defines a PDS (DATASET5). This data set contains two members (A and C).
- OUT2 DD defines a PDS (DATASET2). This data set contains two members (C and E).
- IN6 DD defines a PDS (DATASET6). This data set contains three members (B, C, and D). This data set is to be deleted when processing is completed.
- SYSUT3 defines a temporary spill data set.
- SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement and three INDD statements.
- COPY indicates the start of the copy operation. The OUTDD parameter specifies DATASET2 as the output data set.
- The first INDD statement specifies DATASET1 as the first input data set to be processed. All members (A, B and F) are copied to DATASET2.
- The second INDD statement specifies DATASET6 as the second input data set to be processed.

IEBCOPY Library copy program.

Example 3 – Unload and compress a data set.

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

```
//SAVE      JOB ...
//STEP1     EXEC PGM=IEBCOPY
//SYSPRINT  DD SYSOUT=A
//SYSUT1    DD DSN=PARTPDS,UNIT=disk,VOL=SER=PCP001,DISP=OLD
//SYSUT2    DD DSN=SAVDATA,UNIT=tape,VOL=SER=TAPE03,DISP=(NEW,KEEP),
//             LABEL=(,SL)
//SYSUT3    DD DSN=TEMP1,UNIT=disk,VOL=SER=111111,DISP=(NEW,DELETE),
//             SPACE=(80,(60,45))
//SYSIN     DD DUMMY
//STEP2     EXEC PGM=IEBCOPY,COND=(0,NE),PARM='SIZE=500K'
//SYSPRINT  DD SYSOUT=A
//COMPDS    DD DSN=PARTPDS,UNIT=disk,DISP=OLD,VOL=SER=PCP001
//SYSUT3    DD DSN=TEMPA,UNIT=disk,VOL=SER=111111,DISP=(NEW,DELETE),
//             SPACE=(80,(60,45))
//SYSIN     DD *
  COPY OUTDD=COMPDS,INDD=COMPDS
/*
```

IEBCOPY Library copy program.

Example 3 – Unload and compress a data set.

The control statements are discussed below:

• SYSUT1 DD defines a PDS (PARTPDS) that resides on a disk volume and is assumed to have 700 members.The number of members is used to calculate the space allocation on SYSUT3.
• SYSUT2 DD defines a sequential data set to hold PARTPDS in unloaded form. This data set must be NEW.
• SYSUT3 DD defines the temporary spill data set.
• SYSIN DD defines the control data set. Because SYSIN is dummied and SYSUT2 defines a sequential data set, all members of the SYSUT1 data set will be unloaded to the SYSUT2 data set.
• The second EXEC statement marks the beginning of the compress-in-place operation.
• COMPDS DD defines a PDS (PARTPDS).
• SYSUT3 DD defines the temporary spill data set to be used if there is not enough space in main storage for the input data set's directory entries.
• SYSIN DD defines the control data set, which follows in the input stream.
• COPY marks the beginning of the copy operation. Because the same DD statement is specified for both the INDD and OUTDD operands, the data set is compressed in place.

Example 4 – Convert a PDS to a PDSE.

In this example, a PDS is converted to a PDSE.

```
//CONVERT   JOB ...
//STEP1     EXEC PGM=IEBCOPY
//SYSPRINT  DD SYSOUT=A
//SYSUT1    DD DSN=PDSSET,DISP=SHR,DSNTYPE=PDS
//SYSUT2    DD DSN=PDSESET,LIKE=PDSSET,DSNTYPE=LIBRARY,
//             DISP=(NEW,CATLG),STORCLAS=SCLASX,DATACLAS=DCLASY
```

The control statements are discussed below:

- SYSUT1 DD defines the input PDS. The DSNTYPE keyword has no effect because it is an existing data set.
- SYSUT2 DD defines the output PDSE. This new data set will be SMS-managed because it has a storage class. The LIKE parameter indicates that the DCB and SPACE attributes for PDSESET are to be copied from PDSSET. The DSNTYPE parameter defines the new data set as a PDSE rather than as a PDS. DATACLAS=DCLASY identifies the PPDSE as a program object PDSE with undefined logical record length. The SMS chooses an appropriate volume for the allocation, based on how SCLASX was defined.

DFDSS utility.

This program is the primary disk dump and restore program provided with z/OS. It is capable of filtering and selecting which data sets to dump or restore, but it is intended more to operate on all data sets in a volume or all data sets having some SMS management class rather than on individual data sets.

The purpose of dumping a disk is usually to provide a backup of the contents that can be restored, if needed. A common use is to dump complete volumes but restore only a specific data set that was accidentally destroyed.

A backup is usually written to tape, but can be written to a disk data set. A disk can be dumped track-by-track (known as a physical dump) or data set-by-data set (known as a logical dump). When a logical dump is performed, multiple data set extents may be combined into a single extent, PDSs are compressed, and free space is all in a single extent.

DFDSS utility.

You can use DFDSS interactively from a terminal similar to the way ISPF is used (using Interactive Storage Management Facility – ISMF), or you can invoke it through JCL.

DFDSS provides several facilities not in the other utility programs, such as releasing unused space in data sets, combining data set extents , consolidating free space on volumes, and backing up and restoring data sets.

DFDSS can convert data sets to be SMS-managed data sets. It is intended more for site management than for individual application programmer.

Data facility data set services - DFDSS.

BUILDSA command.

The BUILDSA function builds the IPL-able core image under the current operating system.

Use the BUILDSA command to build the IPL-able core image for the Stand-Alone Services program. You can specify the device (card reader, tape drive, or DASD volume) from which Stand-Alone Services will be IPLed.

Data facility data set services - DFDSS.

BUILDSA command – an example.

Core Image for IPL from Tape

In this example, Stand-Alone Services is created for IPLing in stand-alone mode from a tape. The core image is then placed on an unlabeled tape.

```
//BUILDSA   JOB accounting information,REGION=nnnnK
//STEP1     EXEC PGM=ADRDSSU,PARM='UTILMSG=YES'
//SAMODS    DD DSN=SYS1.SADRYLIB,DISP=SHR
//TAPEDD    DD DSN=ADRSA.IPLT,UNIT=3480,LABEL=(,NL),
//             DISP=(NEW,KEEP),VOL=SER=TAPE01,
//             DCB=(DSORG=PS,RECFM=U,BLKSIZE=32760,LRECL=32760)
//SYSPRINT  DD SYSOUT=A
//SYSIN     DD *
  BUILDSA -
     INDD(SAMODS) -
     OUTDD(TAPEDD) -
     IPL(TAPE)
/*
```

Data facility data set services - DFDSS.

COMPRESS command.

The COMPRESS command compresses PDSs on a specified volume. Compressing (degassing) removes unused space between members in a PDS. This command is useful for compressing system PDSs before you apply maintenance (to avoid certain space-related abends).

Restriction: You must not compress data sets that contain DFSMSdss or IEBCOPY executable code.

COMPRESS command – an example.

Example of Compress Operations

The following example compresses a selected PDS:

```
//JOB1       JOB  accounting information,REGION=nnnnK
//STEP1      EXEC PGM=ADRDSSU
//SYSPRINT   DD SYSOUT=A
//SYSIN      DD *
  COMPRESS -
     DYNAM(338000)         /* DYNAM ALLOC VOL 338000 */ -
     EXCLUDE(SYS1.**)      /* EXCL 'SYS1....' DATA SETS */ -
                           /* IF THEY MEET THIS CRITERION */ -
     BY((DSCHA EQ 0))      /* DATA SET WAS BACKED UP */
/*
```

Compress PDSs on volume 338000 if:

- **They are not system data sets (EXCLUDE(SYS1.**)), and**
- **They have not been updated (DSCHA EQ 0) since the last time they were backed up (dumped).**

CONVERTV command.

The CONVERTV command is used to convert existing volumes to and from SMS management without data movement. The CONVERTV command performs three functions:

- **Locks volumes that are ready for conversion to prevent new data set allocations (PREPARE keyword).**

- **Examines volumes identified by SMS to determine if they can be converted to SMS management (TEST keyword). No conversion is actually performed, but DFSMSdss identifies any data sets that cannot be converted to SMS management.**

- **Performs conversion of volumes into or out of SMS management.**

CONVERTV command – an example.

Using the CONVERTV Command to Simulate Conversion

```
//JOB1       JOB  accounting information,REGION=nnnnK
//STEP1      EXEC PGM=ADRDSSU
//SYSPRINT   DD SYSOUT=A
//SYSIN      DD *
  CONVERTV SMS -
     DYNAM((VOL001,3380),(VOL002,3380),(VOL003)) -
     TEST
```

The preceding example uses the TEST keyword to simulate conversion. The TEST keyword produces a report that indicates whether the three volumes (VOL001, VOL002, and VOL003) can be converted to SMS management.

COPY command.

The DFSMSdss COPY command performs data set movement, volume movement, and track movement from one DASD volume to another. You can copy data sets to another volume of either like or unlike device types. Like devices have the same track capacity (3390 Model 2 and 3390 Model 3), while unlike devices have different track capacities (3380 Model K and 3390 Model 3).

DFSMSdss offers two ways to process COPY commands as follows:

• *Logical processing* is data set-oriented, which means that it operates against data sets and volumes independently of physical device format.

• *Physical processing* operates against volumes and tracks, but moves data at the track-image level.

Data facility data set services - DFDSS.

COPY command – an example.

A Tracks Copy with Track Relocation

The following example shows a tracks copy operation in which the contents of cylinder 1, tracks 0 through 14, on source volume 338000 are copied to cylinder 3, tracks 0 through 14, on target volume 338001. The operation stops if a permanent error occurs on the source volume (CANCELERROR). The data written to the target volume is to be verified (WRITECHECK).

```
//JOB2        JOB  accounting information,REGION=nnnnK
//STEP1       EXEC PGM=ADRDSSU
//SYSPRINT    DD SYSOUT=A
//SYSIN       DD *
  COPY TRACKS(1,0,1,14)      /* SOURCE TRACKS */ -
     OUTTRACKS(3,0)          /* TARGET TRACKS */ -
     INDYNAM(338000)         /* ALLOC VOL 338000 DYNAMICALLY */ -
     OUTDYNAM(338001)        /* ALLOC VOL 338001 DYNAMICALLY */ -
     CANCELERROR             /* STOP ON INPUT ERROR */ -
     WRITECHECK              /* VERIFY DATA WRITTEN TO OUT VOL */
/*
```

Data facility data set services - DFDSS.

DEFRAG command.

When you issue the DEFRAG command, DFSMSdss relocates data set extents on a DASD volume to reduce or eliminate free space fragmentation. A summary report is printed that lists the before and after statistics of the volume.

Attention: Canceling the DEFRAG command is strongly discouraged. Canceling an in-process DEFRAG can damage data in numerous and unpredictable ways.

Data facility data set services - DFDSS.

DEFRAG command – an example.

A DEFRAG Operation with Excluded Data Sets

```
//JOB1       JOB  accounting information,REGION=nnnnK
//STEP1      EXEC PGM=ADRDSSU
//SYSPRINT   DD SYSOUT=A
//DASD       DD UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//A1         DD DSN=USER2.EXCLUDE,DISP=SHR
//SYSIN      DD *
  DEFRAG DDNAME(DASD) -
     EXCLUDE(LIST(USER2.**.LIST,*.LOAD))
/*
```

In the example, DASD volume 111111 is defragmented. All data sets whose first and last qualifiers are USER2 and LIST, respectively, are to be excluded from this operation, as are data sets with two qualifiers whose second qualifier is LOAD.

Data facility data set services - DFDSS.

DUMP command.

With the DUMP command, you can dump DASD data to a sequential data set. The storage medium for the sequential data set can be a tape or DASD. You can dump data sets, an entire volume, or ranges of tracks.

DFSMSdss offers two ways to process DUMP commands:

- ***Logical processing* is data set-oriented, which means it operates against data sets independently of physical device format.**

- ***Physical processing* can operate against data sets, volumes, and tracks, but is oriented toward moving data at the track-image level.**

Data facility data set services - DFDSS.

DUMP command – an example.

Dumping a User Catalog and its Aliases

To dump a user catalog, you perform a logical data set dump with the fully qualified user catalog name as the data set name. If the user catalog has any aliases, the aliases are automatically dumped.

```
//JOB2        JOB  accounting information,REGION=nnnnK
//STEP1       EXEC PGM=ADRDSSU
//SYSPRINT    DD SYSOUT=A
//DASD1       DD UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//TAPE        DD UNIT=3480,VOL=SER=TAPE02,
//               LABEL=(1,SL),DISP=(NEW,CATLG),DSNAME=USER2.BACKUP
//SYSIN       DD *
  DUMP    OUTDDNAME(TAPE) -
  DATASET(INCLUDE(MY.USER.CAT))
/*
```

PRINT command.

With the PRINT command, you can print:

- **A single-volume non-VSAM data set, as specified by a fully qualified name.**

- **A single-volume VSAM data set component (not cluster). The component name specified must be the name in the VTOC, not the name in the catalog.**

- **Ranges of tracks.**

- **All or part of the VTOC. The VTOC location need not be known.**

Data facility data set services - DFDSS.

PRINT command – an example.

Printing a Component of a VSAM Cluster

```
//JOB3       JOB  accounting information,REGION=nnnnK
//STEP1      EXEC PGM=ADRDSSU
//SYSPRINT   DD SYSOUT=A
//SYSIN      DD *
  PRINT INDYNAM(338000)                /* ALLOC VOL 338000 DYNAMICALLY */ -
     DATASET(PARTS.VSAM1.INDEX)        /* DATA SET THAT HAS BAD TRACK */ -
     WAIT(0,0)                         /* DO NOT WAIT IF ENQ FAILS */ -
     TOL(ENQF)                         /* IGNORE ENQ FAILURES */ -
     PSWD(PARTS.VSAM1/USERPSWD)        /* PASSWORD FOR CLUSTER */
/*
```

RELEASE command.

The RELEASE command releases allocated but unused space from all eligible sequential, partitioned, and extended-format VSAM data sets. DFSMSdss offers two ways to process RELEASE commands:

- **Logical processing operates on a single selected data set at a time.**

- **Physical processing operates on all selected data sets that reside on a single volume.**

RELEASE command – an example.

Example of a Release Operation

The following is an example of a release operation on selected sequential and PDSs:

```
//JOB1       JOB  accounting information,REGION=nnnnK
//STEP1      EXEC PGM=ADRDSSU
//SYSPRINT   DD SYSOUT=A
//SYSIN      DD *
  RELEASE INCLUDE(**) -
     DYNAM(338000)       /* DYNAM ALLOC VOL 338000 */ -
     MINTRKS(10)         /* THERE ARE 10 OR MORE UNUSED TRKS */ -
                         /* MINSEC NOT SPEC. IT DEFAULTS TO 1 */
/*
```

Unused tracks of sequential and PDSs on volume 338000 are to be released if both:

- **The number of unused tracks in the data set is greater than or equal to 10.**
- **The data set can be extended later if required (MINSEC(1)). This need not be specified, because it is the default.**

RESTORE command.

With the RESTORE command, you can restore data to DASD volumes from DFSMSdss-produced dump volumes. You can restore data sets, an entire volume, or ranges of tracks. You can restore to unlike devices from a logical dump tape.

DFSMSdss offers two ways to process RESTORE commands:

- ***Logical processing*** **is data set-oriented, which means it operates against data sets independently of physical device format.**

- ***Physical processing*** **can operate against data sets, volumes, and tracks, but is oriented toward moving data at the track-image level.**

RESTORE command – an example.

Tracks Restore Operation

Example shows that DASD volume numbered 111111 will be restored from the first data set of standard label tape volumes called TAPE01 and TAPE02. The command input for a tracks restore operation are shown below.

```
//JOB1      JOB  accounting information,REGION=nnnnK
//STEP1     EXEC PGM=ADRDSSU
//SYSPRINT  DD SYSOUT=A
//TAPE      DD UNIT=3480,VOL=SER=(TAPE01,TAPE02),
//             LABEL=(1,SL),DISP=(OLD,KEEP),DSNAME=USER2.BACKUP
//DASD      DD UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//SYSIN     DD *
  RESTORE TRACKS(1,0,1,5) INDDNAME(TAPE) -
     OUTDDNAME(DASD) PURGE
/*
```

Data Facility Sort (DFSORT).

An IBM licensed program that is a high-speed data-processing utility. DFSORT provides a method for sorting, merging, and copying operations, as well as providing versatile data manipulation at the record, field, and bit level.

Data Facility Sort (DFSORT).

DFSORT sample JCL.

```
//JS10      EXEC PGM=SORT,
//               PARM=sort parms...
```

The EXEC statement invokes DFSORT via SORT or ICEMAN entry point names and passes an optional PARM field value (ABEND, AVRGLEN, DYNALLOC, FILSZ, LIST, MSGPRT, SIZE, SKIPREC, VERIFY, and others).

```
//STEPLIB   DD DSN=...,DISP=SHR
```

This DD is only needed if DFSORT is not on the linklist, or if you are using a MODS statement that names STEPLIB as the library from which the sort exits are to be fetched.

```
//SORTLIB   DD DSN=...,DISP=SHR
```

This DD is needed if you are using tape drives as sort work units, or if a merge operation couldn't use the BLOCKSET merge method.

```
//DFSPARM   DD *
  sort parms and sort control statements described below...
```

This DD is optional if you want to provide both the EXEC PARM options and the control statements together from one source. This is a help when you are calling DFSORT from another program.

DFSORT sample JCL - continuation.

```
//SYMNAMES  DD DSN=...,DISP=SHR
```
Defines the SYMNAMES data set containing statements to be used for symbol processing. Required only if symbol processing is to be performed.

```
//SYMNOUT   DD DSN=...,DISP=SHR
```
Defines the data set in which SYMNAMES statements and the symbol table are to be listed. Optional if SYMNAMES DD is specified. Otherwise ignored.

```
//SYSOUT    DD SYSOUT=*
```
This DD statement is required because DFSORT messages are written to this file.

```
//SORTIN    DD DSN=input..DSN.to.sort,
//             DISP=SHR
```
This required DD statement points to the input for a sort operation. The input file can be a sequential file or PDS member, or a VSAM data set. Concatenated non-VSAM files are supported if they all have the same RECFM with the largest block size first in the concatenation.

DFSORT sample JCL - continuation.

```
//SORTINnn  DD DSN=input..DSN.to.sort,
//             DISP=SHR
```

Identifies one of the pre-sorted input files for a merge operation. 'nn' is a number in the range 01 through 99. SORTINnn files can't be concatenated. All SORTINnn files have to have the same LRECL if fixed-length records are used.

```
//SORTOUT   DD DSN=sorted.output.dsn,
//             DISP=SHR
```

Identifies the output file for both sort and merge operations. This can be a sequential file, a PDS member, or a VSAM file.

```
//SORTWKnn  DD UNIT=SYSDA,DISP=SHR,
//             SPACE=(TRK,(30,30),RLSE)
```

Defines sort work files, where 'nn' is a suffix in the range 01 through 32. SORTWKnn files aren't needed if sorting a small data set whose records will all fit in storage at once, or if you told DFSORT to obtain its sort work files dynamically via the DYNALLOC parameter on an OPTION or SORT control statement.

DFSORT sample JCL - continuation.

```
//outfil    DD UNIT=SYSDA,DISP=SHR,
//             SPACE=(TRK,(30,30),RLSE)
```

The OUTFIL DD statements describe the characteristics of the data sets in which the processed (sorted, merged, copied) records are to be placed and indicate their location.
Although the ddname SORTOUT can actually be used for an OUTFIL data set, the term "SORTOUT" will be used to denote the single non-OUTFIL output data set. Simply, you can specify different DDname if you do not want to use SORTOUT.

```
//SYSIN     DD *
  sort control statements described below...
```

The control statements for DFSORT are input from this file. If DFSORT is dynamically invoked from a PL/1 or COBOL program, the SORTCNTL DD is used to override the control statements passed by the invoking program.

```
//SORTCNTL  DD DSN=...,DISP=SHR
  sort control statements described below...
```

Defines the data set from which additional or changed DFSORT control statements can be read when DFSORT is program-invoked. It is the same as SYSIN DD but control statement are saved in data set.

Data Facility Sort (DFSORT).

DFSORT sample JCL - continuation.

```
//SORTCKPT  DD DSN=CHECKPT,UNIT=TAPE,
//             DISP=(,KEEP),VOL=SER=ABC123
```

This DD statement allocates a sequential file to be used as the checkpoint data set when sort checkpointing is requested via the CKPT parameter on an OPTION or SORT control statement.

```
//SORTSNAP  DD SYSOUT=*
```

SNAP dump output.

```
//SYSUDUMP  DD SYSOUT=*
```

User dump output.

```
//SYSMDUMP  DD SYSOUT=*
```

Machine dump output.

```
//SYSABEND  DD SYSOUT=*
```

Abend dump output.

DFSORT sample JCL - continuation.

```
//SORTDIAG  DD DUMMY
```

The presence of this DD statement makes DFSORT write all messages (including special diagnostics) to the SYSOUT file. The SORTDIAG DD should be used with CAUTION since its usage substantially degrades DFSORT performance.

DFSORT control statements – SORT.

The SORT control statement is used to initiate a sort and describe which field(s) in each record are to be used as sort fields. The format of the SORT control statement is:

```
SORT FIELDS=(pos,length,format,sequence,...) |
     FIELDS=(pos,length,sequence,...),FORMAT=format |
     FIELDS={(} COPY {)} |
     {,CKPT}
     {,DYNALLOC{=(d | ,n | d,n | OFF)}}
     {,EQUALS | ,NOEQUALS}
     {,FILSZ={n | En | Un} }
     {,SIZE={n | En | Un} }
     {,SKIPREC=n}
     {,STOPAFT=n}
     {,Y2PAST={s | f}
```

DFSORT control statements – RECORD.

The RECORD statement tells DFSORT the record format and lengths of the records in the input file. You only need a RECORD statement if the INPUT is from a VSAM file. The format of the RECORD control statement is:

```
RECORD TYPE={F | D},LENGTH=(len1{,len2,len3}) |
       TYPE=V,LENGTH=(len1{,len2,len3,len4,len5,len6,len7})
```

DFSORT control statements – MERGE.

The MERGE control statement requests a merge operation. The MERGE control statement is similar to the SORT control statement, but a few SORT parameters are not needed. The format of the MERGE statement is:

```
MERGE FIELDS=(pos,length,format,sequence,...) |
      FIELDS=(pos,length,sequence,...),FORMAT=format |
      FIELDS={(} COPY {)}
      {,EQUALS | ,NOEQUALS}
      {,FILES=n}
      {,FILSZ=n | ,SIZE=n}
      {,SKIPREC=n}
      {,STOPAFT=n}
      {,Y2PAST={s | f}
```

DFSORT control statements – END.

The END statement tells DFSORT to stop reading the input control statement data set; it marks a logical 'end-of-file' on the control statement input. You can insert an END statement to make DFSORT ignore control statements after the END statement.

DFSORT control statements – OPTIONS.

The OPTION control statement is used to set or override various DFSORT execution options. Some of the options that you can set on the OPTION control statement can also be set via the JCL PARM= field or on a SORT or MERGE control statement. The format of the OPTION statement is:

```
OPTION {ARESALL={ n | nK | nM} }
       {,ARESINV={n | nK | nM}
       {,AVGRLEN=n}
       ...
       ...
       many others
       ...
       ...
```

DFSORT control statements – INCLUDE.

The INCLUDE control statement is used to establish selection criteria for the records to be included in the output data set. You can include a record by comparing the contents of its fields to a constant or to another field in the record. The format of the INCLUDE statement is:

```
INCLUDE COND=({expression,{{AND | OR}|{ALL | NONE}},expression},...)
{,FORMAT=x}
```

DFSORT control statements – OMIT.

The OMIT control statement is used to establish selection criteria for the records to be omitted from the output data set. You can omit a record by comparing the contents of its fields to a constant or to another field in the record. The format of the OMIT statement is:

```
OMIT COND=({expression,{{AND | OR}|{ALL | NONE}},expression},...)
{,FORMAT=x}
```

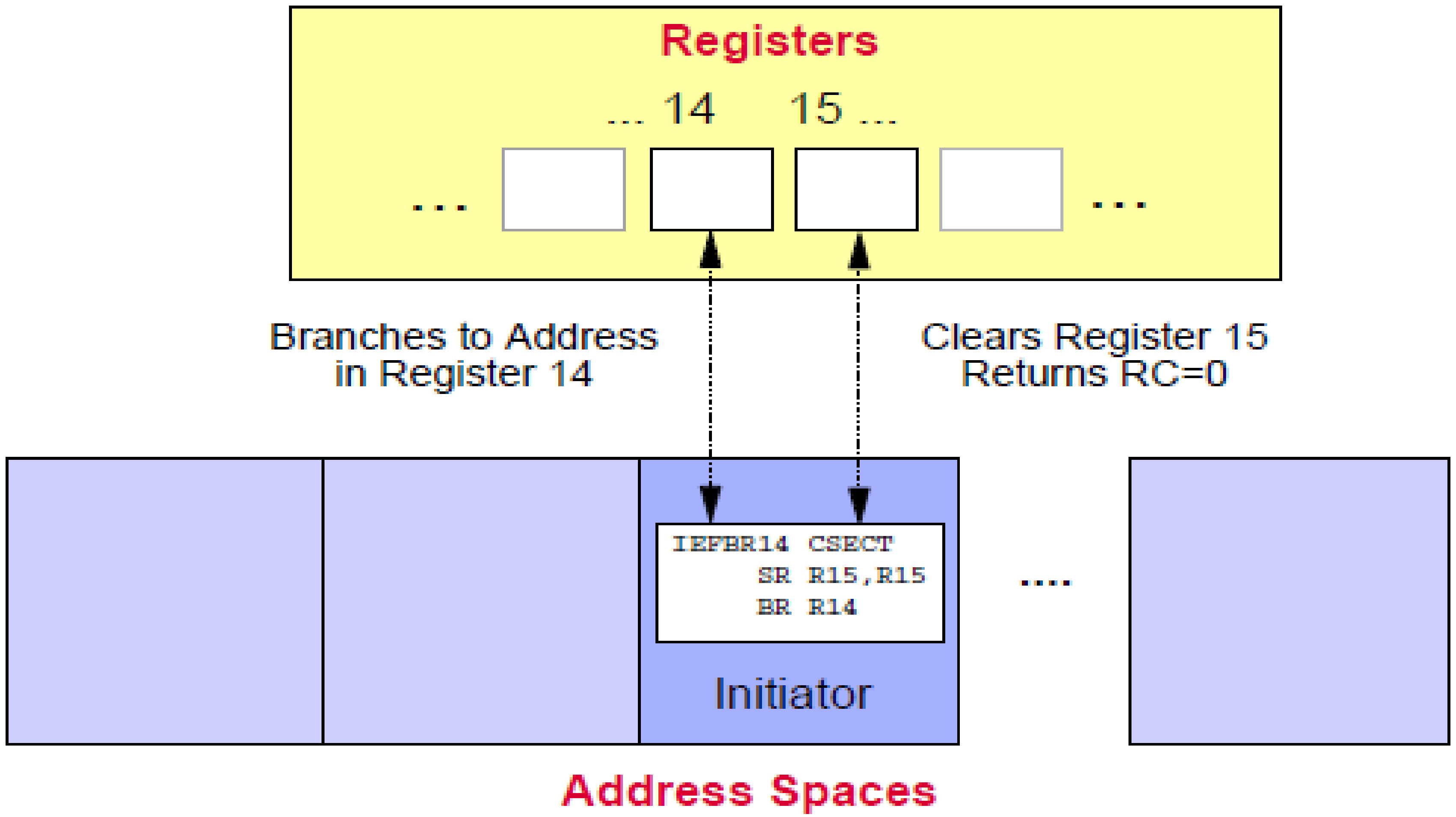
DFSORT control statements – ALTSEQ.

The ALTSEQ statement tells DFSORT to change the collating sequence for some specified character(s). The ALTSEQ process causes degraded performance. The format of the ALTSEQ statement:

```
ALTSEQ CODE=(ccpp{,ccpp...})
```

The shortest program in the World.

One more special utility – IEFBR14.


Registers
... 14
15 ...
...
...
Branches to Address in Register 14
Clears Register 15 Returns RC=0
Initiator
....
Address Spaces

```
IEFBR14  CSECT
         SR  R15,R15
         BR  R14
```

Unit summary.

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

- Determine the job stream.
- Print sequential data sets.
- Edit sequential data sets.
- Convert input into a PDS.
- Maintain source libraries.
- List PDS directories.
- Copy PDS or PDSE members.
- Sort data.

JCL

Chapter c4
Sample utility application

Job Control Language

Chapter c4

Sample utility application

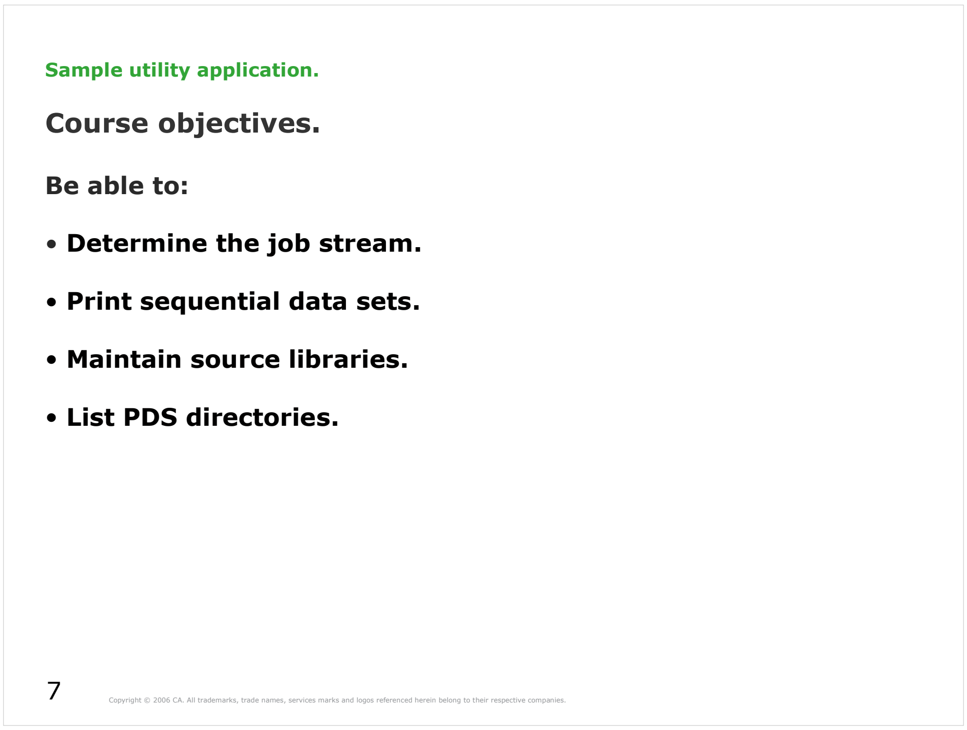
Sample utility application.

Unit introduction.

This unit analyzes examples of typical processing tasks that can be accomplished with utilities.

Although available utilities, required JCL, and control statements will vary with different installations, this unit provides a sampling of the tasks that utilities can perform.

The material also introduces a process that can be applied in choosing and executing any utility for processing needs.



Sample utility application.

Course objectives.

Be able to:

- **Determine the job stream.**
- **Print sequential data sets.**
- **Maintain source libraries.**
- **List PDS directories.**

Determining the job stream.

Utility programs can be used in a variety of situations to provide the processing required.

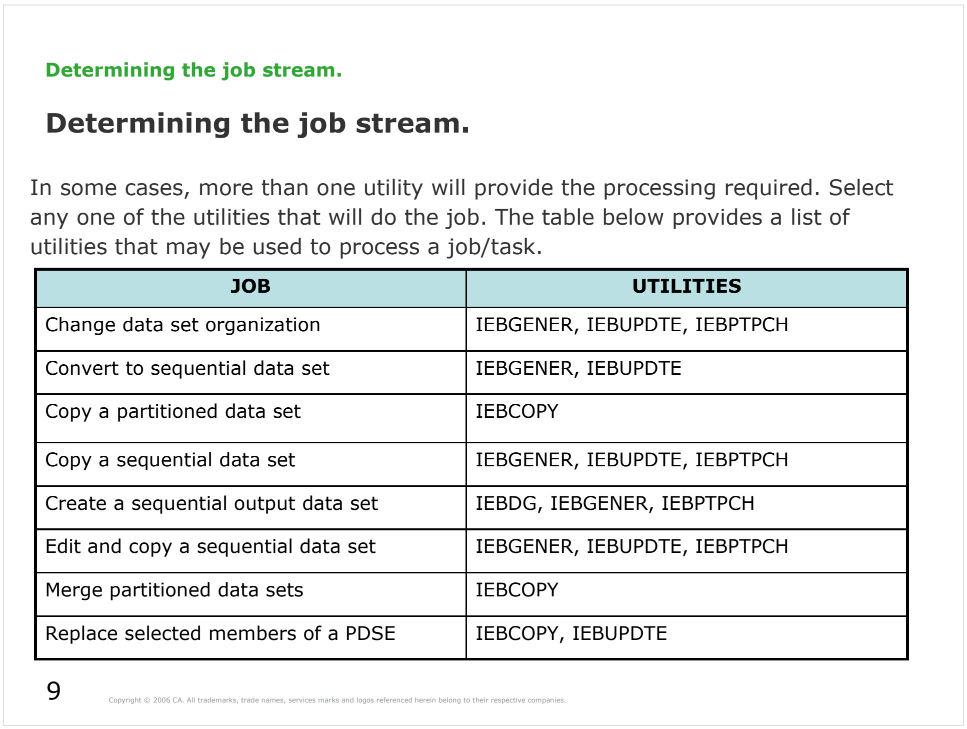
How to determine job streams?

An effective method for determining the job stream that will accomplish the necessary processing is as follows:

1. Use the Utilities Manual table to find the utility that may be appropriate to accomplish the task.

2. Once an appropriate utility or utilities is selected, read the chapters of the Utilities Manual that describe the utility in detail.

3. Look for an example that approximates the processing requirements in the Utilities Manual. Modify the example (as needed) to accomplish the task.



Determining the job stream.

Determining the job stream.

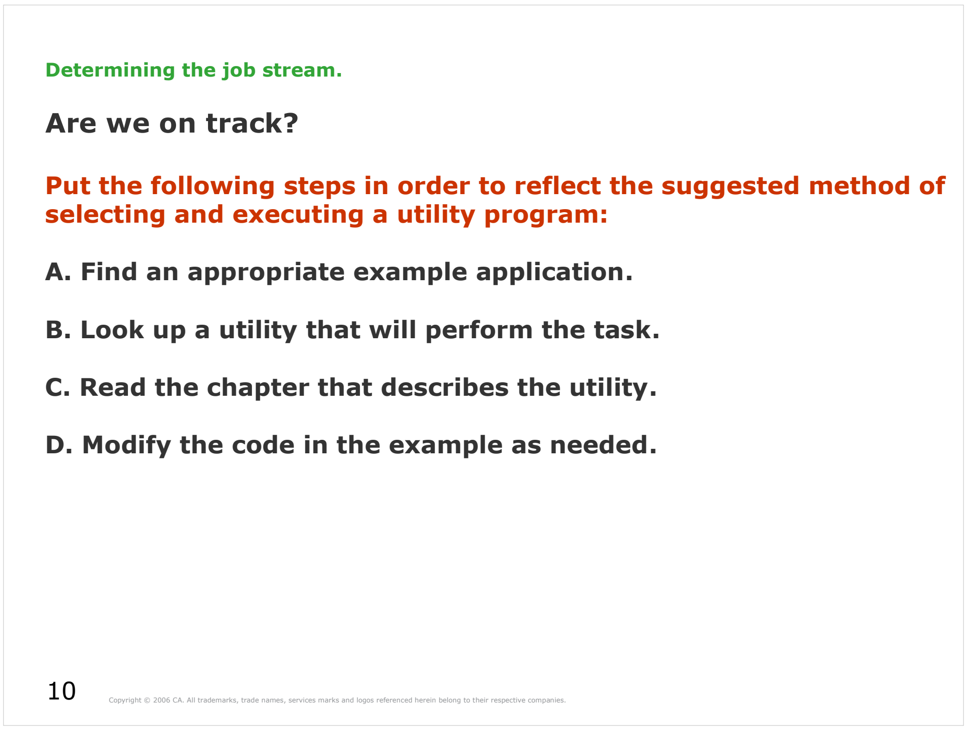
In some cases, more than one utility will provide the processing required. Select any one of the utilities that will do the job. The table below provides a list of utilities that may be used to process a job/task.

JOB	UTILITIES
Change data set organization	IEBGENER, IEBUPDTE, IEBPTPCH
Convert to sequential data set	IEBGENER, IEBUPDTE
Copy a partitioned data set	IEBCOPY
Copy a sequential data set	IEBGENER, IEBUPDTE, IEBPTPCH
Create a sequential output data set	IEBDG, IEBGENER, IEBPTPCH
Edit and copy a sequential data set	IEBGENER, IEBUPDTE, IEBPTPCH
Merge partitioned data sets	IEBCOPY
Replace selected members of a PDSE	IEBCOPY, IEBUPDTE

9

Usually you will select the one you are most familiar with and that requires the least amount of utility control statement coding.

See the „z/OS DFSMSdfp Utilities“ book. 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“.



Determining the job stream.

Are we on track?

Put the following steps in order to reflect the suggested method of selecting and executing a utility program:

A. Find an appropriate example application.

B. Look up a utility that will perform the task.

C. Read the chapter that describes the utility.

D. Modify the code in the example as needed.

10

The correct order is B., C., A., D.

Print sequential data sets.

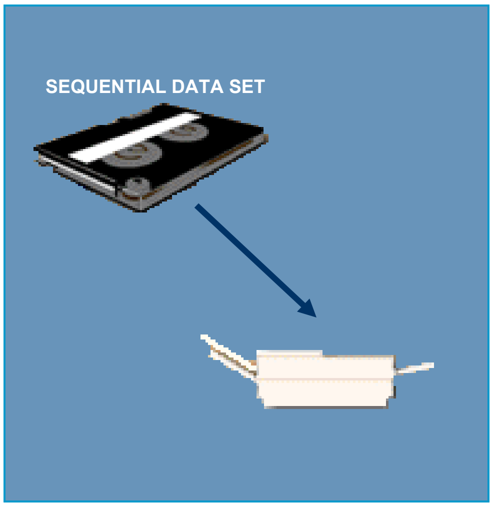
Selecting a utility.

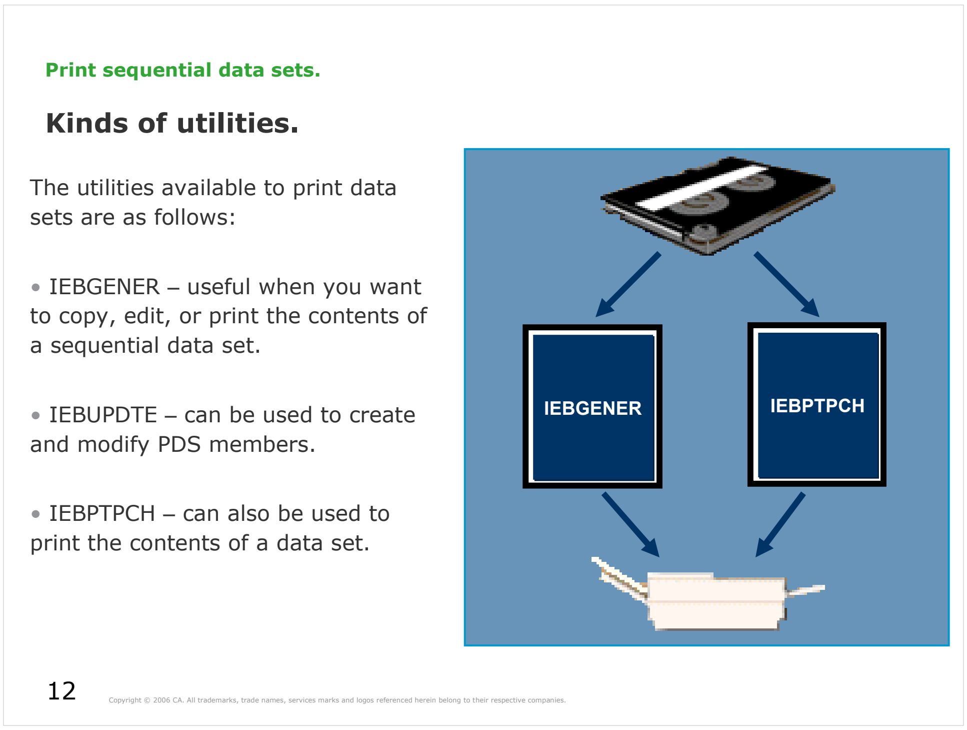
How to select a utility?

Assume you need to list the contents of a data set in order to determine what it contains. The data set resides on a tape volume, as shown on the right. In this case, consider the following:

1. The need to print the contents of a data set.

2. The data set resides on a tape volume, which implies a sequential data set.



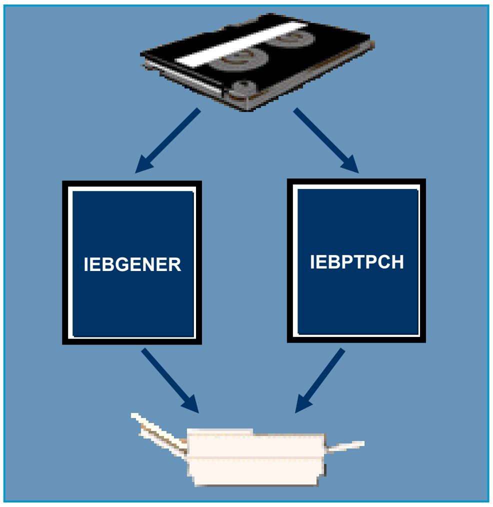


Print sequential data sets.

Kinds of utilities.

The utilities available to print data sets are as follows:

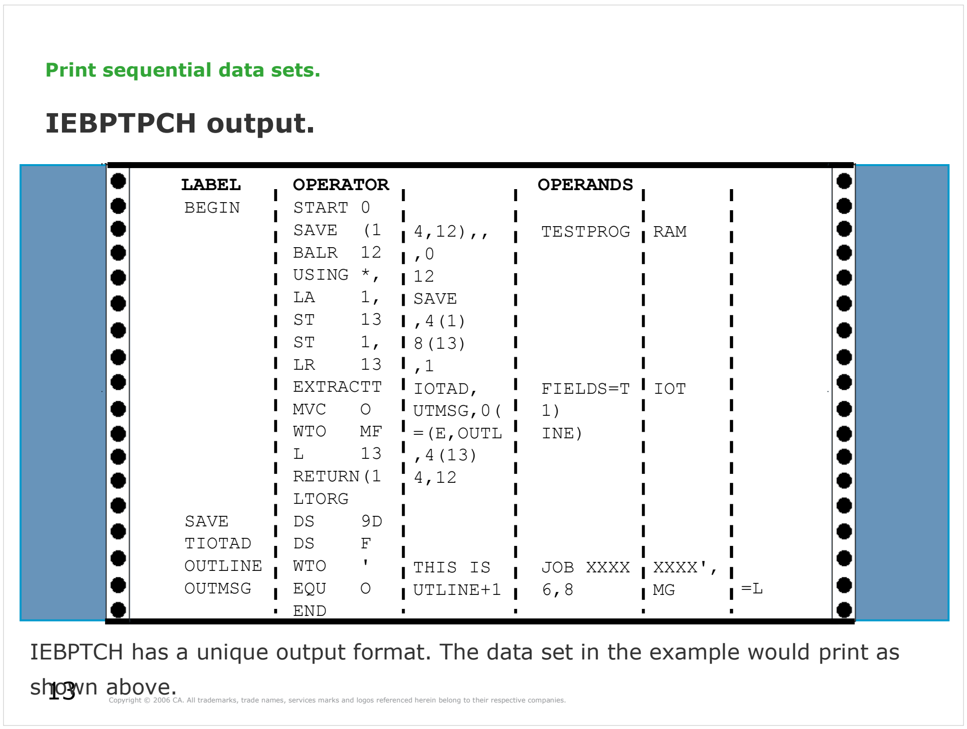
- IEBGENER – useful when you want to copy, edit, or print the contents of a sequential data set.
- IEBUPDTE – can be used to create and modify PDS members.
- IEBPTPCH – can also be used to print the contents of a data set.



12

The IEBUPDTE utility is not appropriate for this application because it does not involve printing. Both IEBGENER and IEBPTPCH can print sequential data sets.

The examples in the Utilities Manual indicate that IEBPTPCH requires control statements, whereas IEBGENER does not.



Print sequential data sets.

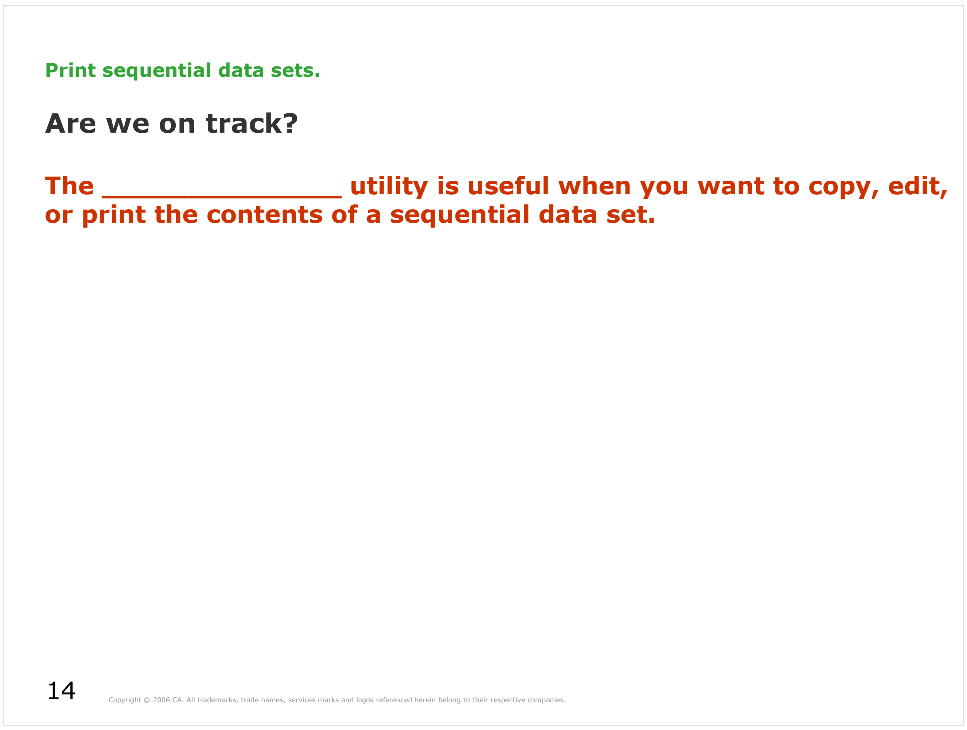
IEBPTPCH output.

```
LABEL     OPERATOR            OPERANDS
BEGIN     START 0
          SAVE  (1  4,12),,   TESTPROG  RAM
          BALR  12  ,0
          USING *,  12
          LA    1,  SAVE
          ST    13  ,4(1)
          ST    1,  8(13)
          LR    13  ,1
          EXTRACTT  IOTAD,    FIELDS=T  IOT
          MVC   O   UTMSG,0(  1)
          WTO   MF  =(E,OUTL  INE)
          L     13  ,4(13)
          RETURN(1  4,12
          LTORG
SAVE      DS    9D
TIOTAD    DS    F
OUTLINE   WTO   '   THIS IS   JOB XXXX  XXXX',    
OUTMSG    EQU   O   UTLINE+1  6,8       MG        =L
          END
```

IEBPTCH has a unique output format. The data set in the example would print as shown above.

13

Although this solution seems like it will do the job, IEBPTPCH has a unique output format. Groups of eight characters are transferred from the input record to the output record, separated by two blanks. This format repeats until all input characters of a record are transferred to the output record.



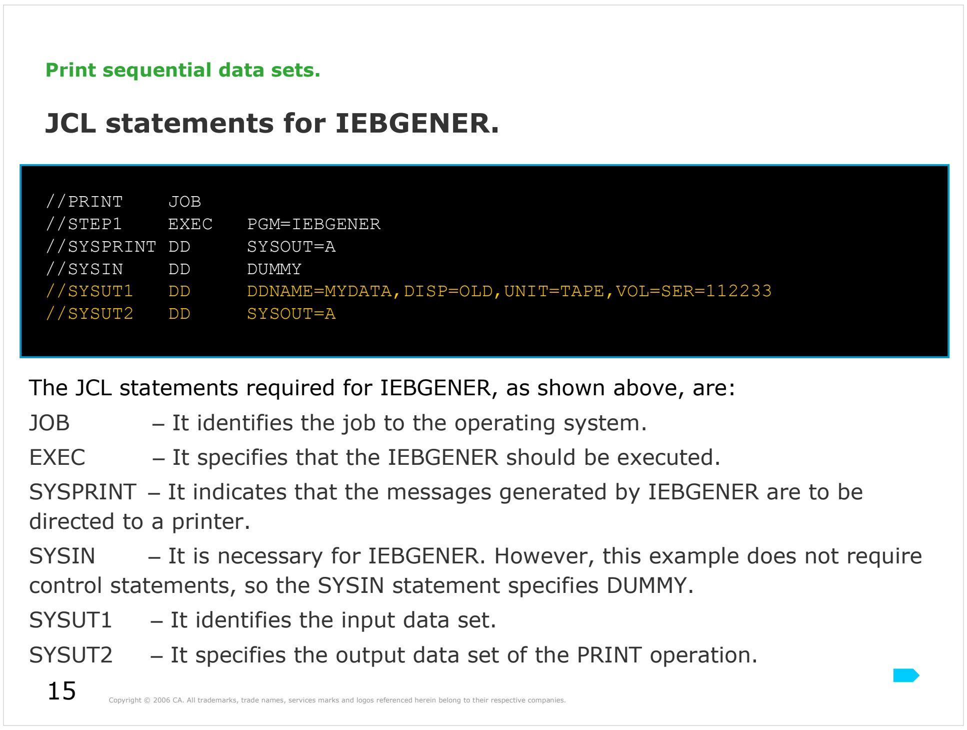
Print sequential data sets.

Are we on track?

The ________________ utility is useful when you want to copy, edit, or print the contents of a sequential data set.

14

The correct answer is IEBGENER.



Print sequential data sets.

JCL statements for IEBGENER.

```
//PRINT    JOB
//STEP1    EXEC   PGM=IEBGENER
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     DUMMY
//SYSUT1   DD     DDNAME=MYDATA,DISP=OLD,UNIT=TAPE,VOL=SER=112233
//SYSUT2   DD     SYSOUT=A
```

The JCL statements required for IEBGENER, as shown above, are:

JOB – It identifies the job to the operating system.

EXEC – It specifies that the IEBGENER should be executed.

SYSPRINT – It indicates that the messages generated by IEBGENER are to be directed to a printer.

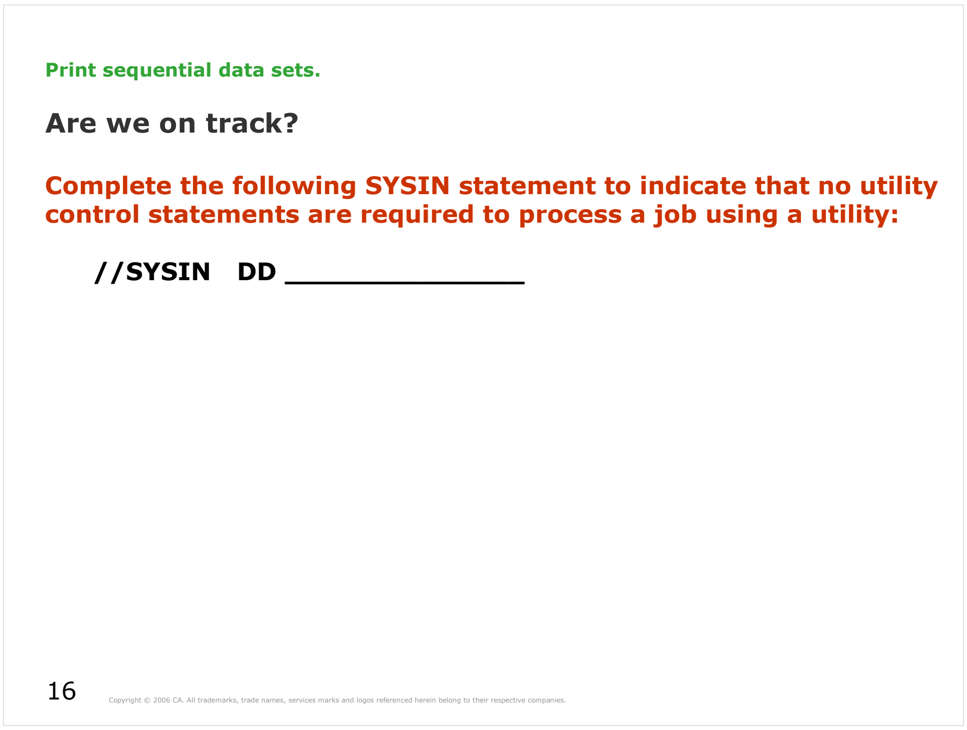
SYSIN – It is necessary for IEBGENER. However, this example does not require control statements, so the SYSIN statement specifies DUMMY.

SYSUT1 – It identifies the input data set.

SYSUT2 – It specifies the output data set of the PRINT operation.

15

DCB information may also be specified on the SYSUT2 statement. This DCB information specifies the output record and block sizes. In this case, you omit DCB information from the SYSUT2 statement. By default the utility uses the same record and block size on output as it receives from input.



Print sequential data sets.

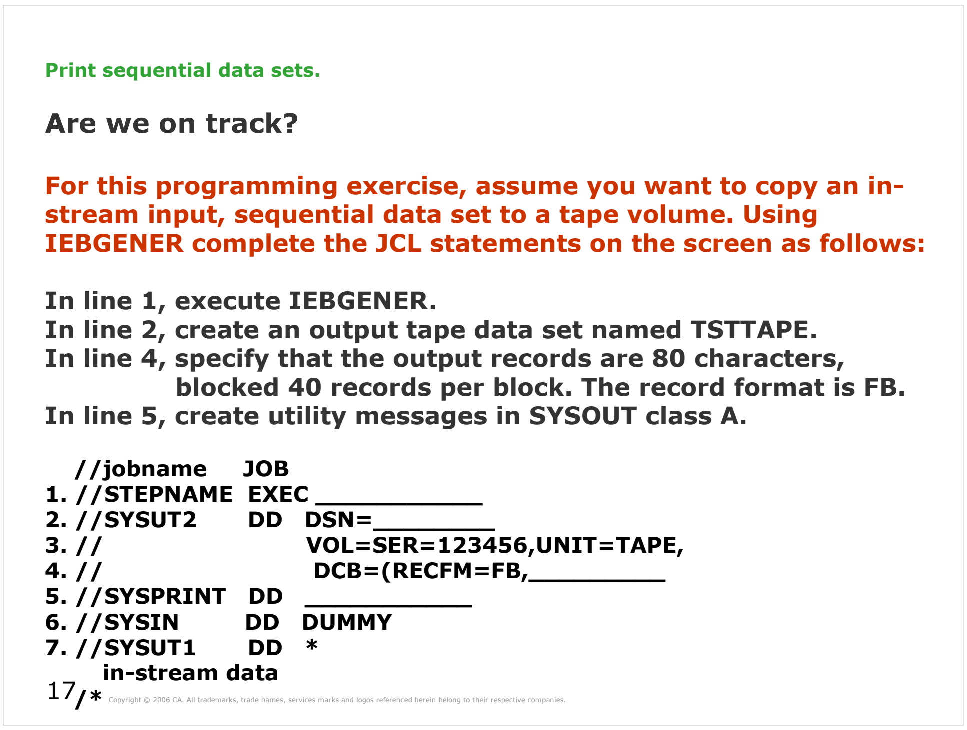
Are we on track?

Complete the following SYSIN statement to indicate that no utility control statements are required to process a job using a utility:

```
//SYSIN   DD _______________
```

16

The correct answer is DUMMY.



Print sequential data sets.

Are we on track?

For this programming exercise, assume you want to copy an in-stream input, sequential data set to a tape volume. Using IEBGENER complete the JCL statements on the screen as follows:

In line 1, execute IEBGENER.
In line 2, create an output tape data set named TSTTAPE.
In line 4, specify that the output records are 80 characters, blocked 40 records per block. The record format is FB.
In line 5, create utility messages in SYSOUT class A.

```
   //jobname    JOB
1. //STEPNAME  EXEC ____________
2. //SYSUT2     DD   DSN=_________
3. //                VOL=SER=123456,UNIT=TAPE,
4. //                 DCB=(RECFM=FB,__________
5. //SYSPRINT   DD   ____________
6. //SYSIN      DD   DUMMY
7. //SYSUT1     DD   *
     in-stream data
   /*
```

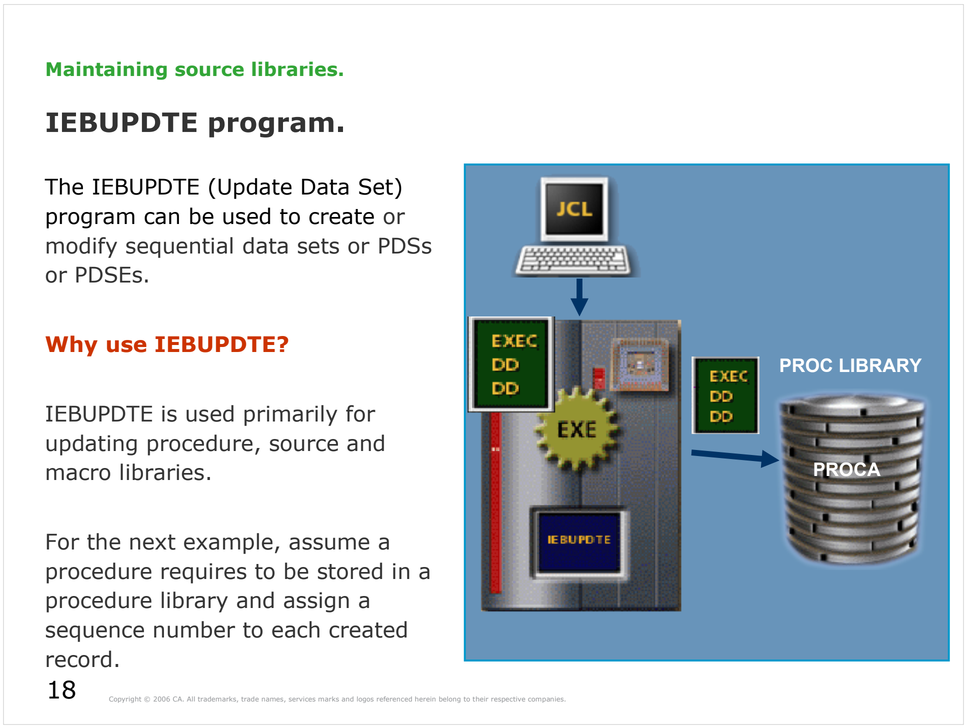
The correct answer is:

1. PGM=IEBGENER

2. TSTTAPE

4. LRECL=80,BLKSIZE=3200)

5. SYSOUT=A



Maintaining source libraries.

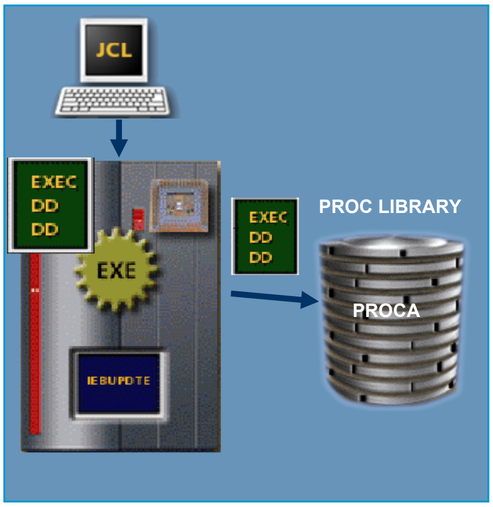
IEBUPDTE program.

The IEBUPDTE (Update Data Set) program can be used to create or modify sequential data sets or PDSs or PDSEs.

Why use IEBUPDTE?

IEBUPDTE is used primarily for updating procedure, source and macro libraries.

For the next example, assume a procedure requires to be stored in a procedure library and assign a sequence number to each created record.



18

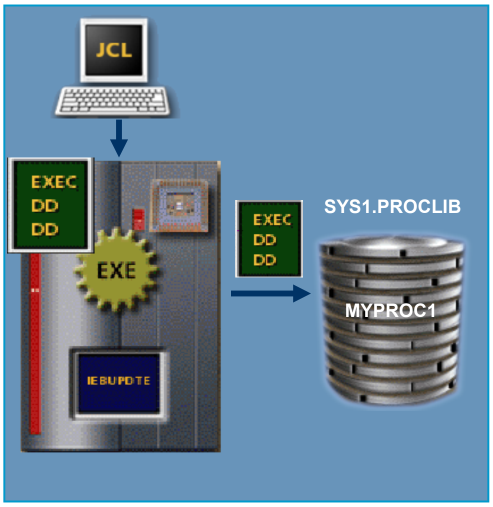
These sequence numbers will be used later if changes need to be made to individual records within the procedure.

Maintaining source libraries.

IEBUPDTE program – an example.

Assume the processing requirements are as follows:

- The procedure will be entered in the job stream.
- The output data set is a procedure library named SYS1.PROCLIB.
- The new member will be named MYPROC1.
- A unique sequence number should be given to each record of the output member. The starting sequence number is to be 10 and the numbers should be incremented by 10.
- Each record should be listed as it is added to the PDS.



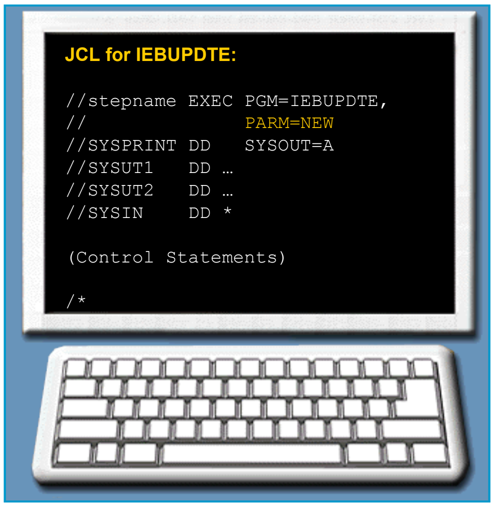
Maintaining source libraries.

General form for IEBUPDTE.

The example on the right shows the general form for IEBUPDTE.

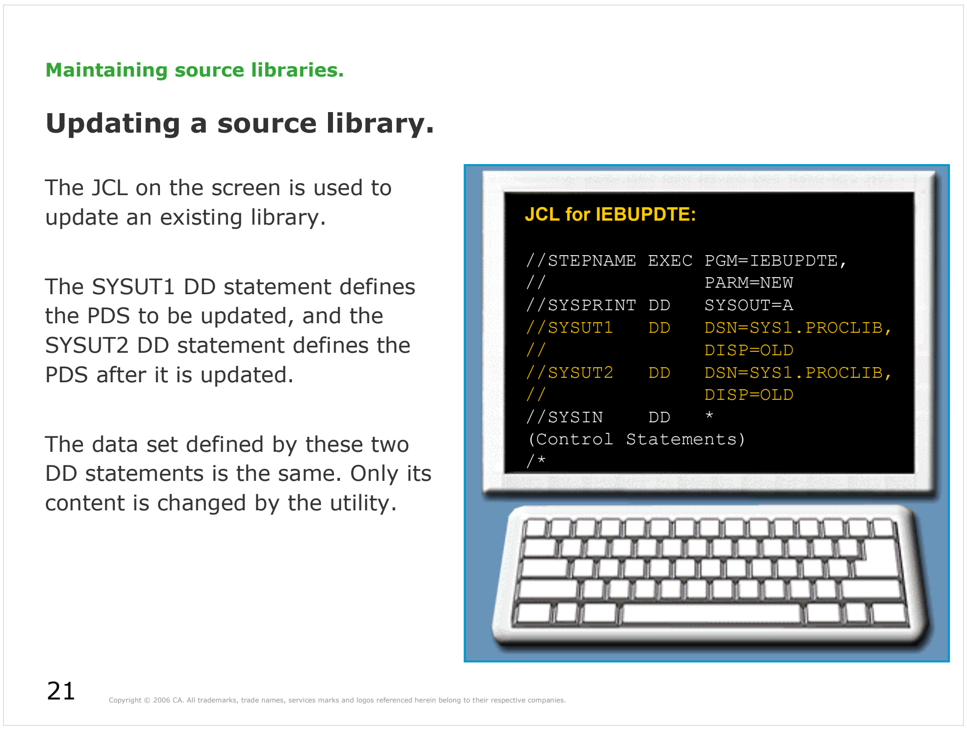
When adding a new member to an existing PDS, specify PARM=NEW on the EXEC statement invoking IEBUPDTE.

PARM=MOD would indicate that the output data set existed and has to be modified. MOD is the default if PARM is omitted.


JCL for IEBUPDTE:

```
//stepname EXEC PGM=IEBUPDTE,
//              PARM=NEW
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD …
//SYSUT2   DD …
//SYSIN    DD *

(Control Statements)

/*
```

20



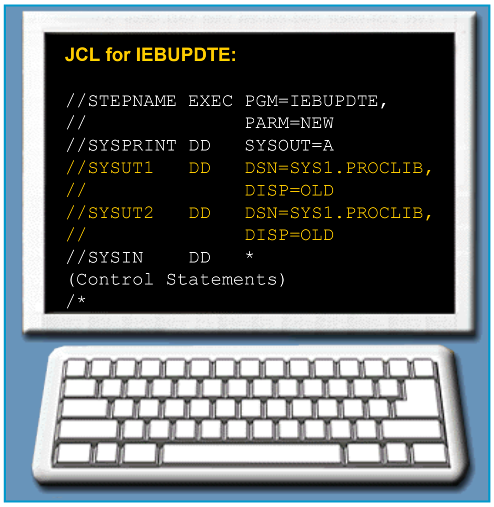
Maintaining source libraries.

Updating a source library.

The JCL on the screen is used to update an existing library.

The SYSUT1 DD statement defines the PDS to be updated, and the SYSUT2 DD statement defines the PDS after it is updated.

The data set defined by these two DD statements is the same. Only its content is changed by the utility.

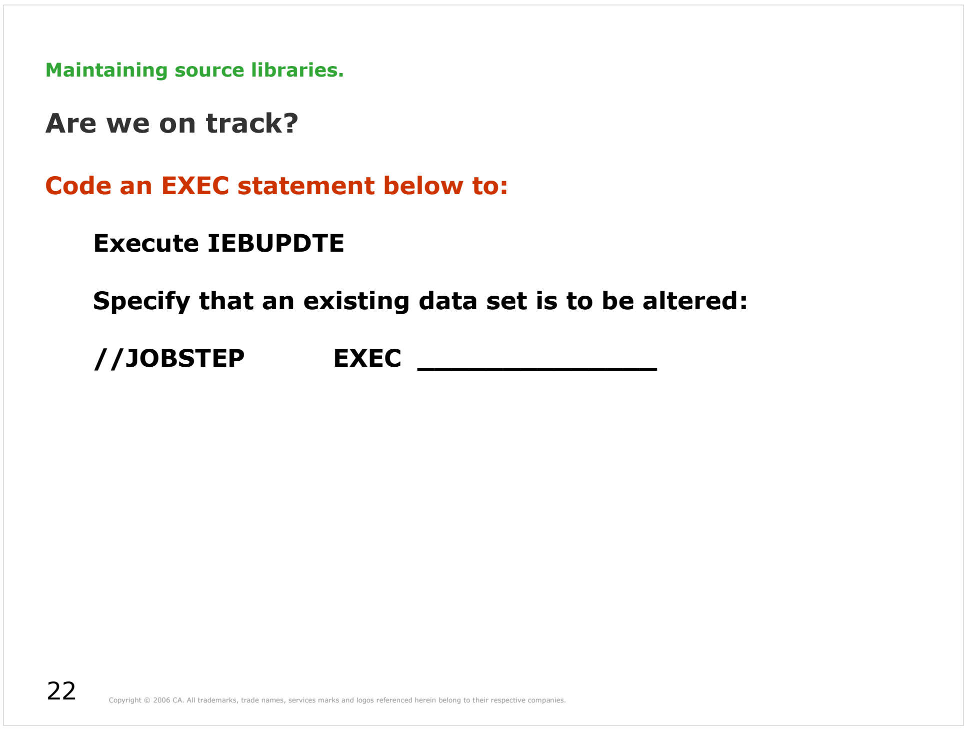


JCL for IEBUPDTE:

```
//STEPNAME EXEC PGM=IEBUPDTE,
//              PARM=NEW
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   DSN=SYS1.PROCLIB,
//              DISP=OLD
//SYSUT2   DD   DSN=SYS1.PROCLIB,
//              DISP=OLD
//SYSIN    DD   *
(Control Statements)
/*
```

21

See following slides to see the control statements...


Maintaining source libraries.

Are we on track?

Code an EXEC statement below to:

Execute IEBUPDTE

Specify that an existing data set is to be altered:

```
//JOBSTEP      EXEC _______________
```

22

The correct answer is PGM=IEBUPDTE,PARM=MOD

Maintaining source libraries.

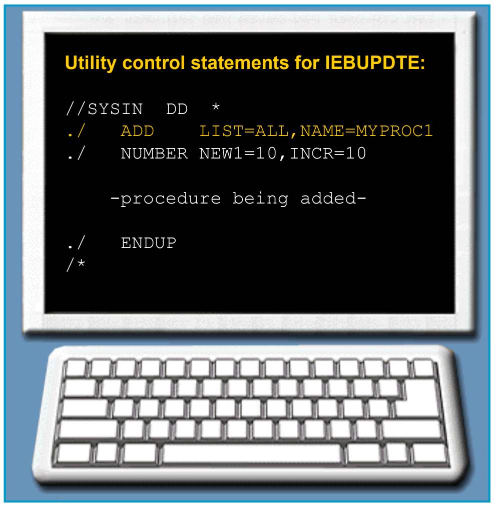
IEBUPDTE uses special utility control statements with the ./ characters in positions 1 and 2. These two characters distinguish the control statements from the input program that is also in the job stream.

The following control statements are required:

./ ADD indicates that a new member is to be added to the PDS.

LIST=ALL is an option to list the entire member on SYSPRINT.

NAME=names the newly created member is a procedure MYPROC1.



Utility control statements for IEBUPDTE:

```
//SYSIN   DD   *
./    ADD     LIST=ALL,NAME=MYPROC1
./    NUMBER  NEW1=10,INCR=10

    -procedure being added-

./    ENDUP
/*
```

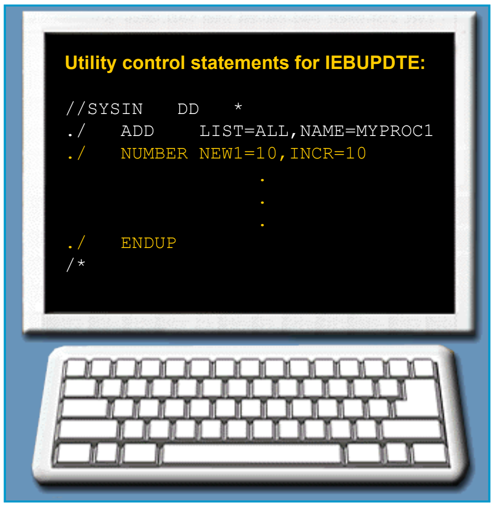
Maintainig source libraries.

./ NUMBER indicates that the new procedure is to be assigned sequence numbers.

NEW1= specifies the first sequence number (in this case, 10).

INCR=10 indicates that the records will be assigned sequence numbers in increments of 10 (10, 20,etc).

./ ENDUP marks the end of the member being added.



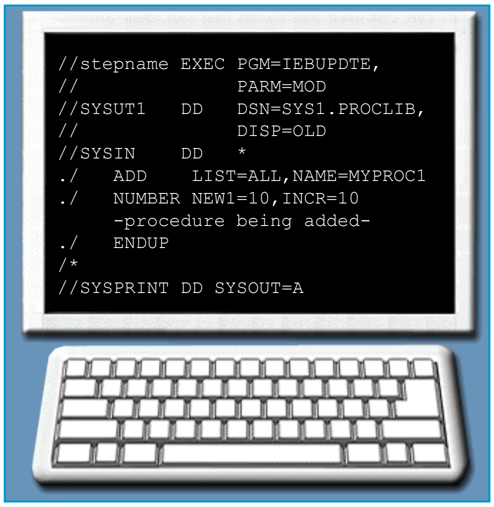
Utility control statements for IEBUPDTE:

```
//SYSIN    DD    *
./    ADD     LIST=ALL,NAME=MYPROC1
./    NUMBER  NEW1=10,INCR=10
                 .
                 .
                 .
./    ENDUP
/*
```

Maintaining source libraries.

The example on the right shows the job stream needed for these processing requirements.

Notice that the characters ./ in the control statements begin in column 1.



```
//stepname EXEC PGM=IEBUPDTE,
//              PARM=MOD
//SYSUT1   DD   DSN=SYS1.PROCLIB,
//              DISP=OLD
//SYSIN    DD   *
./   ADD    LIST=ALL,NAME=MYPROC1
./   NUMBER NEW1=10,INCR=10
     -procedure being added-
./   ENDUP
/*
//SYSPRINT DD SYSOUT=A
```

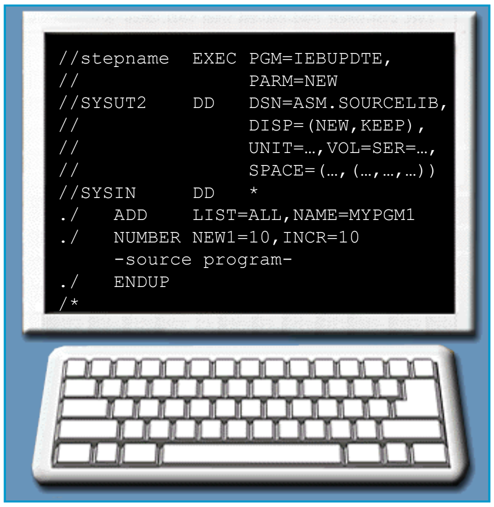
Maintaining source libraries.

In the previous example, a new member was added to an existing PDS.

If you changed the requirements to add a new member to a new PDS, the job stream would look like the screen on the right.

The output data set is ASM.SOURCLIB and the output member is named MYPGM1. In this case:

- PARM=NEW is on the EXEC statement. This specifies a new master data set.
- When creating a PDS, no //SYSUT1 DD is needed.
- The new partitioned master is defined on the SYSUT2 statement.



```
//stepname  EXEC PGM=IEBUPDTE,
//               PARM=NEW
//SYSUT2     DD   DSN=ASM.SOURCELIB,
//               DISP=(NEW,KEEP),
//               UNIT=…,VOL=SER=…,
//               SPACE=(…,(…,…,…))
//SYSIN      DD   *
./    ADD    LIST=ALL,NAME=MYPGM1
./    NUMBER NEW1=10,INCR=10
      -source program-
./    ENDUP
/*
```

Maintaining source libraries.

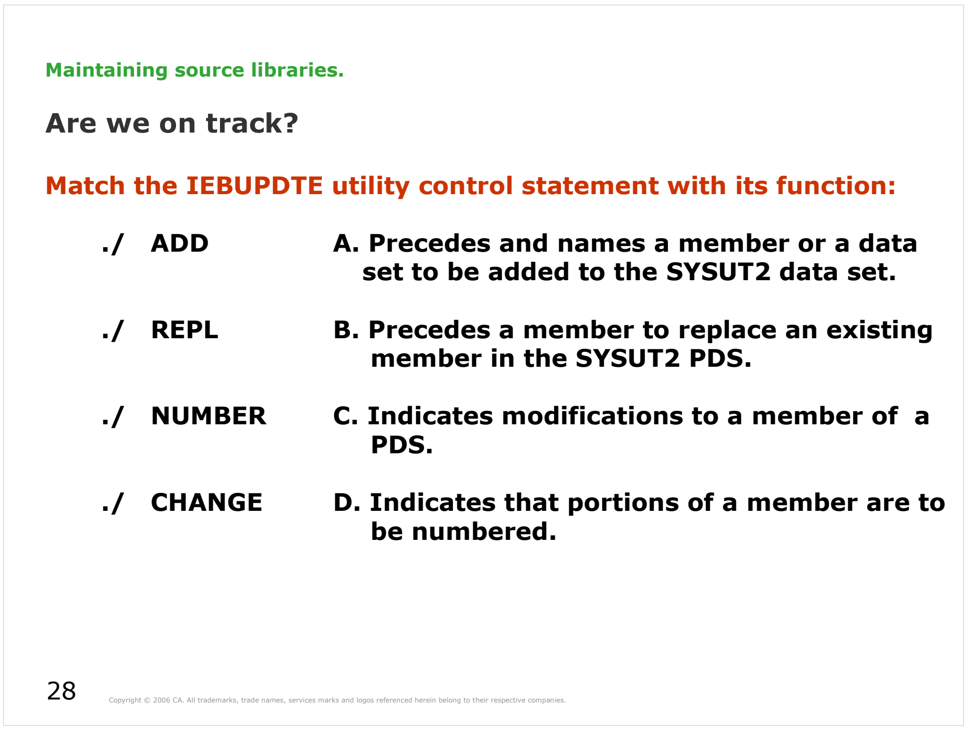
IEBUPDTE can be used to add (./ ADD) modify (./ CHANGE), or replace (./ REPL) members in a PDS, depending on the control statements used.

./ ADD precedes and names a member or data set to be added. The general form is as follows:

./ ADD NAME=member, LIST=ALL

./ CHANGE is used when deleting, numbering or adding data in a member or data set. It is followed by DELETE, NUMBER or data statements.

./ REPL precedes a member to replace an existing member in the SYSUT2 data set.



Maintaining source libraries.

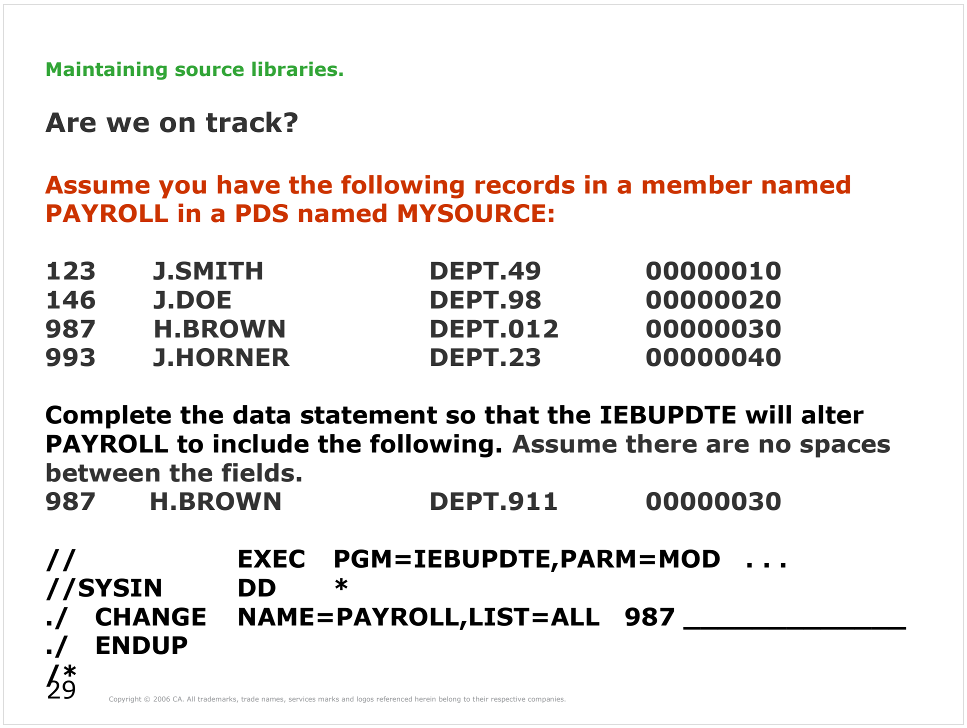
Are we on track?

Match the IEBUPDTE utility control statement with its function:

./ ADD	**A. Precedes and names a member or a data set to be added to the SYSUT2 data set.**
./ REPL	**B. Precedes a member to replace an existing member in the SYSUT2 PDS.**
./ NUMBER	**C. Indicates modifications to a member of a PDS.**
./ CHANGE	**D. Indicates that portions of a member are to be numbered.**

28

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



Maintaining source libraries.

Are we on track?

Assume you have the following records in a member named PAYROLL in a PDS named MYSOURCE:

```
123    J.SMITH            DEPT.49     00000010
146    J.DOE              DEPT.98     00000020
987    H.BROWN            DEPT.012    00000030
993    J.HORNER           DEPT.23     00000040
```

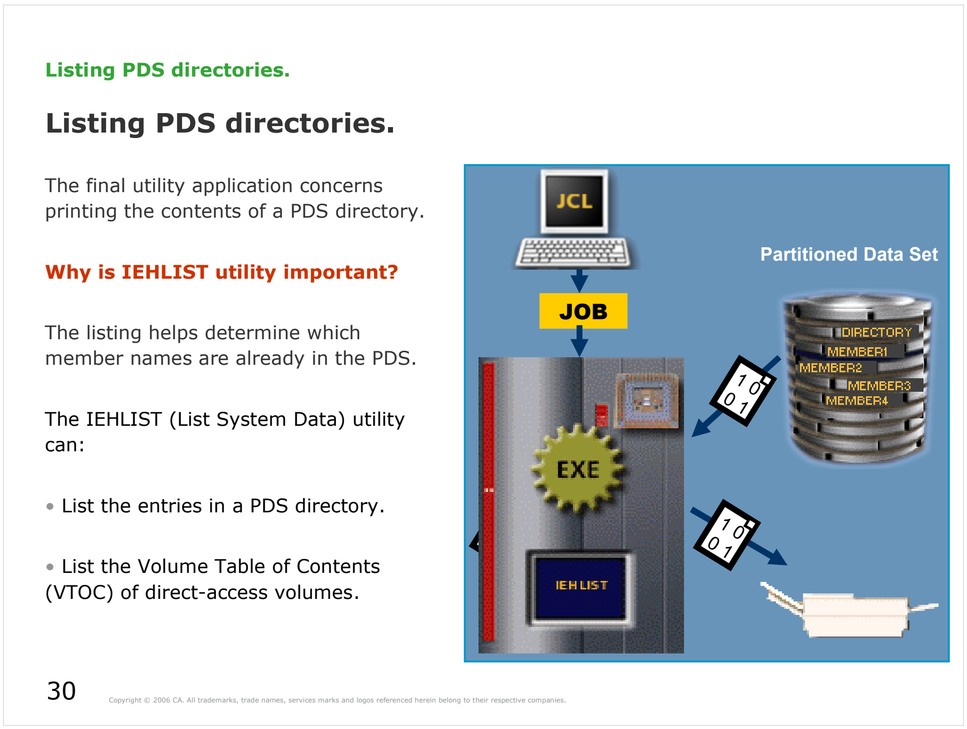
Complete the data statement so that the IEBUPDTE will alter PAYROLL to include the following. Assume there are no spaces between the fields.

```
987    H.BROWN            DEPT.911    00000030
```

```
//              EXEC   PGM=IEBUPDTE,PARM=MOD  . . .
//SYSIN         DD     *
./   CHANGE     NAME=PAYROLL,LIST=ALL   987 _______________
./   ENDUP
/*
```

29

The correct answer is H.BROWNDEPT.91100000030



Listing PDS directories.

Listing PDS directories.

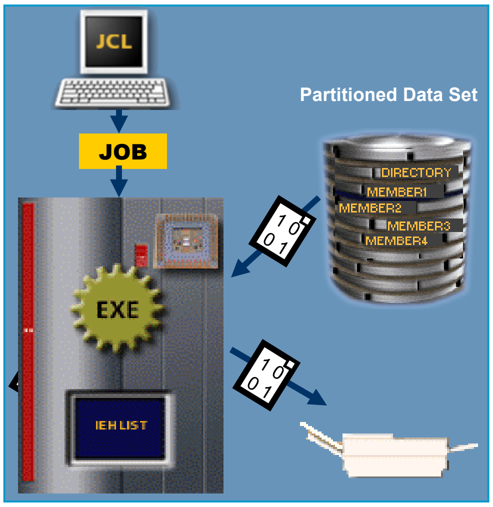
The final utility application concerns printing the contents of a PDS directory.

Why is IEHLIST utility important?

The listing helps determine which member names are already in the PDS.

The IEHLIST (List System Data) utility can:

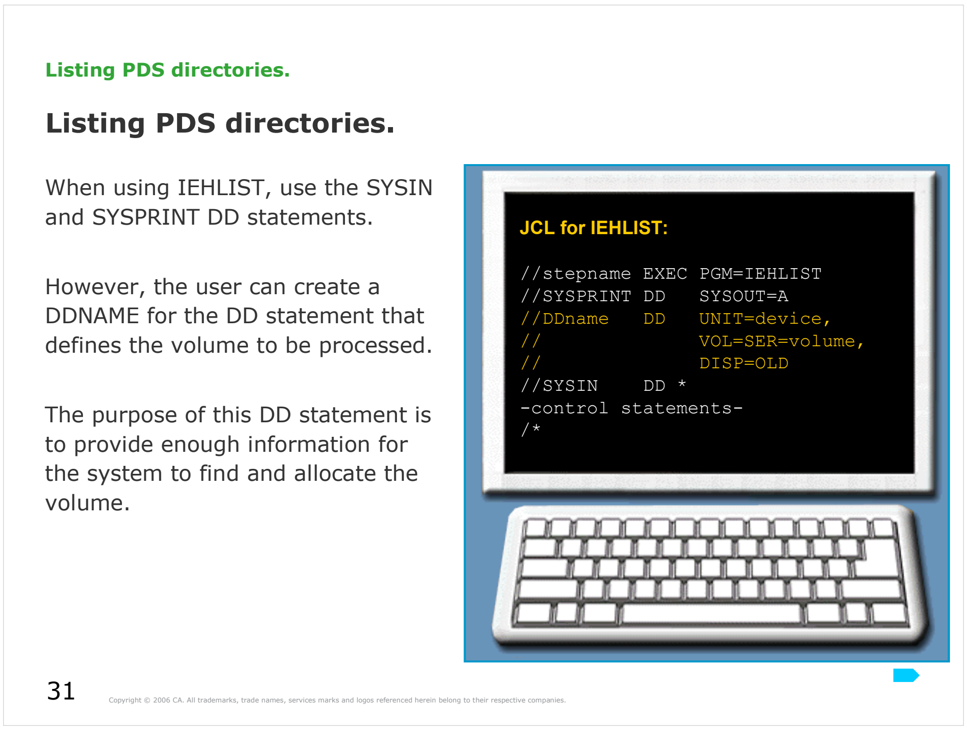
- List the entries in a PDS directory.
- List the Volume Table of Contents (VTOC) of direct-access volumes.



30

Printing the contents of a PDS directory is often advisable before attempting to add a new member to a PDS, since a unique name is required.

See MCOE.EDU.JCL.JCL(VTOCLST).



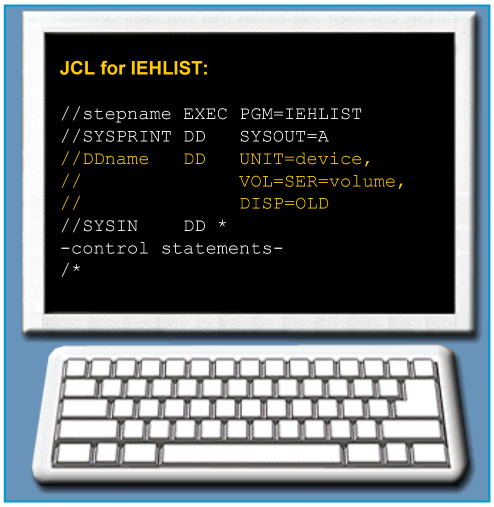
Listing PDS directories.

Listing PDS directories.

When using IEHLIST, use the SYSIN and SYSPRINT DD statements.

However, the user can create a DDNAME for the DD statement that defines the volume to be processed.

The purpose of this DD statement is to provide enough information for the system to find and allocate the volume.



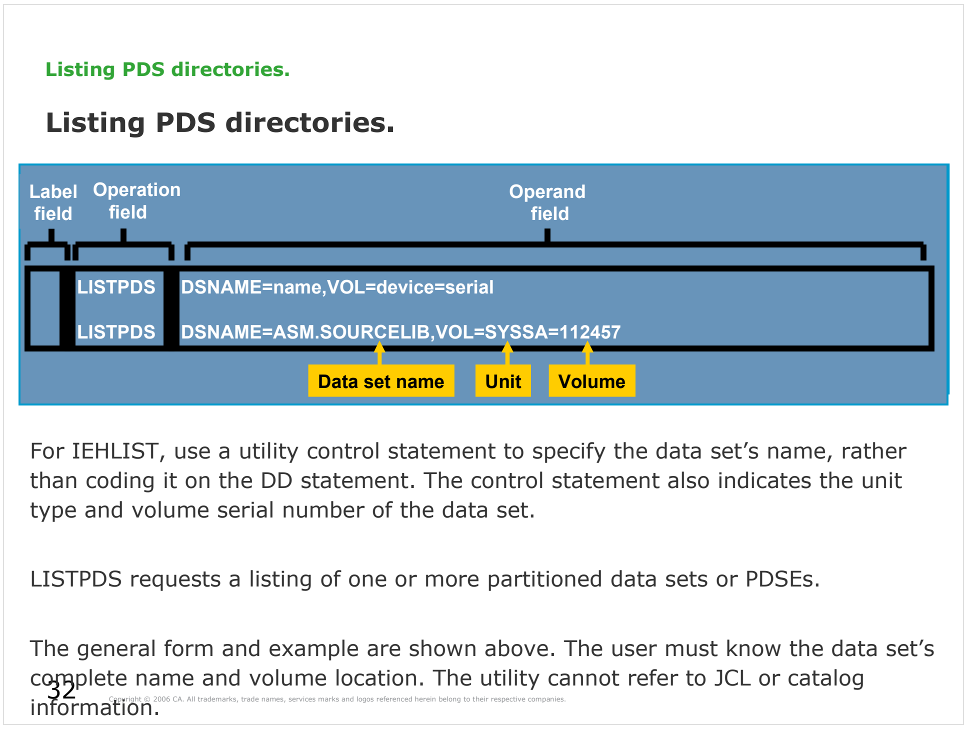
JCL for IEHLIST:

```
//stepname EXEC PGM=IEHLIST
//SYSPRINT DD   SYSOUT=A
//DDname   DD   UNIT=device,
//              VOL=SER=volume,
//              DISP=OLD
//SYSIN    DD *
-control statements-
/*
```

31

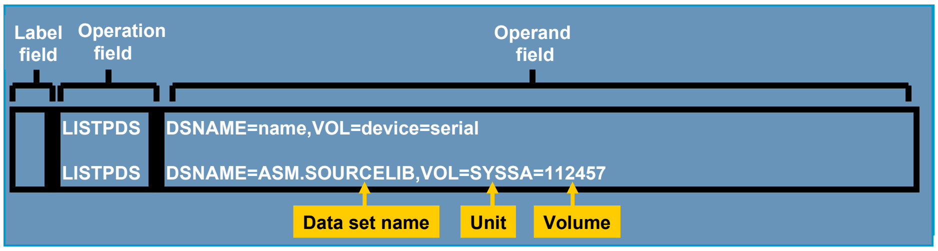
SYSIN is for IEHLIST control statements.

SYSPRINT is for the output.



Listing PDS directories.

Listing PDS directories.



```
LISTPDS  DSNAME=name,VOL=device=serial
LISTPDS  DSNAME=ASM.SOURCELIB,VOL=SYSSA=112457
```

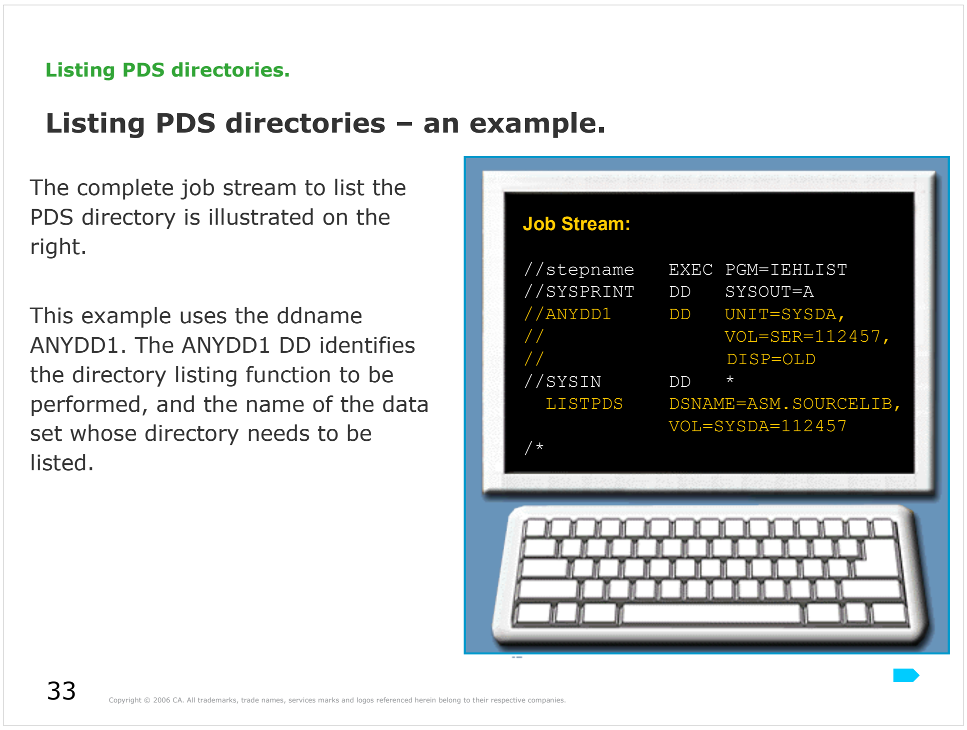
For IEHLIST, use a utility control statement to specify the data set’s name, rather than coding it on the DD statement. The control statement also indicates the unit type and volume serial number of the data set.

LISTPDS requests a listing of one or more partitioned data sets or PDSEs.

The general form and example are shown above. The user must know the data set’s complete name and volume location. The utility cannot refer to JCL or catalog information.

32

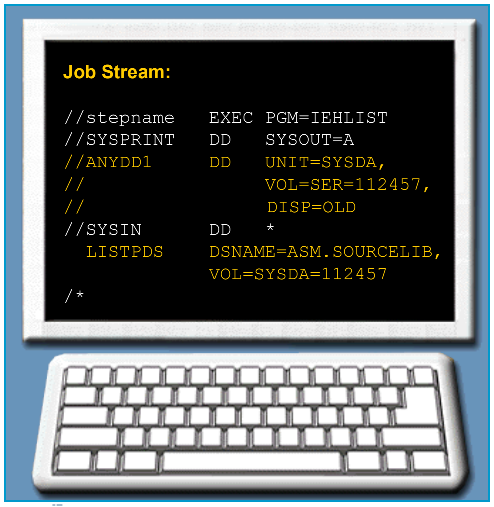
The format of the utility control statement differs from that of a JCL statement specifying volume and unit.


Listing PDS directories.

Listing PDS directories – an example.

The complete job stream to list the PDS directory is illustrated on the right.

This example uses the ddname ANYDD1. The ANYDD1 DD identifies the directory listing function to be performed, and the name of the data set whose directory needs to be listed.


Job Stream:

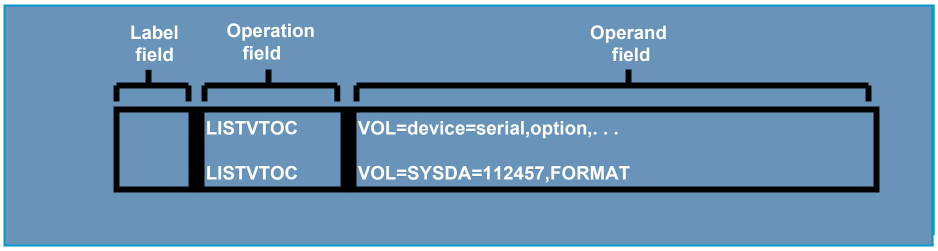
```
//stepname   EXEC PGM=IEHLIST
//SYSPRINT   DD   SYSOUT=A
//ANYDD1     DD   UNIT=SYSDA,
//                VOL=SER=112457,
//                DISP=OLD
//SYSIN      DD   *
  LISTPDS    DSNAME=ASM.SOURCELIB,
             VOL=SYSDA=112457
/*
```

33

ANYDD1 DD defines a device on which a disk volume (112457) is mounted.

Listing PDS directories.

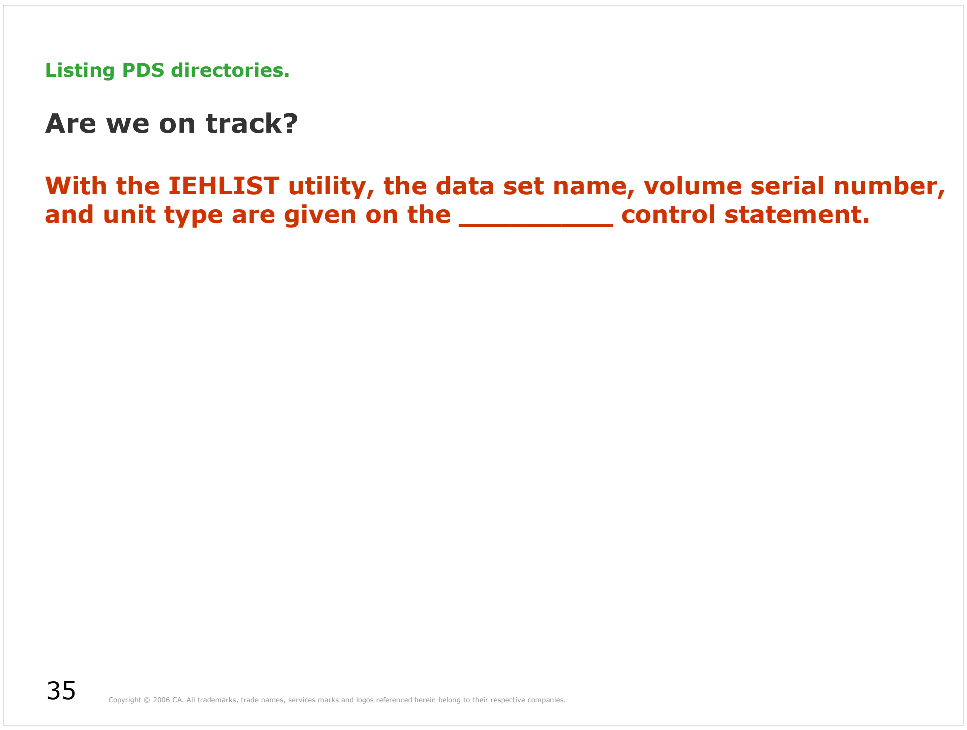
Listing Volume Table Of Contents – an example.



```
LISTVTOC   VOL=device=serial,option,. . .
LISTVTOC   VOL=SYSDA=112457,FORMAT
```

One other control statement is often used with the IEHLIST utility. LISTVTOC requests a listing of all or part of a volume table of contents.

The example above would list the volume table of contents of DASD volume 112457, using a unit designation of SYSDA. The FORMAT option lists the VTOC in edited form.

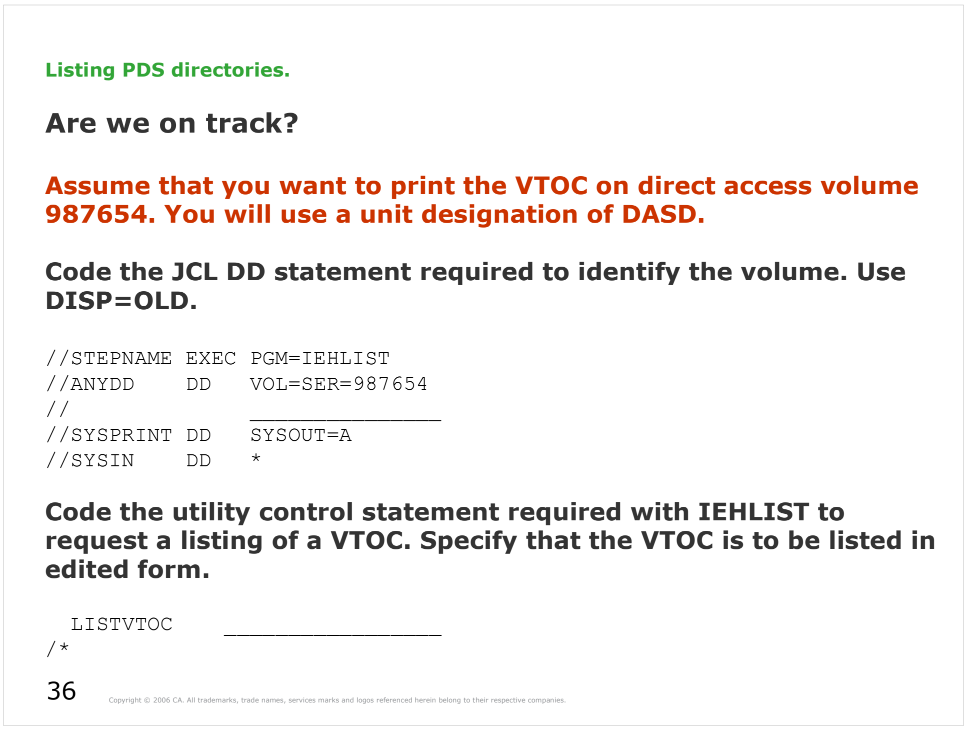


Listing PDS directories.

Are we on track?

With the IEHLIST utility, the data set name, volume serial number, and unit type are given on the __________ control statement.

The correct answer is LISTPDS.



Listing PDS directories.

Are we on track?

Assume that you want to print the VTOC on direct access volume 987654. You will use a unit designation of DASD.

Code the JCL DD statement required to identify the volume. Use DISP=OLD.

```
//STEPNAME EXEC PGM=IEHLIST
//ANYDD     DD   VOL=SER=987654
//             _______________
//SYSPRINT DD   SYSOUT=A
//SYSIN     DD   *
```

Code the utility control statement required with IEHLIST to request a listing of a VTOC. Specify that the VTOC is to be listed in edited form.

```
  LISTVTOC    _________________
/*
```

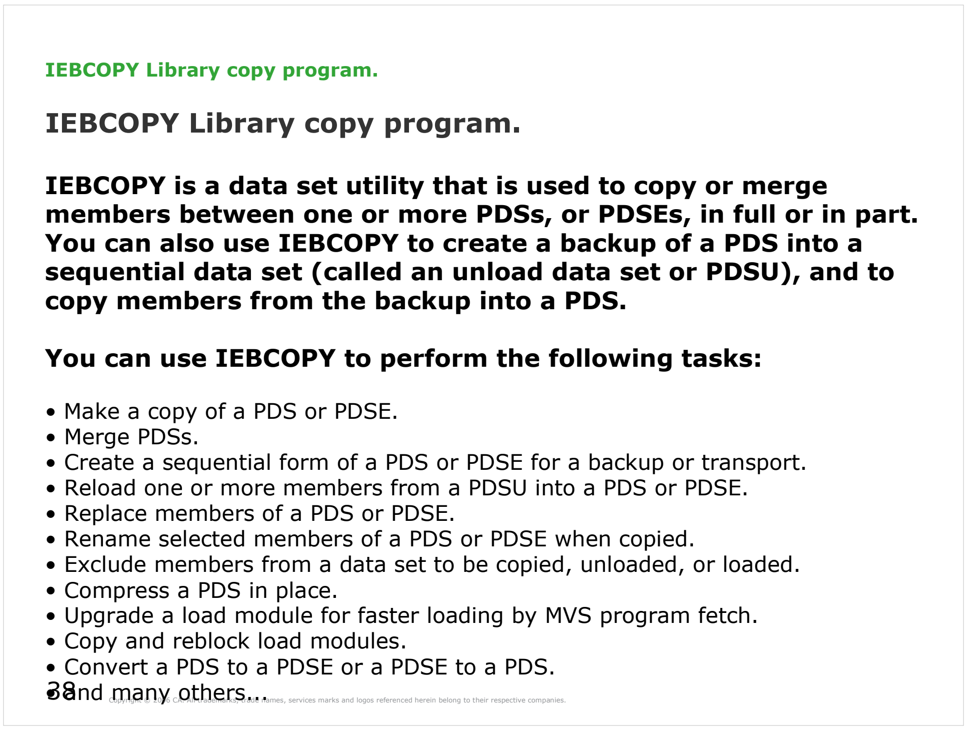
36

The correct answer is:

1. SER=987654,UNIT=DASD,DISP=OLD
2. FORMAT,VOL=DASD=987654

Glossary.

LISTPDS
An IEHLIST control statement that lists the directory of a partitioned data set.



IEBCOPY Library copy program.

IEBCOPY Library copy program.

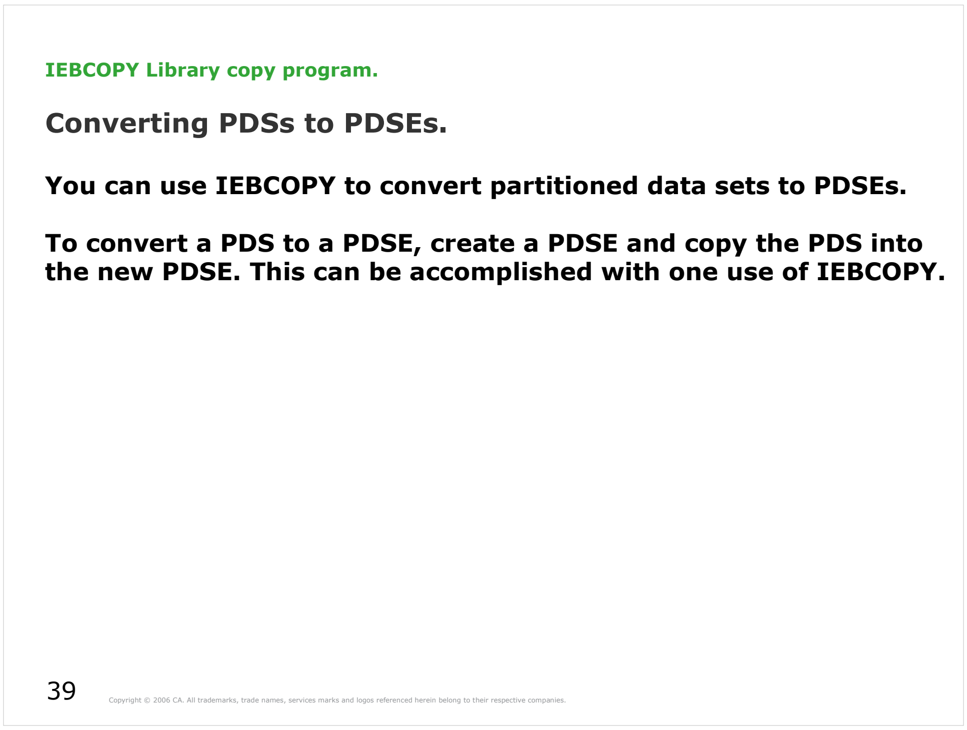
IEBCOPY is a data set utility that is used to copy or merge members between one or more PDSs, or PDSEs, in full or in part. You can also use IEBCOPY to create a backup of a PDS into a sequential data set (called an unload data set or PDSU), and to copy members from the backup into a PDS.

You can use IEBCOPY to perform the following tasks:

- Make a copy of a PDS or PDSE.
- Merge PDSs.
- Create a sequential form of a PDS or PDSE for a backup or transport.
- Reload one or more members from a PDSU into a PDS or PDSE.
- Replace members of a PDS or PDSE.
- Rename selected members of a PDS or PDSE when copied.
- Exclude members from a data set to be copied, unloaded, or loaded.
- Compress a PDS in place.
- Upgrade a load module for faster loading by MVS program fetch.
- Copy and reblock load modules.
- Convert a PDS to a PDSE or a PDSE to a PDS.
- and many others...

38

In addition, IEBCOPY automatically lists the number of unused directory blocks and the number of unused tracks that are available for member records if the output data set is a partitioned data set.



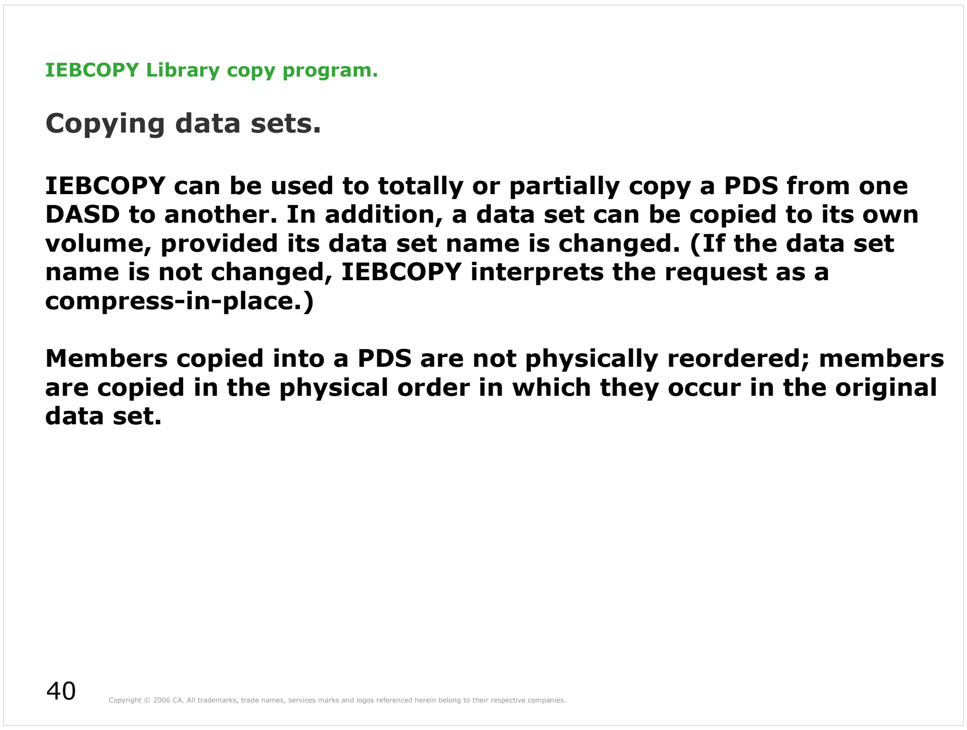
IEBCOPY Library copy program.

Converting PDSs to PDSEs.

You can use IEBCOPY to convert partitioned data sets to PDSEs.

To convert a PDS to a PDSE, create a PDSE and copy the PDS into the new PDSE. This can be accomplished with one use of IEBCOPY.

39



IEBCOPY Library copy program.

Copying data sets.

IEBCOPY can be used to totally or partially copy a PDS from one DASD to another. In addition, a data set can be copied to its own volume, provided its data set name is changed. (If the data set name is not changed, IEBCOPY interprets the request as a compress-in-place.)

Members copied into a PDS are not physically reordered; members are copied in the physical order in which they occur in the original data set.

40

See KOTMI01.WORK.JCL(COMPRESS).

Merging data sets.

Merging data sets is done by copying or loading the additional members to an existing PDS. The merge operation (ordering of the directory of the output data set) is automatically performed by IEBCOPY.

Increasing Directory Space for a PDS
IEBCOPY cannot increase the number of directory blocks in a PDS. (A PDSE directory automatically expands as needed.) If you are not sure there will be enough directory blocks in the output PDS you are merging to, then you should expand the output data set directory space before beginning the merge operation.

IEBCOPY Library copy program.

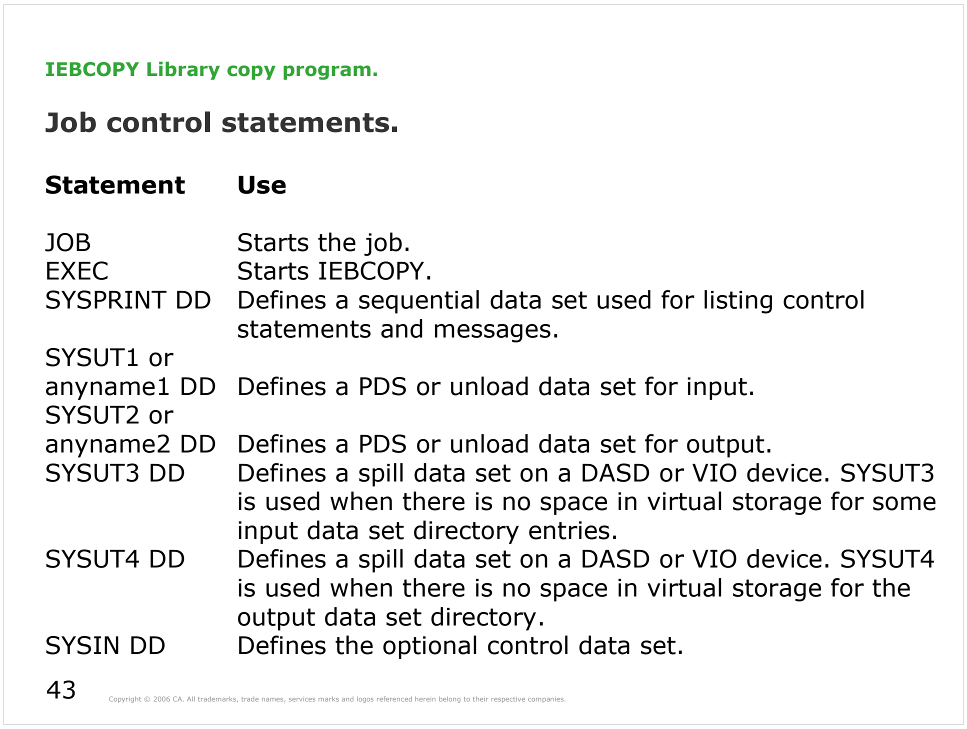
Unloading (backing up) data sets.

IEBCOPY can be used to create a backup copy of a PDS by copying (unloading) it to a sequential data set on DASD, tape, or other device supported by QSAM.

IEBCOPY creates an unload data set when you specify physical sequential organization (DSORG=PS) for the output data set. To create a PDS, specify DSORG=PO and DSNTYPE=PDS or DSNTYPE=LIBRARY.

To unload more than one PDS to the same tape volume in one execution of IEBCOPY, multiple copy operations must be used and multiple sequential data sets must be allocated to successive files on the tape.

The PDSE directory can contain attributes in addition to those traditionally kept in a PDS directory entry.
Some PDSE extended attributes are recorded on an unload data set and will be reloaded when the target is a PDSE.

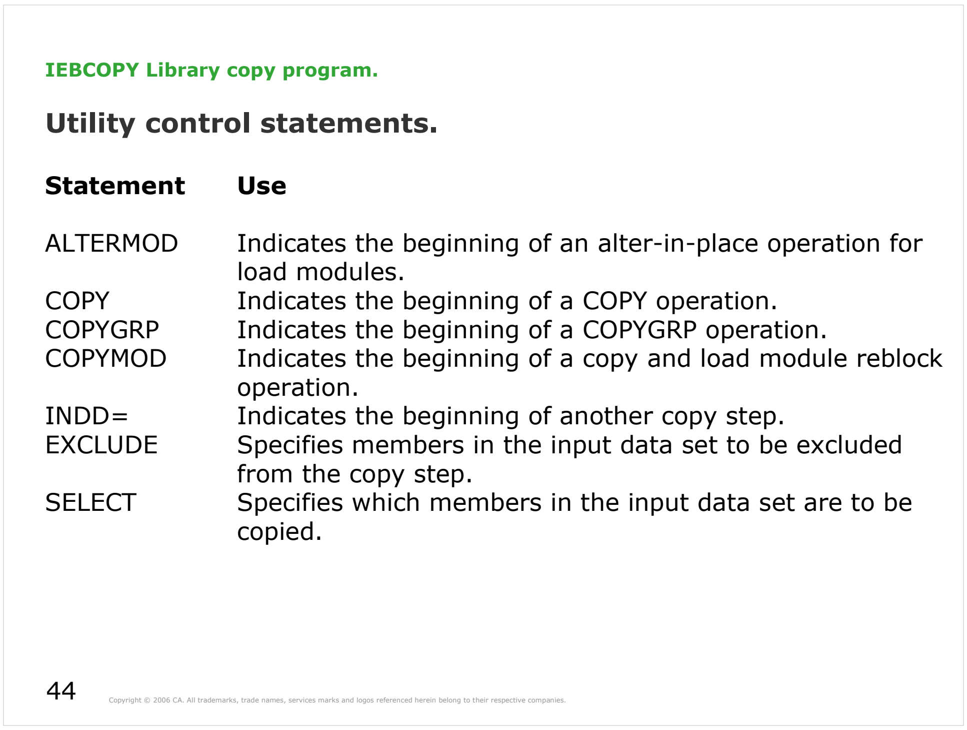


IEBCOPY Library copy program.

Job control statements.

Statement	Use
JOB	Starts the job.
EXEC	Starts IEBCOPY.
SYSPRINT DD	Defines a sequential data set used for listing control statements and messages.
SYSUT1 or anyname1 DD	Defines a PDS or unload data set for input.
SYSUT2 or anyname2 DD	Defines a PDS or unload data set for output.
SYSUT3 DD	Defines a spill data set on a DASD or VIO device. SYSUT3 is used when there is no space in virtual storage for some input data set directory entries.
SYSUT4 DD	Defines a spill data set on a DASD or VIO device. SYSUT4 is used when there is no space in virtual storage for the output data set directory.
SYSIN DD	Defines the optional control data set.

43



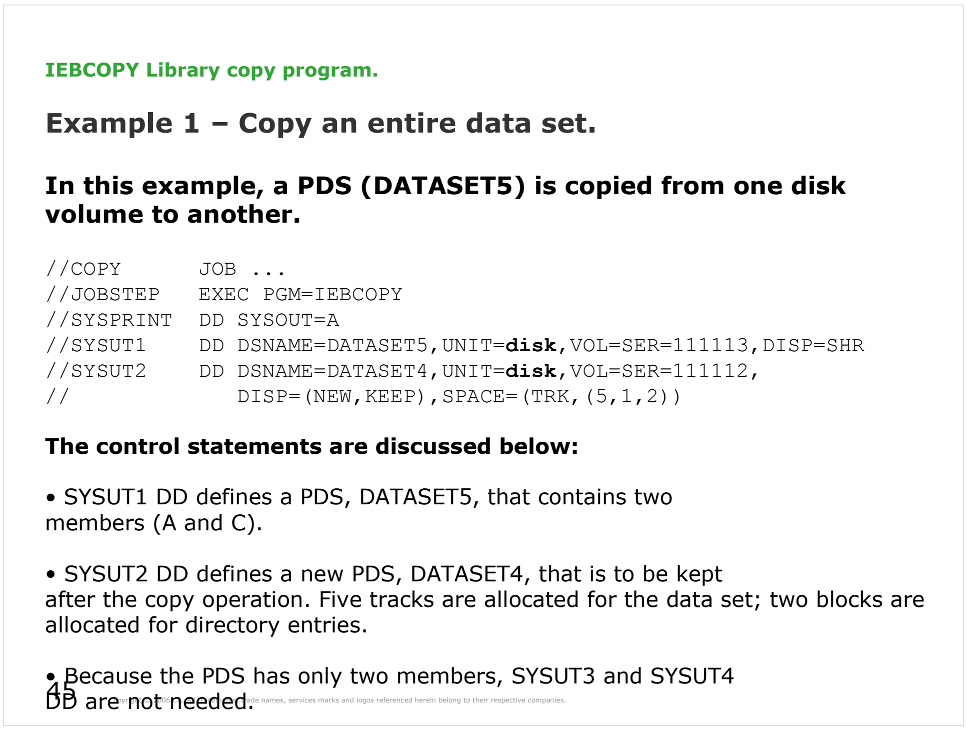
IEBCOPY Library copy program.

Utility control statements.

Statement	Use
ALTERMOD	Indicates the beginning of an alter-in-place operation for load modules.
COPY	Indicates the beginning of a COPY operation.
COPYGRP	Indicates the beginning of a COPYGRP operation.
COPYMOD	Indicates the beginning of a copy and load module reblock operation.
INDD=	Indicates the beginning of another copy step.
EXCLUDE	Specifies members in the input data set to be excluded from the copy step.
SELECT	Specifies which members in the input data set are to be copied.

44

See MCOE.EDU.JCL.JCL(LIBMT and MTLIB).


IEBCOPY Library copy program.

Example 1 – Copy an entire data set.

In this example, a PDS (DATASET5) is copied from one disk volume to another.

```
//COPY      JOB ...
//JOBSTEP   EXEC PGM=IEBCOPY
//SYSPRINT  DD SYSOUT=A
//SYSUT1    DD DSNAME=DATASET5,UNIT=disk,VOL=SER=111113,DISP=SHR
//SYSUT2    DD DSNAME=DATASET4,UNIT=disk,VOL=SER=111112,
//             DISP=(NEW,KEEP),SPACE=(TRK,(5,1,2))
```

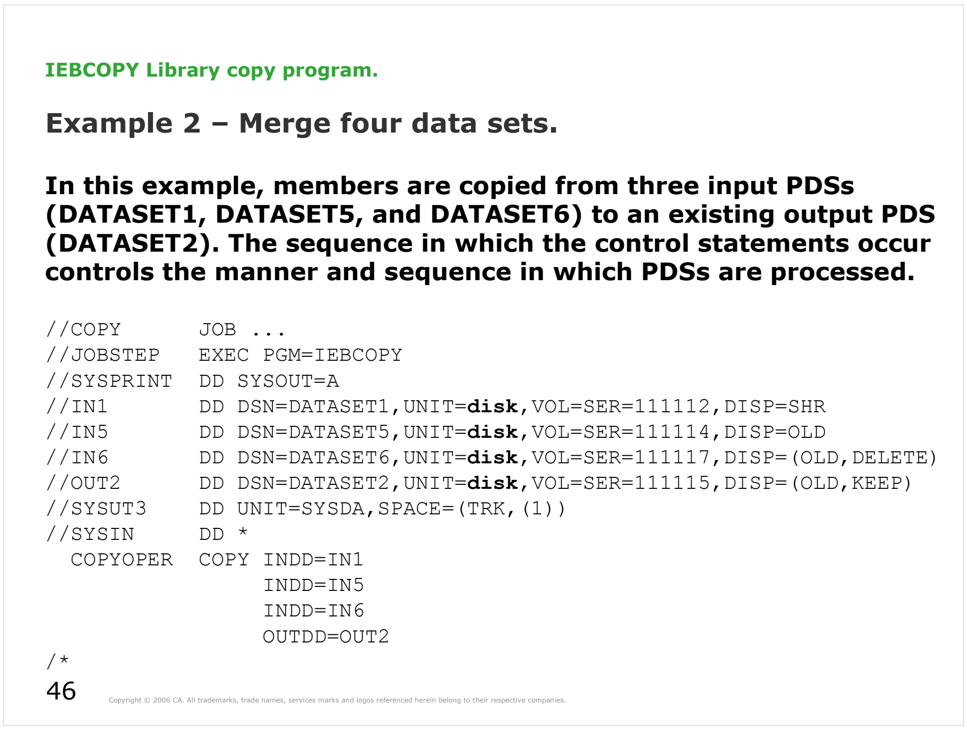
The control statements are discussed below:

• SYSUT1 DD defines a PDS, DATASET5, that contains two members (A and C).

• SYSUT2 DD defines a new PDS, DATASET4, that is to be kept after the copy operation. Five tracks are allocated for the data set; two blocks are allocated for directory entries.

• Because the PDS has only two members, SYSUT3 and SYSUT4 DD are not needed.

Because the input and output data sets are identified as SYSUT1 and SYSUT2, the SYSIN data set is not needed. The SYSUT1 data set will be copied in full to the SYSUT2 data set. After the copy operation is finished, DATASET4 will contain the same members that are in DATASET5. However, there will be no embedded, unused space in DATASET4. If you are copying a PDSE, the processing is the same, except that there is no embedded, unused space in a PDSE.



IEBCOPY Library copy program.

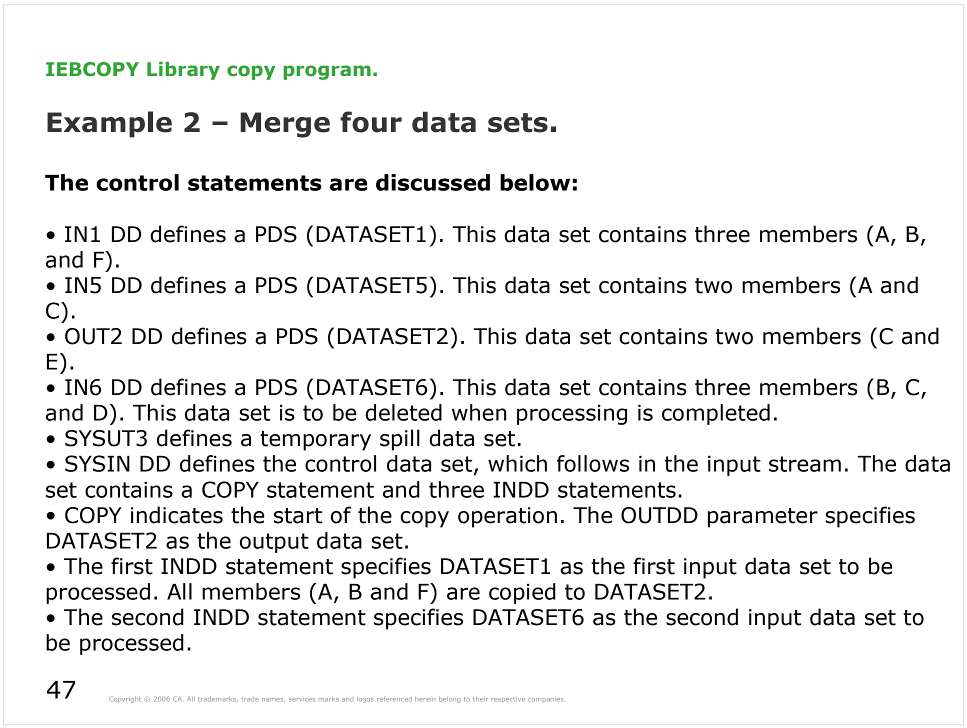
Example 2 – Merge four data sets.

In this example, members are copied from three input PDSs (DATASET1, DATASET5, and DATASET6) to an existing output PDS (DATASET2). The sequence in which the control statements occur controls the manner and sequence in which PDSs are processed.

```
//COPY       JOB ...
//JOBSTEP    EXEC PGM=IEBCOPY
//SYSPRINT   DD SYSOUT=A
//IN1        DD DSN=DATASET1,UNIT=disk,VOL=SER=111112,DISP=SHR
//IN5        DD DSN=DATASET5,UNIT=disk,VOL=SER=111114,DISP=OLD
//IN6        DD DSN=DATASET6,UNIT=disk,VOL=SER=111117,DISP=(OLD,DELETE)
//OUT2       DD DSN=DATASET2,UNIT=disk,VOL=SER=111115,DISP=(OLD,KEEP)
//SYSUT3     DD UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN      DD *
  COPYOPER   COPY INDD=IN1
                  INDD=IN5
                  INDD=IN6
                  OUTDD=OUT2
/*
```

46

COPYOPER is a label only.


IEBCOPY Library copy program.

Example 2 – Merge four data sets.

The control statements are discussed below:

• IN1 DD defines a PDS (DATASET1). This data set contains three members (A, B, and F).
• IN5 DD defines a PDS (DATASET5). This data set contains two members (A and C).
• OUT2 DD defines a PDS (DATASET2). This data set contains two members (C and E).
• IN6 DD defines a PDS (DATASET6). This data set contains three members (B, C, and D). This data set is to be deleted when processing is completed.
• SYSUT3 defines a temporary spill data set.
• SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement and three INDD statements.
• COPY indicates the start of the copy operation. The OUTDD parameter specifies DATASET2 as the output data set.
• The first INDD statement specifies DATASET1 as the first input data set to be processed. All members (A, B and F) are copied to DATASET2.
• The second INDD statement specifies DATASET6 as the second input data set to be processed.

47

Processing occurs as follows:

1. Since replacement is not specified, members B and C, which already exist in DATASET2, are not copied to DATASET2.

2. Member D is copied to DATASET2.

3. All members in DATASET6 are lost when the data set is deleted. The third INDD statement specifies DATASET5 as the third input data set to be processed. No members are copied to DATASET2 because all exist in DATASET2.

Example 3 – Unload and compress a data set.

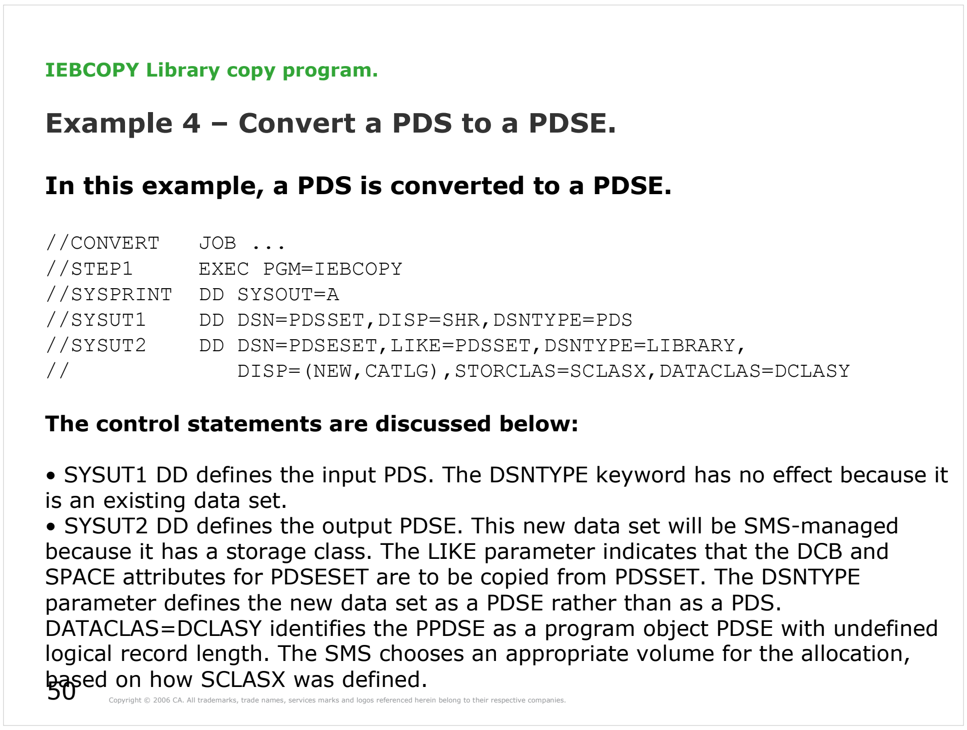
In this example, a PDS is unloaded to a tape volume to create a backup copy of the data set. If this step is successful, the PDS is to be compressed in place.

```
//SAVE       JOB ...
//STEP1      EXEC PGM=IEBCOPY
//SYSPRINT   DD SYSOUT=A
//SYSUT1     DD DSN=PARTPDS,UNIT=disk,VOL=SER=PCP001,DISP=OLD
//SYSUT2     DD DSN=SAVDATA,UNIT=tape,VOL=SER=TAPE03,DISP=(NEW,KEEP),
//              LABEL=(,SL)
//SYSUT3     DD DSN=TEMP1,UNIT=disk,VOL=SER=111111,DISP=(NEW,DELETE),
//              SPACE=(80,(60,45))
//SYSIN      DD DUMMY
//STEP2      EXEC PGM=IEBCOPY,COND=(0,NE),PARM='SIZE=500K'
//SYSPRINT   DD SYSOUT=A
//COMPDS     DD DSN=PARTPDS,UNIT=disk,DISP=OLD,VOL=SER=PCP001
//SYSUT3     DD DSN=TEMPA,UNIT=disk,VOL=SER=111111,DISP=(NEW,DELETE),
//              SPACE=(80,(60,45))
//SYSIN      DD *
  COPY OUTDD=COMPDS,INDD=COMPDS
/*
```

Example 3 – Unload and compress a data set.

The control statements are discussed below:

• SYSUT1 DD defines a PDS (PARTPDS) that resides on a disk volume and is assumed to have 700 members.The number of members is used to calculate the space allocation on SYSUT3.
• SYSUT2 DD defines a sequential data set to hold PARTPDS in unloaded form. This data set must be NEW.
• SYSUT3 DD defines the temporary spill data set.
• SYSIN DD defines the control data set. Because SYSIN is dummied and SYSUT2 defines a sequential data set, all members of the SYSUT1 data set will be unloaded to the SYSUT2 data set.
• The second EXEC statement marks the beginning of the compress-in-place operation.
• COMPDS DD defines a PDS (PARTPDS).
• SYSUT3 DD defines the temporary spill data set to be used if there is not enough space in main storage for the input data set's directory entries.
• SYSIN DD defines the control data set, which follows in the input stream.
• COPY marks the beginning of the copy operation. Because the same DD statement is specified for both the INDD and OUTDD operands, the data set is compressed in place.



IEBCOPY Library copy program.

Example 4 – Convert a PDS to a PDSE.

In this example, a PDS is converted to a PDSE.

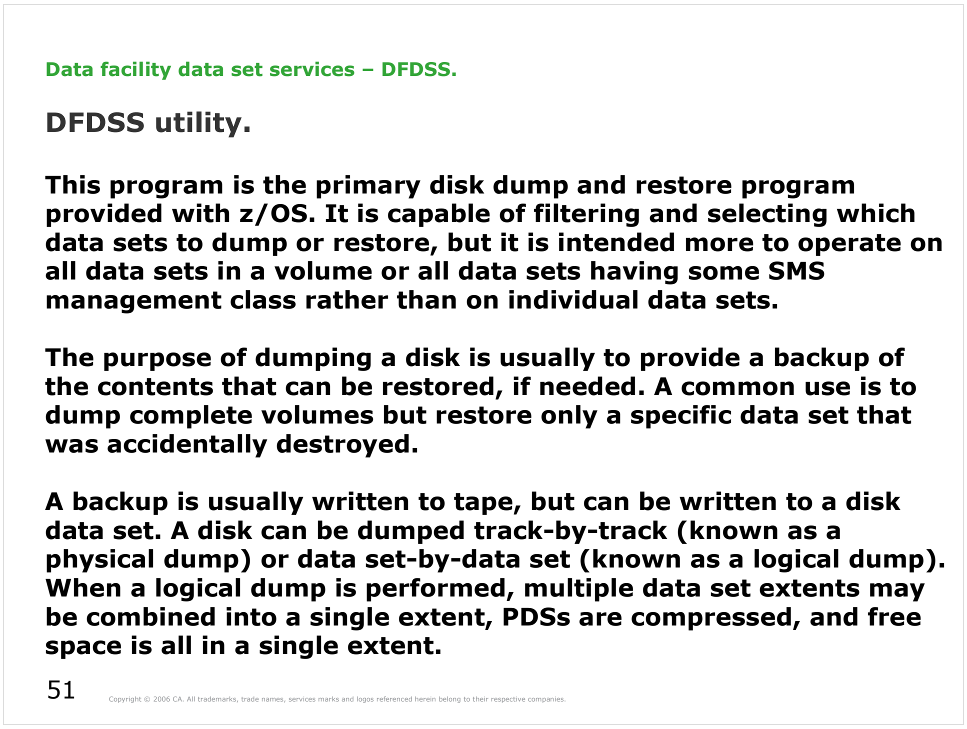
```
//CONVERT   JOB ...
//STEP1     EXEC PGM=IEBCOPY
//SYSPRINT  DD SYSOUT=A
//SYSUT1    DD DSN=PDSSET,DISP=SHR,DSNTYPE=PDS
//SYSUT2    DD DSN=PDSESET,LIKE=PDSSET,DSNTYPE=LIBRARY,
//             DISP=(NEW,CATLG),STORCLAS=SCLASX,DATACLAS=DCLASY
```

The control statements are discussed below:

• SYSUT1 DD defines the input PDS. The DSNTYPE keyword has no effect because it is an existing data set.
• SYSUT2 DD defines the output PDSE. This new data set will be SMS-managed because it has a storage class. The LIKE parameter indicates that the DCB and SPACE attributes for PDSESET are to be copied from PDSSET. The DSNTYPE parameter defines the new data set as a PDSE rather than as a PDS. DATACLAS=DCLASY identifies the PPDSE as a program object PDSE with undefined logical record length. The SMS chooses an appropriate volume for the allocation, based on how SCLASX was defined.

50

Since the DDnames “SYSUT1” and “SYSUT2” are used to define the input and output data sets, no SYSIN data set is required.



Data facility data set services – DFDSS.

DFDSS utility.

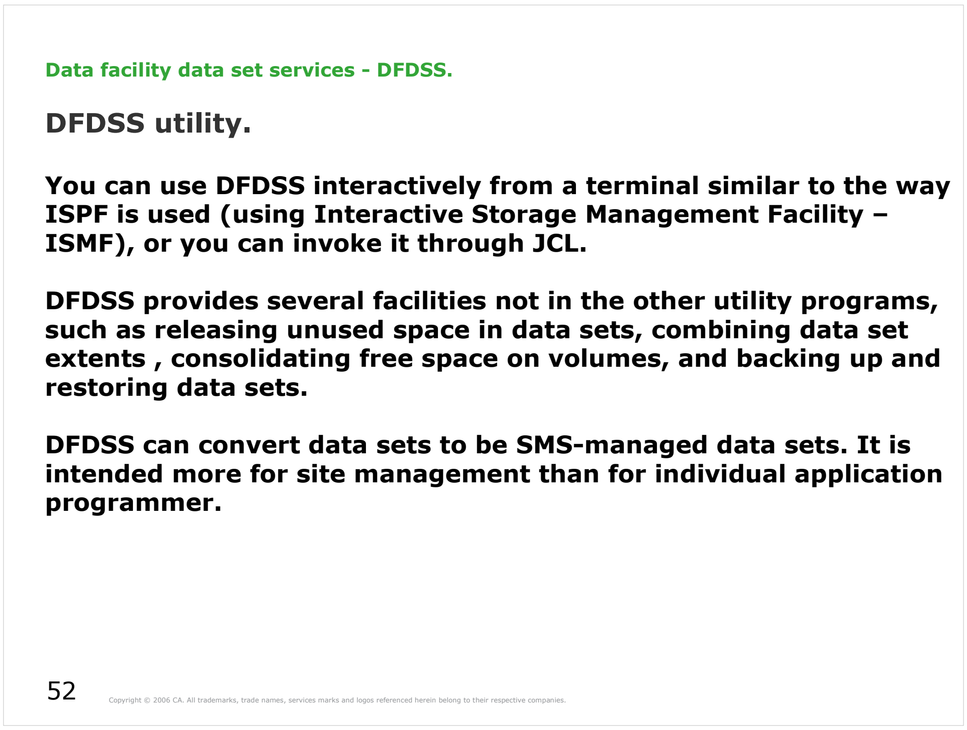
This program is the primary disk dump and restore program provided with z/OS. It is capable of filtering and selecting which data sets to dump or restore, but it is intended more to operate on all data sets in a volume or all data sets having some SMS management class rather than on individual data sets.

The purpose of dumping a disk is usually to provide a backup of the contents that can be restored, if needed. A common use is to dump complete volumes but restore only a specific data set that was accidentally destroyed.

A backup is usually written to tape, but can be written to a disk data set. A disk can be dumped track-by-track (known as a physical dump) or data set-by-data set (known as a logical dump). When a logical dump is performed, multiple data set extents may be combined into a single extent, PDSs are compressed, and free space is all in a single extent.

51

See „DFSMSdss.pdf“, „z/OS DFSMS Storage Administration Reference (for DFSMSdfp, DFSMSdss, DFSMShsm)“ and „z/OSDFSMSdss Storage Administration Guide“ for more information.



Data facility data set services - DFDSS.

DFDSS utility.

You can use DFDSS interactively from a terminal similar to the way ISPF is used (using Interactive Storage Management Facility – ISMF), or you can invoke it through JCL.

DFDSS provides several facilities not in the other utility programs, such as releasing unused space in data sets, combining data set extents , consolidating free space on volumes, and backing up and restoring data sets.

DFDSS can convert data sets to be SMS-managed data sets. It is intended more for site management than for individual application programmer.

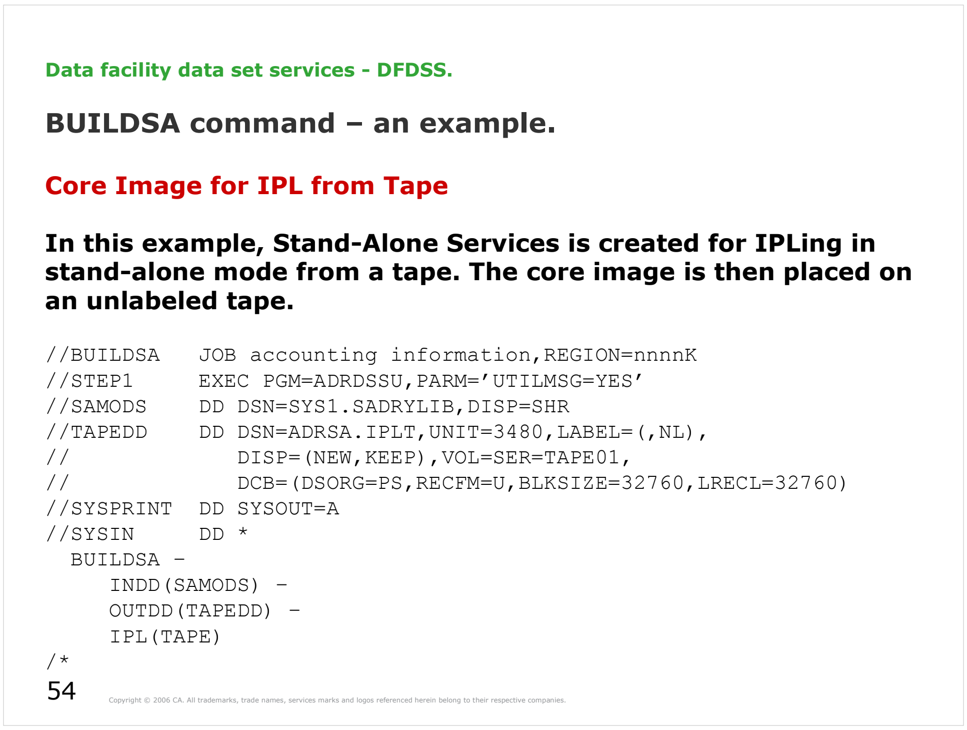
52

Data facility data set services - DFDSS.

BUILDSA command.

The BUILDSA function builds the IPL-able core image under the current operating system.

Use the BUILDSA command to build the IPL-able core image for the Stand-Alone Services program. You can specify the device (card reader, tape drive, or DASD volume) from which Stand-Alone Services will be IPLed.



BUILDSA command – an example.

Core Image for IPL from Tape

In this example, Stand-Alone Services is created for IPLing in stand-alone mode from a tape. The core image is then placed on an unlabeled tape.

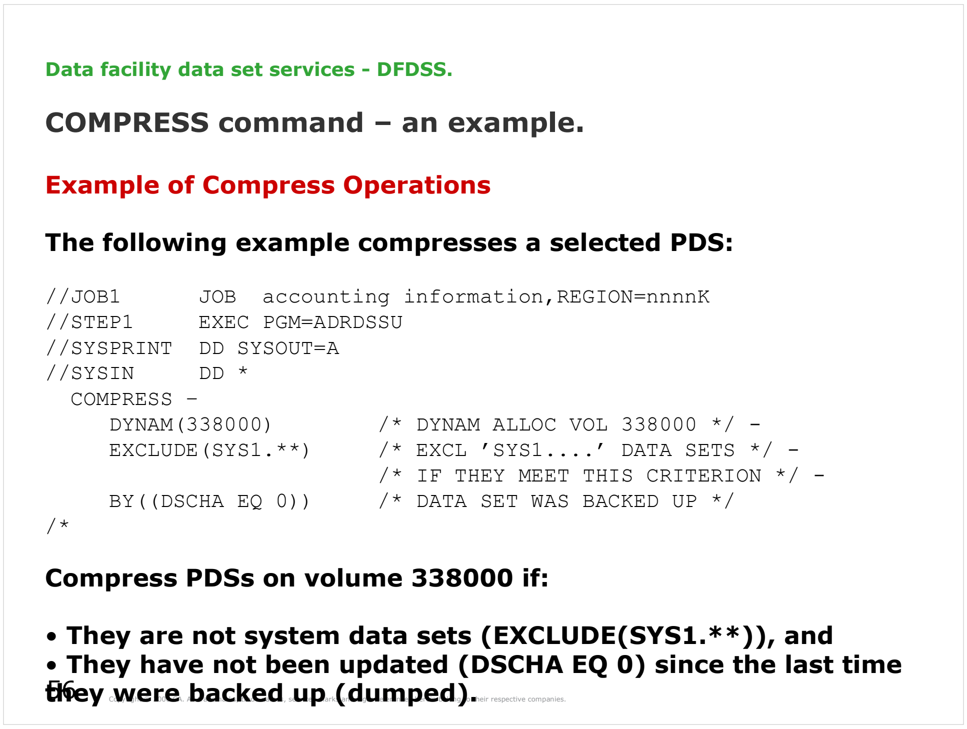
```
//BUILDSA   JOB accounting information,REGION=nnnnK
//STEP1     EXEC PGM=ADRDSSU,PARM='UTILMSG=YES'
//SAMODS    DD DSN=SYS1.SADRYLIB,DISP=SHR
//TAPEDD    DD DSN=ADRSA.IPLT,UNIT=3480,LABEL=(,NL),
//             DISP=(NEW,KEEP),VOL=SER=TAPE01,
//             DCB=(DSORG=PS,RECFM=U,BLKSIZE=32760,LRECL=32760)
//SYSPRINT  DD SYSOUT=A
//SYSIN     DD *
  BUILDSA -
     INDD(SAMODS) -
     OUTDD(TAPEDD) -
     IPL(TAPE)
/*
```

Note: The DCB parameters must be coded as shown.

COMPRESS command.

The COMPRESS command compresses PDSs on a specified volume. Compressing (degassing) removes unused space between members in a PDS. This command is useful for compressing system PDSs before you apply maintenance (to avoid certain space-related abends).

Restriction: You must not compress data sets that contain DFSMSdss or IEBCOPY executable code.



Data facility data set services - DFDSS.

COMPRESS command – an example.

Example of Compress Operations

The following example compresses a selected PDS:

```
//JOB1      JOB  accounting information,REGION=nnnnK
//STEP1     EXEC PGM=ADRDSSU
//SYSPRINT  DD SYSOUT=A
//SYSIN     DD *
  COMPRESS -
     DYNAM(338000)        /* DYNAM ALLOC VOL 338000 */ -
     EXCLUDE(SYS1.**)     /* EXCL 'SYS1....' DATA SETS */ -
                          /* IF THEY MEET THIS CRITERION */ -
     BY((DSCHA EQ 0))     /* DATA SET WAS BACKED UP */
/*
```

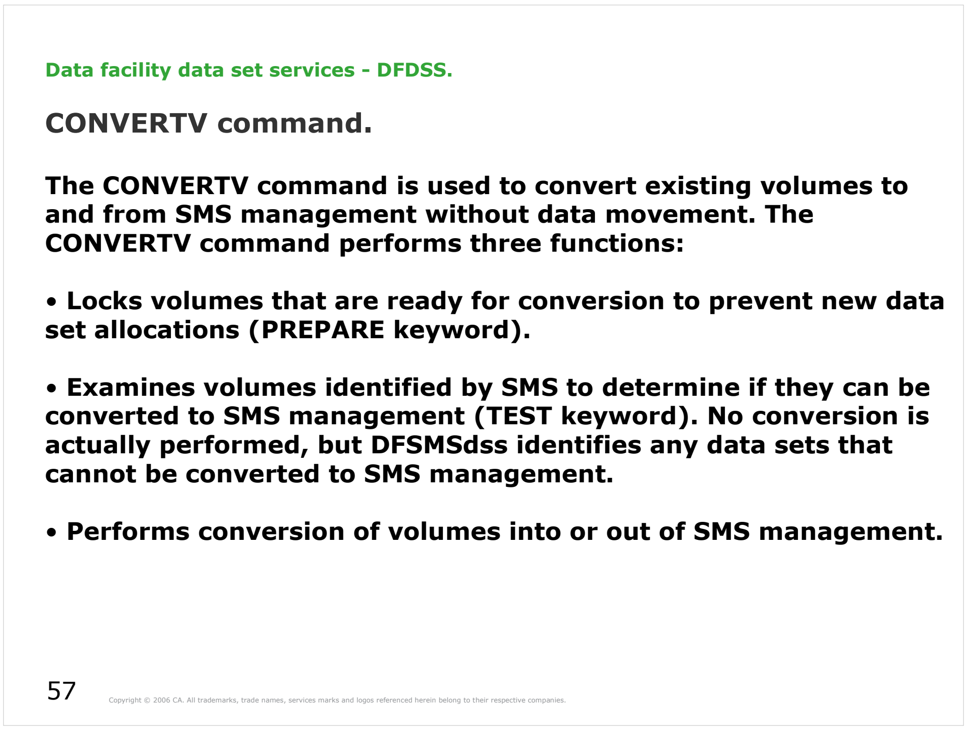
Compress PDSs on volume 338000 if:

- **They are not system data sets (EXCLUDE(SYS1.**)), and**
- **They have not been updated (DSCHA EQ 0) since the last time they were backed up (dumped).**

56

Note: It seems that the example specifies what I want to EXCLUDE, but it does not specify what I want to INCLUDE – the INCLUDE keyword is missing. In fact, in that case, the COMPRESS command compresses all PDSs located on the 338000 volume – the information is in the VTOC.

See „Filtering by Data Set Characteristics“ in „DFSMSdss Storage Administration Reference“ for DSCHA parameter description.



Data facility data set services - DFDSS.

CONVERTV command.

The CONVERTV command is used to convert existing volumes to and from SMS management without data movement. The CONVERTV command performs three functions:

- **Locks volumes that are ready for conversion to prevent new data set allocations (PREPARE keyword).**

- **Examines volumes identified by SMS to determine if they can be converted to SMS management (TEST keyword). No conversion is actually performed, but DFSMSdss identifies any data sets that cannot be converted to SMS management.**

- **Performs conversion of volumes into or out of SMS management.**

57

Guideline: Proper RACF security authorization might be required.

Data facility data set services - DFDSS.

CONVERTV command – an example.

Using the CONVERTV Command to Simulate Conversion

```
//JOB1      JOB  accounting information,REGION=nnnnK
//STEP1     EXEC PGM=ADRDSSU
//SYSPRINT  DD SYSOUT=A
//SYSIN     DD *
  CONVERTV SMS -
     DYNAM((VOL001,3380),(VOL002,3380),(VOL003)) -
     TEST
```

The preceding example uses the TEST keyword to simulate conversion. The TEST keyword produces a report that indicates whether the three volumes (VOL001, VOL002, and VOL003) can be converted to SMS management.

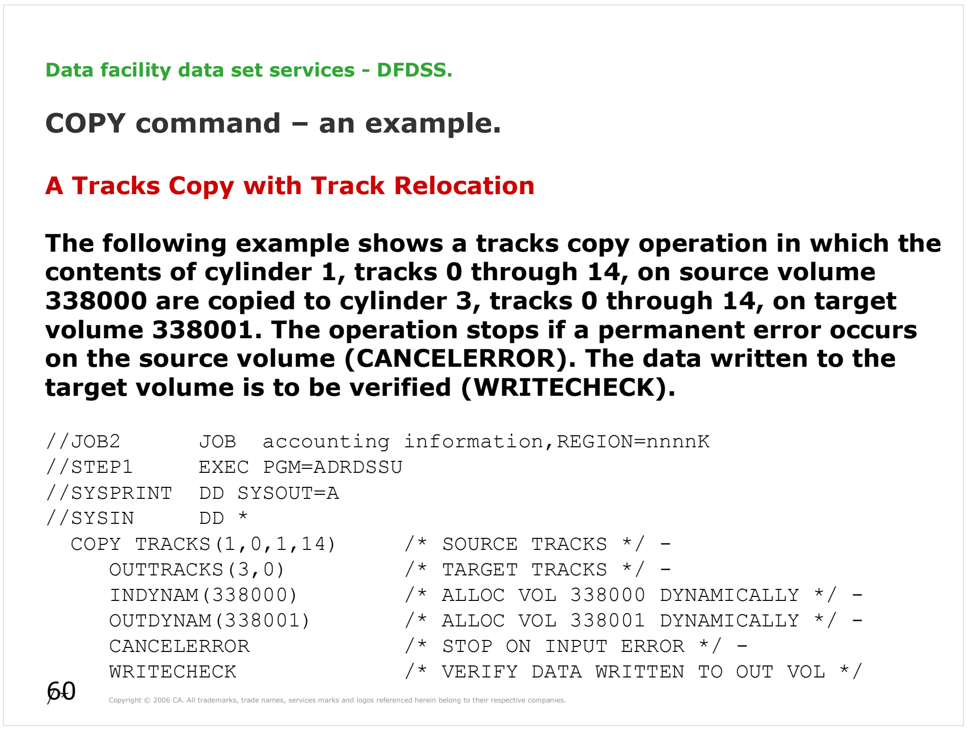
COPY command.

The DFSMSdss COPY command performs data set movement, volume movement, and track movement from one DASD volume to another. You can copy data sets to another volume of either like or unlike device types. Like devices have the same track capacity (3390 Model 2 and 3390 Model 3), while unlike devices have different track capacities (3380 Model K and 3390 Model 3).

DFSMSdss offers two ways to process COPY commands as follows:

• *Logical processing* is data set-oriented, which means that it operates against data sets and volumes independently of physical device format.

• *Physical processing* operates against volumes and tracks, but moves data at the track-image level.



Data facility data set services - DFDSS.

COPY command – an example.

A Tracks Copy with Track Relocation

The following example shows a tracks copy operation in which the contents of cylinder 1, tracks 0 through 14, on source volume 338000 are copied to cylinder 3, tracks 0 through 14, on target volume 338001. The operation stops if a permanent error occurs on the source volume (CANCELERROR). The data written to the target volume is to be verified (WRITECHECK).

```
//JOB2      JOB  accounting information,REGION=nnnnK
//STEP1     EXEC PGM=ADRDSSU
//SYSPRINT  DD SYSOUT=A
//SYSIN     DD *
  COPY TRACKS(1,0,1,14)     /* SOURCE TRACKS */ -
     OUTTRACKS(3,0)         /* TARGET TRACKS */ -
     INDYNAM(338000)        /* ALLOC VOL 338000 DYNAMICALLY */ -
     OUTDYNAM(338001)       /* ALLOC VOL 338001 DYNAMICALLY */ -
     CANCELERROR            /* STOP ON INPUT ERROR */ -
     WRITECHECK             /* VERIFY DATA WRITTEN TO OUT VOL */
```

60

See MCOE.EDU.JCL.JCL(DSSCPYDS) for logical processing example.

Data facility data set services - DFDSS.

DEFRAG command.

When you issue the DEFRAG command, DFSMSdss relocates data set extents on a DASD volume to reduce or eliminate free space fragmentation. A summary report is printed that lists the before and after statistics of the volume.

Attention: Canceling the DEFRAG command is strongly discouraged. Canceling an in-process DEFRAG can damage data in numerous and unpredictable ways.

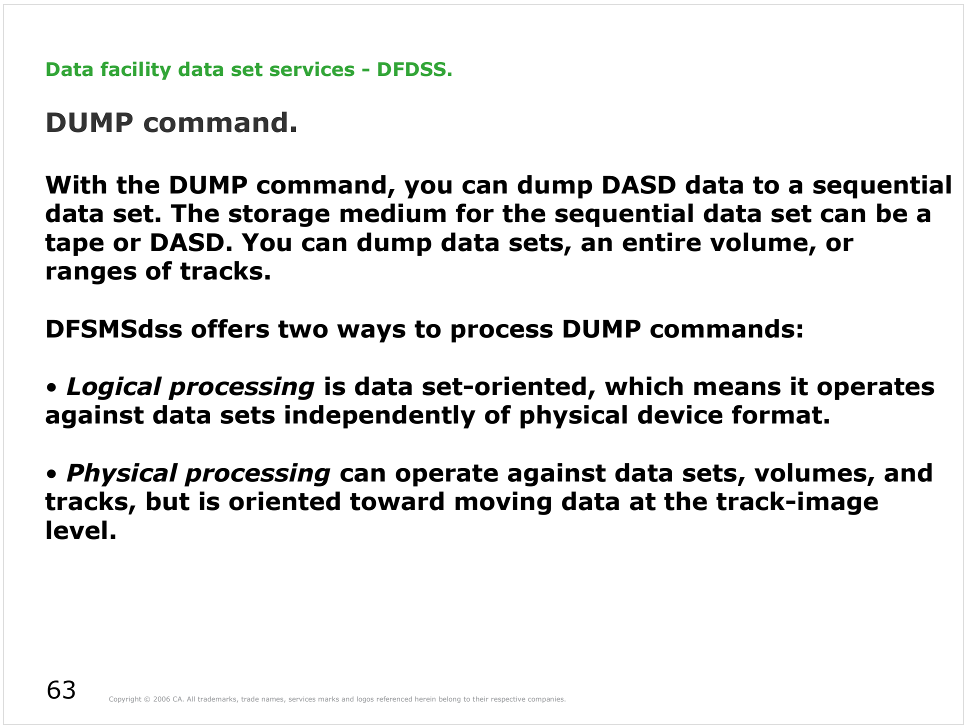
Data facility data set services - DFDSS.

DEFRAG command – an example.

A DEFRAG Operation with Excluded Data Sets

```
//JOB1       JOB  accounting information,REGION=nnnnK
//STEP1      EXEC PGM=ADRDSSU
//SYSPRINT   DD SYSOUT=A
//DASD       DD UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//A1         DD DSN=USER2.EXCLUDE,DISP=SHR
//SYSIN      DD *
  DEFRAG DDNAME(DASD) -
     EXCLUDE(LIST(USER2.**.LIST,*.LOAD))
/*
```

In the example, DASD volume 111111 is defragmented. All data sets whose first and last qualifiers are USER2 and LIST, respectively, are to be excluded from this operation, as are data sets with two qualifiers whose second qualifier is LOAD.



Data facility data set services - DFDSS.

DUMP command.

With the DUMP command, you can dump DASD data to a sequential data set. The storage medium for the sequential data set can be a tape or DASD. You can dump data sets, an entire volume, or ranges of tracks.

DFSMSdss offers two ways to process DUMP commands:

- ***Logical processing*** **is data set-oriented, which means it operates against data sets independently of physical device format.**

- ***Physical processing*** **can operate against data sets, volumes, and tracks, but is oriented toward moving data at the track-image level.**

63

See also RESTORE command.

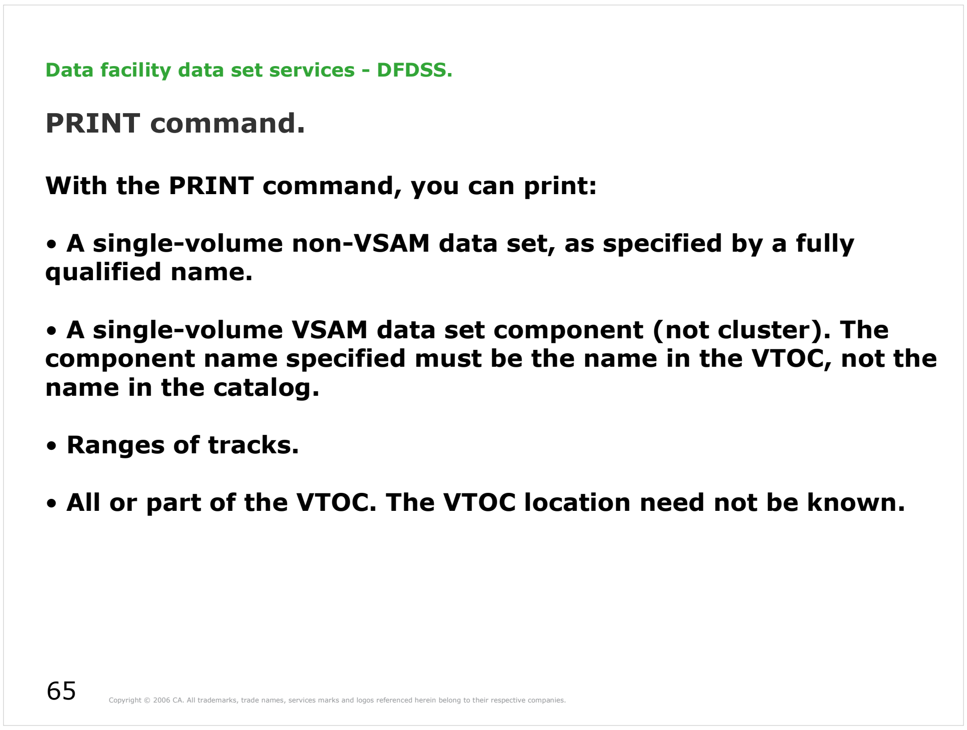
Data facility data set services - DFDSS.

DUMP command – an example.

Dumping a User Catalog and its Aliases

To dump a user catalog, you perform a logical data set dump with the fully qualified user catalog name as the data set name. If the user catalog has any aliases, the aliases are automatically dumped.

```
//JOB2      JOB  accounting information,REGION=nnnnK
//STEP1     EXEC PGM=ADRDSSU
//SYSPRINT  DD SYSOUT=A
//DASD1     DD UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//TAPE      DD UNIT=3480,VOL=SER=TAPE02,
//             LABEL=(1,SL),DISP=(NEW,CATLG),DSNAME=USER2.BACKUP
//SYSIN     DD *
  DUMP    OUTDDNAME(TAPE) -
  DATASET(INCLUDE(MY.USER.CAT))
/*
```



Data facility data set services - DFDSS.

PRINT command.

With the PRINT command, you can print:

- **A single-volume non-VSAM data set, as specified by a fully qualified name.**

- **A single-volume VSAM data set component (not cluster). The component name specified must be the name in the VTOC, not the name in the catalog.**

- **Ranges of tracks.**

- **All or part of the VTOC. The VTOC location need not be known.**

65

Data facility data set services - DFDSS.

PRINT command – an example.

Printing a Component of a VSAM Cluster

```
//JOB3      JOB  accounting information,REGION=nnnnK
//STEP1     EXEC PGM=ADRDSSU
//SYSPRINT  DD SYSOUT=A
//SYSIN     DD *
  PRINT INDYNAM(338000)           /* ALLOC VOL 338000 DYNAMICALLY */ -
     DATASET(PARTS.VSAM1.INDEX)   /* DATA SET THAT HAS BAD TRACK */ -
     WAIT(0,0)                    /* DO NOT WAIT IF ENQ FAILS */ -
     TOL(ENQF)                    /* IGNORE ENQ FAILURES */ -
     PSWD(PARTS.VSAM1/USERPSWD)   /* PASSWORD FOR CLUSTER */
/*
```

RELEASE command.

The RELEASE command releases allocated but unused space from all eligible sequential, partitioned, and extended-format VSAM data sets. DFSMSdss offers two ways to process RELEASE commands:

- **Logical processing operates on a single selected data set at a time.**

- **Physical processing operates on all selected data sets that reside on a single volume.**

RELEASE command – an example.

Example of a Release Operation

The following is an example of a release operation on selected sequential and PDSs:

```
//JOB1       JOB  accounting information,REGION=nnnnK
//STEP1      EXEC PGM=ADRDSSU
//SYSPRINT   DD SYSOUT=A
//SYSIN      DD *
  RELEASE INCLUDE(**)  -
     DYNAM(338000)      /* DYNAM ALLOC VOL 338000 */ -
     MINTRKS(10)        /* THERE ARE 10 OR MORE UNUSED TRKS */ -
                        /* MINSEC NOT SPEC. IT DEFAULTS TO 1 */
/*
```

Unused tracks of sequential and PDSs on volume 338000 are to be released if both:
• The number of unused tracks in the data set is greater than or equal to 10.
• The data set can be extended later if required (MINSEC(1)). This need not be specified, because it is the default.

RESTORE command.

With the RESTORE command, you can restore data to DASD volumes from DFSMSdss-produced dump volumes. You can restore data sets, an entire volume, or ranges of tracks. You can restore to unlike devices from a logical dump tape.

DFSMSdss offers two ways to process RESTORE commands:

• *Logical processing* is data set-oriented, which means it operates against data sets independently of physical device format.

• *Physical processing* can operate against data sets, volumes, and tracks, but is oriented toward moving data at the track-image level.

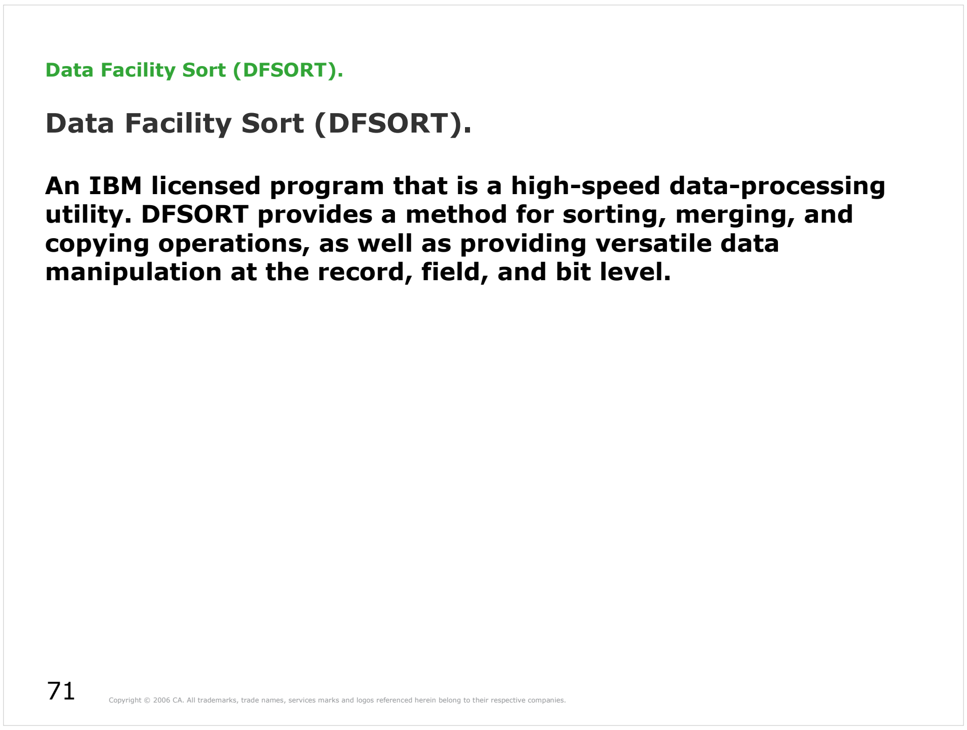
Data facility data set services - DFDSS.

RESTORE command – an example.

Tracks Restore Operation

Example shows that DASD volume numbered 111111 will be restored from the first data set of standard label tape volumes called TAPE01 and TAPE02. The command input for a tracks restore operation are shown below.

```
//JOB1       JOB  accounting information,REGION=nnnnK
//STEP1      EXEC PGM=ADRDSSU
//SYSPRINT   DD SYSOUT=A
//TAPE       DD UNIT=3480,VOL=SER=(TAPE01,TAPE02),
//              LABEL=(1,SL),DISP=(OLD,KEEP),DSNAME=USER2.BACKUP
//DASD       DD UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//SYSIN      DD *
  RESTORE TRACKS(1,0,1,5) INDDNAME(TAPE) -
     OUTDDNAME(DASD) PURGE
/*
```



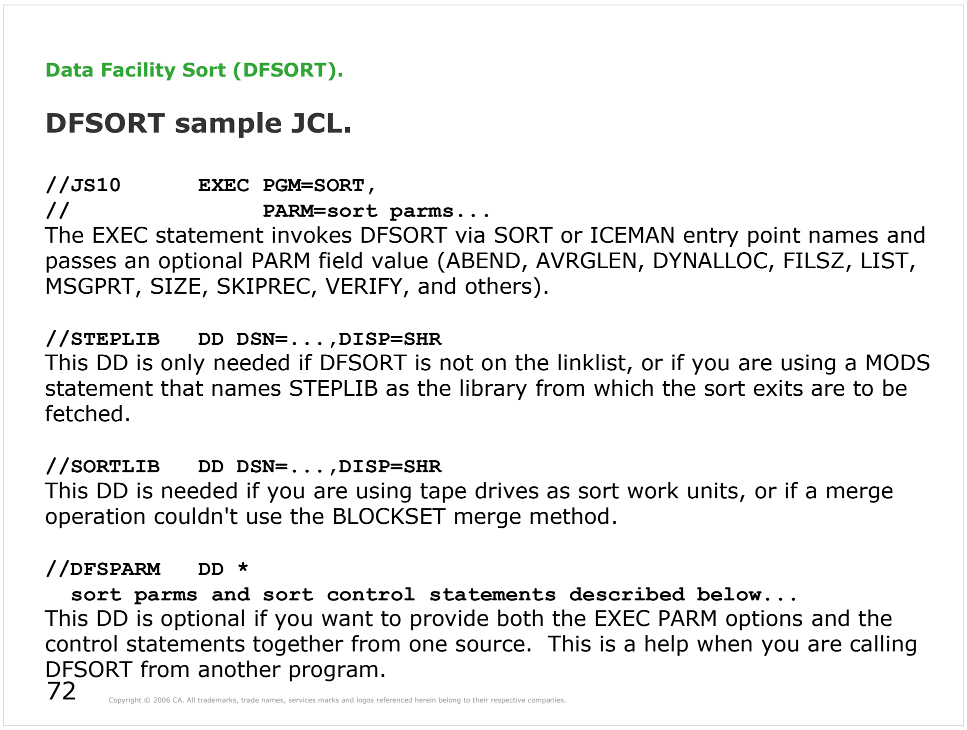
Data Facility Sort (DFSORT).

Data Facility Sort (DFSORT).

An IBM licensed program that is a high-speed data-processing utility. DFSORT provides a method for sorting, merging, and copying operations, as well as providing versatile data manipulation at the record, field, and bit level.

71

See „Utilities.pdf“ and „z/OS DFSORT Application Programming Guide“ for more information.



Data Facility Sort (DFSORT).

DFSORT sample JCL.

```
//JS10      EXEC PGM=SORT,
//               PARM=sort parms...
```

The EXEC statement invokes DFSORT via SORT or ICEMAN entry point names and passes an optional PARM field value (ABEND, AVRGLEN, DYNALLOC, FILSZ, LIST, MSGPRT, SIZE, SKIPREC, VERIFY, and others).

```
//STEPLIB   DD DSN=...,DISP=SHR
```

This DD is only needed if DFSORT is not on the linklist, or if you are using a MODS statement that names STEPLIB as the library from which the sort exits are to be fetched.

```
//SORTLIB   DD DSN=...,DISP=SHR
```

This DD is needed if you are using tape drives as sort work units, or if a merge operation couldn't use the BLOCKSET merge method.

```
//DFSPARM   DD *
  sort parms and sort control statements described below...
```

This DD is optional if you want to provide both the EXEC PARM options and the control statements together from one source. This is a help when you are calling DFSORT from another program.

72

See MCOE.EDU.JCL.JCL(DFSORT).

DFSORT sample JCL - continuation.

```
//SYMNAMES  DD DSN=...,DISP=SHR
```
Defines the SYMNAMES data set containing statements to be used for symbol processing. Required only if symbol processing is to be performed.

```
//SYMNOUT   DD DSN=...,DISP=SHR
```
Defines the data set in which SYMNAMES statements and the symbol table are to be listed. Optional if SYMNAMES DD is specified. Otherwise ignored.

```
//SYSOUT    DD SYSOUT=*
```
This DD statement is required because DFSORT messages are written to this file.

```
//SORTIN    DD DSN=input..DSN.to.sort,
//             DISP=SHR
```
This required DD statement points to the input for a sort operation. The input file can be a sequential file or PDS member, or a VSAM data set. Concatenated non-VSAM files are supported if they all have the same RECFM with the largest block size first in the concatenation.

DFSORT sample JCL - continuation.

```
//SORTINnn  DD DSN=input..DSN.to.sort,
//             DISP=SHR
```

Identifies one of the pre-sorted input files for a merge operation. 'nn' is a number in the range 01 through 99. SORTINnn files can't be concatenated. All SORTINnn files have to have the same LRECL if fixed-length records are used.

```
//SORTOUT   DD DSN=sorted.output.dsn,
//             DISP=SHR
```

Identifies the output file for both sort and merge operations. This can be a sequential file, a PDS member, or a VSAM file.

```
//SORTWKnn  DD UNIT=SYSDA,DISP=SHR,
//             SPACE=(TRK,(30,30),RLSE)
```

Defines sort work files, where 'nn' is a suffix in the range 01 through 32. SORTWKnn files aren't needed if sorting a small data set whose records will all fit in storage at once, or if you told DFSORT to obtain its sort work files dynamically via the DYNALLOC parameter on an OPTION or SORT control statement.

DFSORT sample JCL - continuation.

```
//outfil    DD UNIT=SYSDA,DISP=SHR,
//             SPACE=(TRK,(30,30),RLSE)
```

The OUTFIL DD statements describe the characteristics of the data sets in which the processed (sorted, merged, copied) records are to be placed and indicate their location.
Although the ddname SORTOUT can actually be used for an OUTFIL data set, the term "SORTOUT" will be used to denote the single non-OUTFIL output data set. Simply, you can specify different DDname if you do not want to use SORTOUT.

```
//SYSIN     DD *
  sort control statements described below...
```

The control statements for DFSORT are input from this file. If DFSORT is dynamically invoked from a PL/1 or COBOL program, the SORTCNTL DD is used to override the control statements passed by the invoking program.

```
//SORTCNTL  DD DSN=...,DISP=SHR
  sort control statements described below...
```

Defines the data set from which additional or changed DFSORT control statements can be read when DFSORT is program-invoked. It is the same as SYSIN DD but control statement are saved in data set.

DFSORT sample JCL - continuation.

```
//SORTCKPT  DD DSN=CHECKPT,UNIT=TAPE,
//             DISP=(,KEEP),VOL=SER=ABC123
```
This DD statement allocates a sequential file to be used as the checkpoint data set when sort checkpointing is requested via the CKPT parameter on an OPTION or SORT control statement.

```
//SORTSNAP  DD SYSOUT=*
```
SNAP dump output.

```
//SYSUDUMP  DD SYSOUT=*
```
User dump output.

```
//SYSMDUMP  DD SYSOUT=*
```
Machine dump output.

```
//SYSABEND  DD SYSOUT=*
```
Abend dump output.

DFSORT sample JCL - continuation.

```
//SORTDIAG  DD DUMMY
```

The presence of this DD statement makes DFSORT write all messages (including special diagnostics) to the SYSOUT file. The SORTDIAG DD should be used with CAUTION since its usage substantially degrades DFSORT performance.

DFSORT control statements – SORT.

The SORT control statement is used to initiate a sort and describe which field(s) in each record are to be used as sort fields. The format of the SORT control statement is:

```
SORT FIELDS=(pos,length,format,sequence,...) |
     FIELDS=(pos,length,sequence,...),FORMAT=format |
     FIELDS={(} COPY {)} |
     {,CKPT}
     {,DYNALLOC{=(d | ,n | d,n | OFF)}}
     {,EQUALS | ,NOEQUALS}
     {,FILSZ={n | En | Un} }
     {,SIZE={n | En | Un} }
     {,SKIPREC=n}
     {,STOPAFT=n}
     {,Y2PAST={s | f}
```

DFSORT control statements – RECORD.

The RECORD statement tells DFSORT the record format and lengths of the records in the input file. You only need a RECORD statement if the INPUT is from a VSAM file. The format of the RECORD control statement is:

```
RECORD TYPE={F | D},LENGTH=(len1{,len2,len3}) |
       TYPE=V,LENGTH=(len1{,len2,len3,len4,len5,len6,len7})
```

DFSORT control statements – MERGE.

The MERGE control statement requests a merge operation. The MERGE control statement is similar to the SORT control statement, but a few SORT parameters are not needed. The format of the MERGE statement is:

```
MERGE FIELDS=(pos,length,format,sequence,...) |
      FIELDS=(pos,length,sequence,...),FORMAT=format |
      FIELDS={(} COPY {)}
      {,EQUALS | ,NOEQUALS}
      {,FILES=n}
      {,FILSZ=n | ,SIZE=n}
      {,SKIPREC=n}
      {,STOPAFT=n}
      {,Y2PAST={s | f}
```

DFSORT control statements – END.

The END statement tells DFSORT to stop reading the input control statement data set; it marks a logical 'end-of-file' on the control statement input. You can insert an END statement to make DFSORT ignore control statements after the END statement.

DFSORT control statements – OPTIONS.

The OPTION control statement is used to set or override various DFSORT execution options. Some of the options that you can set on the OPTION control statement can also be set via the JCL PARM= field or on a SORT or MERGE control statement. The format of the OPTION statement is:

```
OPTION {ARESALL={ n | nK | nM} }
       {,ARESINV={n | nK | nM}
       {,AVGRLEN=n}
       ...
       ...
       many others
       ...
       ...
```

DFSORT control statements – INCLUDE.

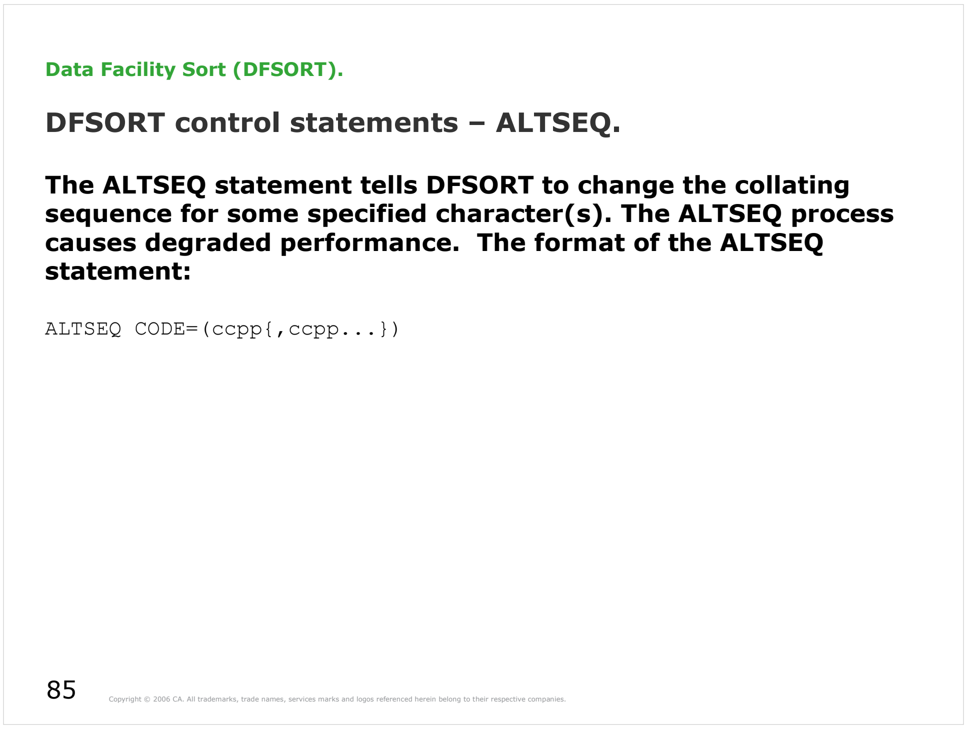
The INCLUDE control statement is used to establish selection criteria for the records to be included in the output data set. You can include a record by comparing the contents of its fields to a constant or to another field in the record. The format of the INCLUDE statement is:

```
INCLUDE COND=({expression,{{AND | OR}|{ALL | NONE}},expression},...)
{,FORMAT=x}
```

DFSORT control statements – OMIT.

The OMIT control statement is used to establish selection criteria for the records to be omitted from the output data set. You can omit a record by comparing the contents of its fields to a constant or to another field in the record. The format of the OMIT statement is:

```
OMIT COND=({expression,{{AND | OR}|{ALL | NONE}},expression},...)
{,FORMAT=x}
```



Data Facility Sort (DFSORT).

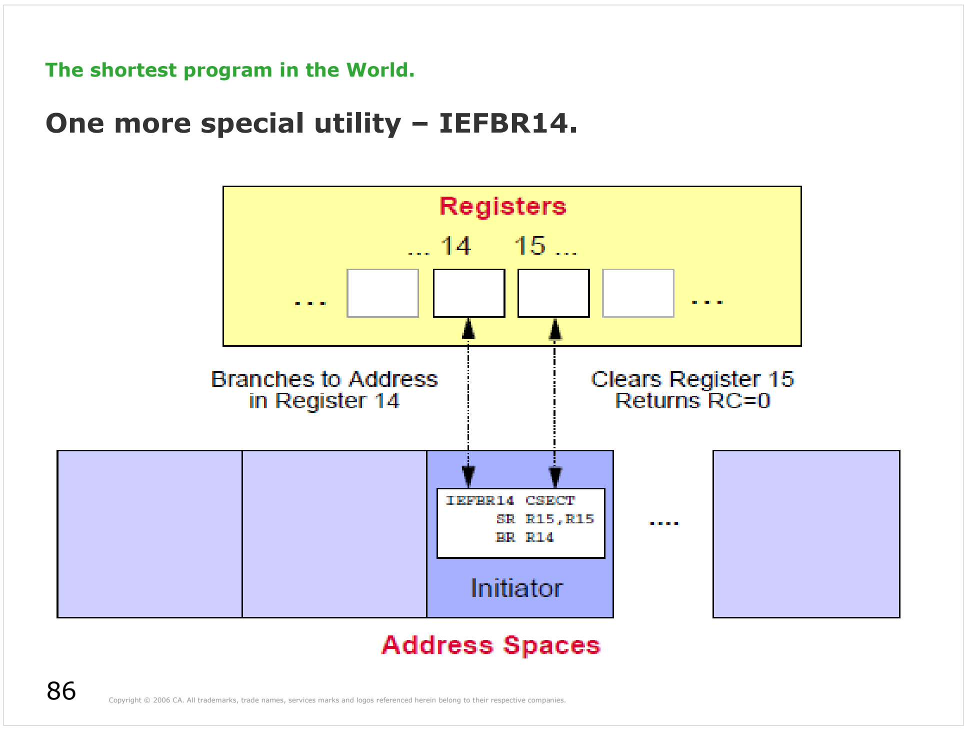
DFSORT control statements – ALTSEQ.

The ALTSEQ statement tells DFSORT to change the collating sequence for some specified character(s). The ALTSEQ process causes degraded performance. The format of the ALTSEQ statement:

```
ALTSEQ CODE=(ccpp{,ccpp...})
```

85

See MCOE.EDU.JCL.JCL(DFSALTQ).


The shortest program in the World.

One more special utility – IEFBR14.

Registers
... 14 15 ...
...
...
Branches to Address in Register 14
Clears Register 15 Returns RC=0
....
Initiator
Address Spaces

```
IEFBR14  CSECT
         SR  R15,R15
         BR  R14
```

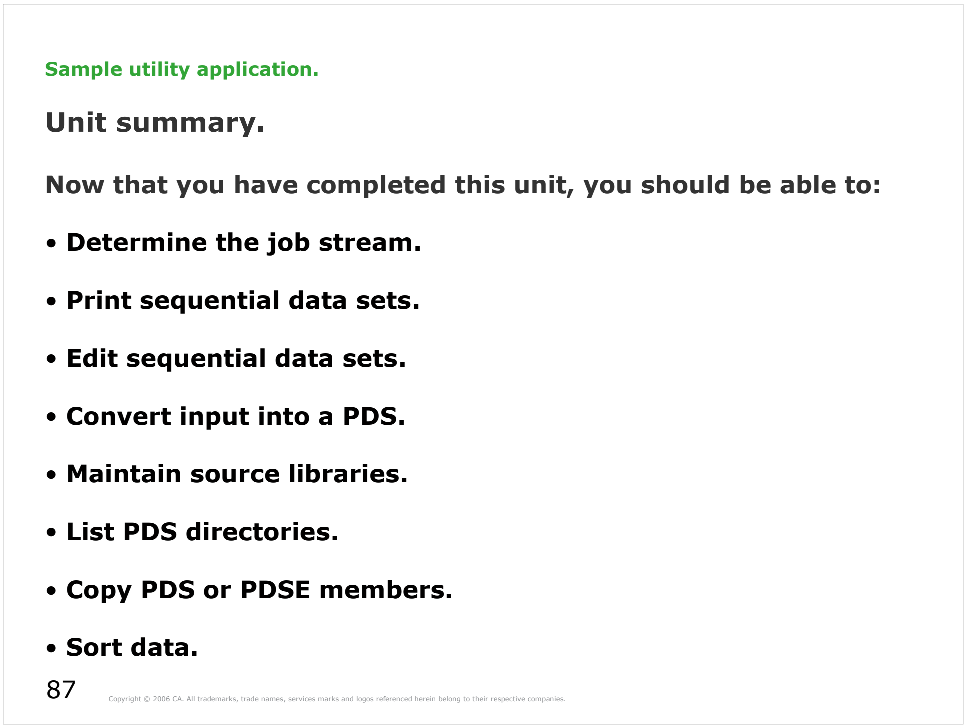
86

The only function of this program is to provide a zero (0) completion code. It is used as a safe vehicle to "execute JCL.“ For example, consider the following job:

```
//DELPDS1  EXEC PGM=IEFBR14
//DD3      DD  DISP=(MOD,DELETE,DELETE),DSN=KOTMI01.TEST.PDS,

//         SPACE=(TRK,(0,0,0))
//CREPDS1  EXEC PGM=IEFBR14
//DD4      DD  DISP=(NEW,CATLG,DELETE),
//         DSN=KOTMI01.TEST.PDS,SPACE=(TRK,(5,2,10)),
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=32720)
```

This is a useful job although the program that is executed (IEFBR14) does nothing. While preparing to run the job, the initiator deletes KOTMI01.TEST.PDS in step DELPDS1 and allocates KOTMI01.TEST.PDS in step CREPDS1 and keeps the data set when the job ends. The DD names DD3 and DD4 have no meaning and are used because the syntax of a DD statement requires a DD name.

IEFBR14 is not a utility, in the sense that it is not included in the Utilities manual.



Sample utility application.

Unit summary.

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

- **Determine the job stream.**
- **Print sequential data sets.**
- **Edit sequential data sets.**
- **Convert input into a PDS.**
- **Maintain source libraries.**
- **List PDS directories.**
- **Copy PDS or PDSE members.**
- **Sort data.**

87