Chapter
2

USING EXISTING DATASETS

TM

**SYS-ED/
Computer
Education
Techniques, Inc.**

Objectives

You will learn:

- Coding DD statement parameters for existing datasets.
- Coding statements for tape datasets.
- Concatenating input datasets.
- Managing tape datasets.
- Tape operation concepts.
- Catalog maintenance.
- IEHLIST utility program - listing the content of a catalog.
- Utility control statements.
- LISTCTLG control statement - generate a catalog listing.
- IEHPROGM utility program - adding and deleting catalog entries.

1 Disk and Tape Datasets - Existing

DD statement parameters are used with existing disk and tape datasets. Additional parameters specific to utilization with tape datasets also can be used.

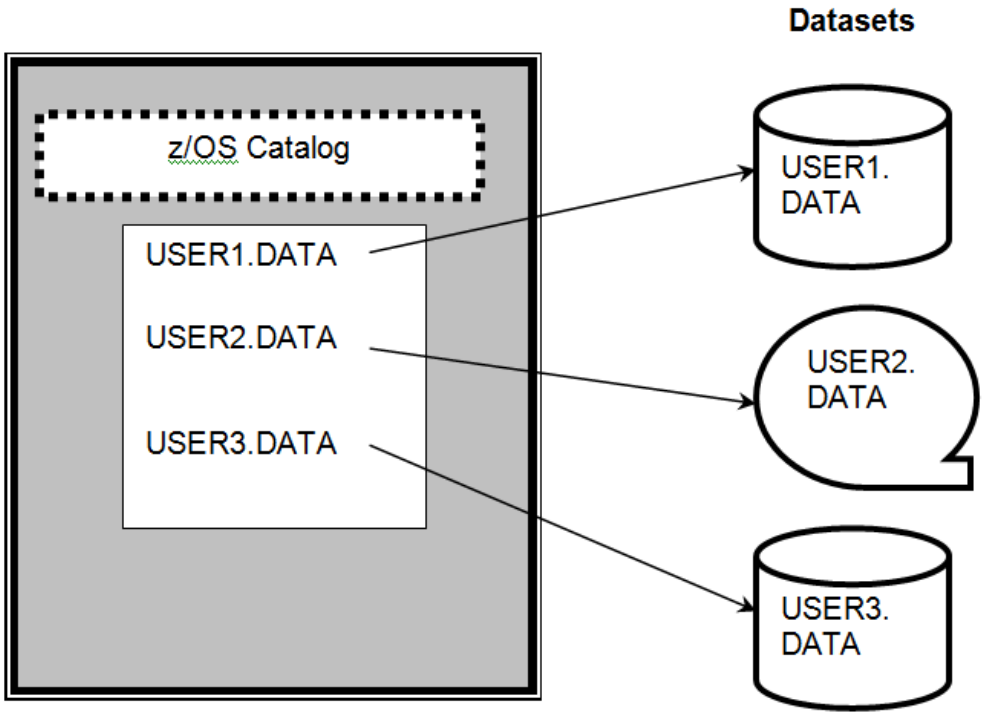
The z/OS operating system uses volume control; the catalog provides assistance in locating existing datasets and maintaining a catalog.

A catalog stores dataset attributes and the volumes on which a dataset resides. When a dataset is cataloged, it can be referred to by name without the programmer needing to specify where the dataset is stored. Datasets can be catalogued or uncatalogued.

In order to find an uncatalogued dataset that has been requested, z/OS requires the following information:

dataset name	volume name	unit = volume device type

Catalog Structure



2 DD Statement Parameters - Basic

Before a program can use an existing dataset, the z/OS operating system must be provided with the name and location of the dataset.

The DD parameters provide this necessary information:

Parameter	Explanation
DSNAME	The name of the dataset to be used.
DISP	The disposition of the dataset.
UNIT	The type of storage device on which the dataset resides.
VOL	The specific volume which contains the dataset.

2.1 DSNAME or DSN Parameter

The DSNAME or DSN parameter specifies the name of a dataset.

It has the following format:

```
DSN=name
DSNAME=name
```

The name is used by the operating system to identify the dataset. Each dataset on a volume must have a unique name. Datasets on different volumes can have the same name; however, this can be confusing and is a naming practice which should be avoided.

A simple dataset name has no more than 8 characters and follows the standard JCL naming rules:

Example:

```
DSN=PAYFILE
```

A qualified dataset name consists of two or more simple dataset names joined by a period.

There can be a maximum of 44 characters including the periods in the dataset name.

Example:

These are both qualified dataset names.

```
DSNAME=SYSED.INPUT.DATA

DSN=THE.MAXIMUM.SIZE.DATA.SET.NAME.OF44.CHARACTRS
```

Most installations have rules governing the naming of datasets. Typically, simple dataset names are prohibited and there will be a requirement to have a high level qualifier. Consult the installation standards manual for the site specific rules.

2.2 DISP Parameter

The DISP or disposition parameter performs two functions.

1. It describes the status of a dataset at allocation.

The status informs the z/OS operating system whether:

- the dataset is new or old.
- other jobs can share the dataset.
- a write over or add to should be performed on an existing data.

2. The DISP instructs the z/OS operating system how to dispose of the dataset during termination.

The disposition portion of the DISP parameter indicates whether to:

- save or delete the dataset.
- change catalog information about the dataset.

Two subparameters specify disposition:

- One controls the dataset's disposition when the program ends normally.
- The other controls disposition in the event of an abend.

The subparameters of DISP are positional, with this format:

```
DISP=(status,[normal],[abnormal])
```

Status Subparameter

Status is the first subparameter.

The status options of the dataset at allocation are.

Option	Explanation
NEW	The dataset does not exist and should be created.
OLD	An existing dataset should be used. Do not share it with other jobs. Write over any data which already resides in the dataset.
SHR	Use an existing disk dataset. Share it with other any other jobs that want it. Write over any existing data.
MOD	Use an existing dataset and do not share it with other jobs. Add new data at the end of the dataset instead of writing over existing data.

Examples:

These two examples demonstrate the status of the subparameter:

```
DSN=INDATA,DISP=SHR

DSN=WRITOVER,DISP=OLD
```

For existing datasets, status is the only subparameter required.

- If only status is coded, the parentheses are not necessary.
- When multiple parameters are specified, parenthesis is required.

Normal Disposition

The second positional subparameter instructs the z/OS operating system what to do with the dataset when the program completes normally.

There are five options to select from:

Option	Explanation
DELETE	Scratch the dataset from the disk. Depending on other parameters, z/OS may uncatalog the dataset.
KEEP	Save the dataset.
PASS	Save the dataset and pass it to the next step.
CATLG	Save the dataset and put the dataset's name in the catalog.
UNCATLG	Save the dataset; but remove the dataset's name from the catalog. A disk file entry remains in the VTOC - Volume Table of Contents which allows access to the data. However, it will be necessary to specify the location of the dataset.

Examples:

These DISP examples will display the dataset's status and normal disposition:

```
DSN=IMPORTANT,DISP=(SHR,KEEP)

DSN=USELESS,DISP=(OLD,DELETE)
```

If this subparameter is not specified, the z/OS operating system returns the dataset to its condition prior to the step.

A new dataset will be deleted; an old dataset will be kept.

Abnormal Disposition

The last DISP subparameter specifies what action the z/OS operating system should take if the program abends. All normal disposition options, except PASS, are permitted:

Option	Explanation
DELETE	Remove the dataset from the disk.
KEEP	Save the dataset.
CATLG	Save the dataset and enter its name in the catalog.
UNCATLG	Save the dataset; remove the dataset's entry from the catalog.

Examples:

This code demonstrates abnormal disposition:

```
DSN=INDATA,DISP=(OLD,DELETE,KEEP)

DSN=OUTDATA,DISP=(OLD,DELETE)
```

The first example is a typical disposition for an input dataset.

- If the step executes, the dataset is no longer needed and can be deleted.
- If the program abends, the dataset must be kept in order that the program can be rerun.

The second example often is used with output datasets.

- If the program completes successfully, the output is saved.
- If the program does not complete successfully, the output is deleted.

The OUTDATA example takes the default for the normal disposition. Since the subparameters are positional, a comma must be coded to indicate its absence.

The default abnormal disposition is to take the action specified for normal disposition. It will be necessary to specify an abnormal disposition subparameter only when there is a requirement to have the system perform a different disposition.

2.3 UNIT Parameter

The UNIT parameter designates to the system which device the dataset resides on.

The UNIT is specified with this format:

UNIT=<unit address>

<device type>

<esoteric name>

Parameter	Explanation
unit address	An address consisting of three hexadecimal characters specifies the channel, control unit, and unit number. Specifying a unit address restricts the operating system and should only be used by systems programmers. **Example:** UNIT=180
device type	This option corresponds to the model number of the device. The system assigns any available device of that type. **Example:** UNIT=3390 is a common disk device. UNIT=3480 is a common tape device.
esoteric name	This name designates a group of devices assigned by the installation. Devices in the group may not be of the same type. When coding an esoteric name, the system assigns any available device from that group. **Example:** UNIT=SYSDA normally refers to all disk drives on the system. Research the esoteric names using the installation's standards manual.

The esoteric name is the preferred option. It removes the need for changing the JCL whenever new I/O devices are installed and enables an installation to improve system performance by grouping devices according to functionality.

2.4 VOLUME or VOL Parameter

The VOLUME or VOL parameter specifies the serial number of the volumes containing the dataset.

The format of the VOLUME parameter is:

VOL=SER=serial

VOLUME=SER=serial

The serial number can be up to six alphameric characters.

Examples:

This code demonstrates the UNIT and VOLUME parameters:

```
DSN=DISKSET,DISP=SHR,UNIT=SYSDA,VOL=SER=SYSED5

DSN=TAPESET,DISP=MOD,UNIT=TAPE,VOL=SER=330218
```

3 Concatenating Datasets

It sometimes will be necessary for a program to treat several input datasets as a single file. Concatenation is a technique used for connecting multiple input datasets into a single dataset.

In order to concatenate datasets, code a DD statement describing each dataset. The first DD statement must specify the ddname required by the programs. The ddname fields on all concatenated statements must be blank.

Example:

In this code, separate datasets for each week of the month are merged into a single MONTH dataset.

```
//MERGE    EXEC PGM=IEBGENER
//SYSUTI   DD DSN=WEEK1,DISP=(OLD,DELETE,KEEP),
//            UNIT=SYSDA,VOL=SER=SYSED8
//         DD DSN=WEEK2,DISP=(OLD,DELETE,KEEP),
//            UNIT=SYSDA,VOL=SER=SYSED5
//         DD DSN=WEEK3,DISP=(OLD,DELETE,KEEP),
//            UNIT=SYSDA,VOL=SER=SYSED8
//         DD DSN=WEEK4,DISP=(OLD,DELETE,KEEP),
//            UNIT=SYSDA,VOL=SER=SYSED3
//SYSUT2   DD DSN=MONTH,DISP=OLD,
//            UNIT=SYSDA,VOL=SER=SYSED0
```

When the program begins reading the file, the z/OS operating system retrieves the first record from the first dataset, WEEK1. When it reaches the last record in WEEK1, the z/OS operating system does not indicate the end-of-file to the program. Instead, z/OS continues without interruption using WEEK2's records. At the end of the last concatenated dataset, WEEK4, z/OS notifies the program that the end-of-file has been reached.

4 Tape Datasets

The DSN, DISP, UNIT, and VOL parameters are applicable to both disk and tape datasets. They will handle the majority of tapes that will be encountered.

In the following situations, the parameters will be insufficient:

- Tapes which contain multiple datasets.
- Datasets which fill multiple volumes.
- Tapes with non standard labels.

These three situations can be addressed by using the LABEL parameter and subparameters of UNIT and VOLUME.

4.1 Tapes with Multiple Datasets

A single tape can hold a significant amount of data. Normally, tape is used for large datasets which do not need the rapid access of disk. Each tape typically contains one dataset. For example, one tape can contain Jan fund dataset, Feb fund dataset, and March fund dataset.

However, tape is sometimes used for small datasets. This would occur in a situation in which it is necessary to back up important datasets or copy datasets for another installation. One reel of tape can be used for holding many small datasets.

LABEL Parameter

On a tape containing multiple datasets, the LABEL parameter designates the sequence number of the specified dataset:

```
LABEL=sequence
```

Example:

```
//INPUT  DD    DSN=SYSED.TAPEDS,
//             DISP=OLD,
//             UNIT=TAPE,
//             VOL=SER=552800,
//             LABEL=3
```

This code has the dataset named SYSED.TAPEDS as the third dataset on tape 522800.

There is a limitation with a tape when using multiple datasets on a volume. During a step, the system can only access one dataset on the tape. This means that a program can not read from two different datasets on one tape. Nor can it read one dataset and write another on the same tape.

4.2 Multiple-volume Datasets

Some very large datasets may not be able to fit on one tape. Instead they span over two or more volumes. JCL provides several options for processing multiple-volume datasets.

Multiple Serial Numbers

On the VOL parameter, specify the serial numbers of all volumes containing the dataset.

```
VOL=SER=(serial,serial,serial...)
```

When more than one volume is needed, enclose the serial numbers in parentheses. They must be listed in sequence within the dataset.

Example:

The dataset MANYTAPE fills all of two and part of a third tape.

```
//DATAFILE  DD  DSN=MANYTAPE,
//          DISP=OLD,
//          UNIT=TAPE,
//          VOL=SER=(330218,230836,532661)
```

The program starts reading DATAFILE with volume 330218, continues with 230836, and ends on 532661. As the program finishes each volume, the system calls for the next one.

Starting Sequence Number

There may be a requirement not to start at the beginning of the dataset. This can be done by providing the z/OS operating system with the sequence number of the volume from which to start the processing.

The starting sequence number is another VOL subparameter.

```
VOL=(,,start,SER=(serial,serial,serial...))
```

The starting sequence number is the third positional subparameter of VOL.

- If the keyword SER is the only subparameter needed, then these positional subparameters can be ignored.
- If the other sub parameters are needed, all the subparameters must be enclosed in parentheses.

As with other positional parameters, absent subparameters must be indicated with a comma.

Example:

This code starts processing with the second volume of the dataset MANYTAPE:

```
//DATAFILE  DD  DSN=MANYTAPE,
//          DISP=OLD,
//          UNIT=TAPE,
//          VOL=(,,2,SER=(330218,230836,532661))
```

When the program begins, it starts reading data on volume 230836. Although the first volume, 330218, will be skipped, its serial number still must be included in the SER subparameter.

4.3 Label Formats - Processing

Tape labels contain information about the dataset such as the name, creation date, number of blocks, sequence number, blocksize, and whether the dataset spans multiple volumes.

The z/OS operating system checks the label during allocation and termination to ensure that the dataset has been fully processed.

This label information may be written in one of several formats. The LABEL parameter designates which format is to be used.

```
LABEL=(sequence,format)
```

Label formats can be one of the following options:

Options	Explanation
SL	Standard Labels are the default for tapes produced by IBM OS type systems.
AL	ANSI Labels are the format used by most non-IBM systems.
NL	No Labels indicates that no label information is present. Typically, NL is used when a company, such as a software vendor, distributes tapes to many different systems.
NSL	Nonstandard Labels are labels which do not conform to any standard format. The installation must provide a special program to process these labels.
LTM	A Leading Tape Mark precedes the label information. This format is standard for IBM DOS type systems.
BLP	Bypass Label Processing. This will ignore existing labels. For most applications, standard labels should be used for all tapes. Tapes that come from another location should either use BLP or AL. When using something other than SL, consult the installation standards in order to determine the label format.

Example:

Label format is the second positional subparameter of the LABEL format.

```
//TAPE1  DD    DSN=ANSITAPE,
//             DISP=OLD,
//             UNIT=TAPE,
//             VOL=SER=000222,
//             LABEL=(,AL)

//TAPE2  DD    DSN=NONLABEL,
//             DISP=OLD,
//             UNIT=TAPE,
//             VOL=SER=000000,
//             LABEL=(3,NL)
```

The first example uses the LABEL parameter for processing the first dataset on a tape with an ANSI label.

The second example designates the third dataset on a tape that does not have labels.

The vast majority of tapes at a installation will have the default standard labels. The other options will be important when it is necessary to process tapes created at another installation.

5 Tape Operations - Improving

The original design of MVS Job Control Language required computer operator intervention when a job needed a tape dataset.

The operator would have to:

1. Find the requested tape in the library.

2. Go to the tape drive and remove the tape used by a previous job.

3. Check the serial number and mount the tape.

During this time the program would wait.

A modern z/OS installation will have an automated tape system such as SILO. The SILO will automatically find and MOUNT the tape.

The z/OS operating system provides two UNIT parameter options for reducing the time a job must wait for tape mounts.

The format of the UNIT parameter is:

UNIT=(device,[count],[DEFER])

Option	Explanation
device	Indicates the unit address, device type, or esoteric name requested for the dataset.
count	Specifies the number of units to be allocated for this dataset.
DEFER	Instructs z/OS not to mount the tape until it is opened.

5.1 Requesting Multiple Units

The count subparameter allows multiple units to be dedicated to a dataset. This permits a program to process a multiple-volume dataset without delays for tape mounts.

Example:

```
//DATAFILE   DD DSN=MANYTAPE,
//           DISP=OLD,
//           UNIT=(TAPE,2),
//           VOL=SER(330218,230836,532661)
```

This subparameter instructs the z/OS operating system to allocate two devices from the TAPE group. During allocation, z/OS mounts the first two volumes: 330218 and 230836. When the program finishes the first tape, it continues processing the second tape without interruption. While the second tape is being processed, the operator replaces the first volume with the third volume, 532661. Accordingly, the program can continue without waiting for tape mounts.

The count subparameter can decrease the amount of time a large job is in the system. However, it can also create many operational problems, especially if tape drives are scarce.

Do not allocate multiple devices without the approval of the operations manager.

5.2 Deferred Mounting

The final UNIT subparameter instructs the z/OS operating system not to mount the tape during allocation. Instead, the tape will be mounted when the program opens the file.

Example:

```
//LOG   DD     DSN=ERRORLOG,
//            DISP=MOD,
//            UNIT=(TAPE,,DEFER),
//            VOL=SER=717484
```

Deferred tape mounts are useful when the program does not OPEN the file. A program uses a tape to log errors detected by the program. Most of the time the program runs without detecting errors; therefore the tape is not needed. When the program detects an error, it must wait for a tape mount before it can continue processing.

For the deferred mounts to be effective, a program must OPEN a file only when it is actually used. The standard technique of opening all files at the beginning of the program, defeats any benefits gained by deferring the mount.

6 Catalogs

Catalogs are hierarchically organized by dataset name qualifiers. All datasets are alphabetically arranged based on the first or high level qualifier. Within that qualifier, datasets are alphabetized on the second qualifier.

Duplicate entries are not permitted. Datasets in a catalog must be uniquely named.

For each dataset, the device type and the volume serial number(s) are listed.

Example:

A portion of a catalog is provided.

SYSED.BUDGET.FY84	DISK	SED007
SYSED.BUDGET.FY85	DISK	SED010
SYSED.PAYROLL.EMPMAST	DISK	SED006
SYSED.PAYROLL.CURTRAN	DISK	SED007
SYSED.PAYROLL.YTDHIST	TAPE	PAY084
	TAPE	PAY117
SYS.1.LINKLIB	DISK	SYSRES
TEST.PAYDATA	TAPE	231005
TEST.SYSED.PAYROLL.EMPMAST	DISK	SED011
TEST.SYSED.PAYROLL.CURTRAN	DISK	SED011

SYSED.PAYROLL.YTDHIST, is a multi-volume dataset; both serial numbers are listed.

6.1 Finding Datasets with a Catalog

Only the name - DSN and disposition - DISP are needed in order to locate a catalogued dataset which is listed in a catalog. During allocation, the z/OS operating system searches the catalog in order to supply the missing parameters.

Example:

```
//PAYTRAN DD DSN=TEST.PAYDATA,DISP=OLD
```

When z/OS allocates PAYTRAN, it looks for an existing dataset named TEST.PAYDATA. Since the JCL does not specify the device or serial number, the z/OS operating system searches the catalog for an entry named TEST.PAYDATA. It finds an entry, allocates a TAPE unit, and instructs the system to mount tape 231005.

If a UNIT or VOL parameter is included in the JCL, it overrides the catalog entry.

Example:

```
//PAYTRAN   DD  DSN=TEST.PAYDATA,
//            DISP=OLD,
//            UNIT=SYSDA,
//            VOL=SER=SED011
```

During allocation, the z/OS operating system finds the UNIT and VOL parameters in the JCL; therefore it does not look in the catalog. Instead, it locates TEST.PAYDATA using the parameters specified on the DD statement, SYSDA volume SED011.

6.2 Catalog Maintenance: DISP Parameter Options

Catalog entries are added and removed using the CATLG, UNCATLG, and DELETE options of the DISP parameter. Catalog changes, like the other disposition options, are made during termination processing.

CATLG

The CATLG option creates a dataset entry in a catalog based on information in the JCL.

Example:

```
//EMPINFO   DD DSN=TEST.PAYDATA.RUN4,
//            DISP=(OLD,CATLG),
//            UNIT=DISK,
//            VOL=SER=SED012
```

During termination, the z/OS operating system creates a catalog entry for TEST.PAYDATA.RUN4. The catalog will show the device type DISK and serial number SED012. The next time that the dataset is used, the UNIT and VOL options can be omitted.

If the catalog already contains an entry for TEST.PAYDATA.RUN4, z/OS includes an error message as part of the termination messages. However, the catalog will not be changed. Future references use the dataset to which the catalog originally pointed, not the more recent intended reference. Accordingly, it will be important to review the termination messages in order to ensure that the datasets are properly cataloged.

UNCATLG

The UNCATLG disposition removes an entry from the catalog without deleting the dataset itself.

These two examples show UNCATLG in use:

```
//PAYTRAN   DD  DSN=TEST.SYSED.PAYROLL,
//          DISP=(OLD,UNCATLG),
//          UNIT=DISK,
//          VOL=SER=SED011
//PAYDATA   DD  DSN=TEST.SYSED.PAYDATA,
//          DISP=(OLD,UNCATLG)
```

The z/OS operating system allocates TEST.SYSED.PAYROLL using the JCL parameters which have been provided. During termination, z/OS removes the dataset entry from the catalog. If there is no entry, the termination messages will indicate the error.

With a DISP of UNCATLG, only the catalog entry is removed. The dataset itself is unchanged, it remains on disk SEDO11. Since the catalog entry is gone, future references to TEST. SYSED.PAYDATA must be fully specified by the JCL.

TEST.PAYDATA is allocated using its catalog entry. During termination, the catalog entry is deleted. All future references to TEST.SYSED.PAYDATA must be fully specified by the JCL.

DELETE

The DELETE option scratches the dataset.

- If z/OS located the dataset through JCL, there is no change to the catalog.
- If the dataset was allocated using a catalog, DELETE also removes the catalog entry.

Examples:

These examples use DELETE instead of UNCATLG:

```
//PAYTRAN     DD DSN=TEST.SYSED.PAYROLL,
//            DISP=(OLD,DELETE),
//            UNIT=DISK,
//            VOL=SER=SED011
//PAYDATA     DD DSN=TEST.SYSED.PAYDATA,
//            DISP=(OLD,DELETE)
```

The first DD statement deletes TEST.SYSED.PAYROLL without changing the catalog.

The second DD statement deletes TEST.SYSED.PAYDATA and removes its entry from the catalog.

6.3 IEHLIST Utility Program: Listing the Catalog

There may be a requirement to know what entries are contained in the catalog. When the z/OS operating system provides an error message as it attempts to CATLG or UNCATLG, it will be necessary to determine whether the catalog contains the correct entry.

The IBM utility IEHLIST prints the content of a catalog. The listing makes it possible to verify the entries for the datasets.

Example:

This JCL will execute IEHLIST:

```
//CATLIST    EXEC PGM=IEHLIST
//SYSPRINT   DD SYSOUT=*
//CATVOL     DD DISP=OLD,
//           UNIT=SYSDA,
//           VOL=SER=SP8403
//SYSIN      DD *
```

The EXEC statement specifies the IEHLIST program. The catalog listing is directed to the SYSPRINT DD statement. The SYSIN DD statement contains control statements which govern IEHLIST's execution.

It will be necessary to include a DD statement which enables the z/OS operating system to allocate the volume containing the catalog.

IEHLIST does not issue an OPEN command using a ddname; accordingly, any ddname can be chosen. Do not specify a DSN, IEHLIST is able to locate the catalog on a particular volume. Ensure that a DISP of OLD or SHR is specified, otherwise, the z/OS operating system will produce an error message.

7 Utility Control Statements

Utility control statements are similar to job control statements. However, they have a more structured format. Utility control statements can have three fields:

```
LABEL  OPERATION  OPERAND
```

Field	Explanation
LABEL	LABEL is an optional field which identifies a particular operation in the input stream. If a label is present, it must begin in column 1. If not, column 1 must be blank.
OPERATION	OPERATION specifies the operation the utility is to perform. It needs to be separated by at least one blank from both the label and operand fields.
OPERAND	OPERAND gives the utility the information it needs to complete the operation. Commas separate multiple operands for the same statement. All utility operands are keyword parameters; therefore, no positional parameters are used.

Utility control statements are continued by:

1. Ending the first part of the statement with a comma in or before column 71.
2. Placing a non-blank character, such as X, in column 72.
3. Starting the next parameter in column 16.

These rules apply to the control statements for all IBM utility programs, except the IDCAMS utility.

7.1 LISTCTLG Control Statement

The LISTCTLG statement directs IEHLIST to generate a catalog listing.

It has the following format:

```
LISTCTLG  VOL=device=serial,[NODE=qualifier]
```

Parameter	Explanation
VOL	A required parameter specifying the catalog's location.
device	The device type containing the catalog. This has the same value as the UNIT parameter in the JCL.
serial	The serial number of the catalog volume. This must be the same as the VOL=SER JCL parameter.
NODE	An optional parameter which limits the size of the listing.
qualifier	One or more high level qualifiers of catalog entries to be listed.

Most catalogs contain hundreds of entries which result in a large printout.

When searching for specific information, always use the NODE option.

Example:

This code produces three catalog listings:

```
//LISTCATS    EXEC PGM=IEHLIST
//SYSPRINT    DD SYSOUT=*
//CATVOL      DD DISP=SHR,
//            UNIT=SYSDA,
//            VOL=SER=SP8403
//ANYNAME     DD DISP=SHR,
//            UNIT=SYSDA,
//            VOL=SER=NSWA02
//SYSIN       DD *
  LISTCTLG VOL=SYSDA=SP8403
  LISTCTLG VOL=SYSDA=NSWA02,NODE=SYSED.TEST
  LISTCTLG VOL=SYSDA=NSWA02,                          X
       NODE=A0391MH
```

The JCL allocates two volumes for this step, SP8403 and NSWA02.

The control statements specify three LISTCTLG operations which will produce these listings:

1. All datasets contained in the catalog on volume SP8403.

2. All datasets named SYSED.TEST.xxx which are entered in the catalog on NSWA02.

3. All A0391MH.xxx dataset entered in NSWA02's catalog.

The third LISTCTLG statement uses an "X" in column 72 to continue the control statement. Apart from readability, it is functionally equivalent to the second statement.

8 IEHPROGM Utility Program: Catalog Maintenance

IEHPROGM is the IBM utility program which allows catalog entries to be added or deleted. When there is a requirement to make several changes at the same time, IEHPROGM is more efficient than using the DISP parameter.

Example:

IEHPROGM JCL Requirements

This JCL is needed to run IEHPROGM:

```
//CATMAINT   EXEC PGM=IEHPROGM
//SYSPRINT   DD SYSOUT=*
//CATVOL     DD DISP=SHR,
//           UNIT=SYSDA,
//           VOL=SER=SED100
//SYSIN      DD *
```

The EXEC statement names IEHPROGM. SYSPRINT receives the listing and SYSIN specifies the control statements.

As with IEHLIST, IEHPROGM must have the z/OS operating system allocate the volume containing the catalog. It must have a DD statement specifying the UNIT and VOL parameters with a DISP of OLD or SHR. Any ddname can be used.

8.1 IEHPROGM Control Statements

The CATLG control statement adds catalog entries.

The UNCATLG control statement deletes catalog entries.

The format of the CATLG and UNCATLG control statement is:

```
CATLG   DSNAME=data.set.name,VOL=device=serial
UNCATLG  DSNAME=data.set.name
```

Parameter	Explanation
DSNAME	The parameter which specifies the dataset name. It cannot be abbreviated DSN.
data.set.name	The fully qualified dataset name as it appears in the catalog.
VOL	A parameter which specifies the device type and serial number(s) of the volume(s) that contain the dataset(s). It is limited to 50 characters.
device	The device type which contains the dataset to be cataloged. It has the same value as the UNIT parameter in the JCL.
serial	The serial number of the volume on which the dataset resides. This must be the same as the VOL=SER JCL parameter.

The CATLG and UNCATLG control statements only change the catalog. They have no affect on the actual datasets. It is possible to create entries for datasets that do not exist or remove entries for datasets which remain intact.

Example:

This code demonstrates the IEHPROGM control statement.

```
//CATMAINT EXEC PGM=IEHPROGM
//SYSPRINT DD SYSOUT=*
//CATVOL    DD DISP=SHR,
//          UNIT=SYSDA,
//          VOL=SER=SED100
//SYSIN      DD *
  CATLG DSNAME=TEST. NEWPAY.DATA2,              X
        VOL=SYSDA=SED011
  UNCATLG DSNAME=TEST.PAYDATA
        CATLG DSNAME=TEST.PAYDATA,              X
        VOL=TAPE=991555
```

There is no control statement for directly changing a catalog entry to point to a different serial number. As shown in the last two statements, it will first be necessary to UNCATLG the current entry, and then create the correct entry with CATLG.

If IEHPROGM is unable to perform the CATLG or UNCATLG as specified, it will include messages describing the error in the SYSPRINT listing. In addition, it provides the step with a return code of 8. This indicates an unsuccessful completion.