MVS Utility Programs - IBM and User Written

January, 2022

Program	Source	Function
AMASPZAP	IBM	Inspect and/or modify a load module or data record stored on disk.
AMBLIST	IBM	Produce a formatted listing of an object or load module, produce a map and cross-reference listing of a nucleus, produce a list of IDRDATA from a CSECT, or produce a map of all modules in the reenterable load module area (the Link Pack area).
CMPRSEQ	IBM	Compare two sequential datasets or members of two partitioned datasets at the logical record level.
COPYMODS	CBT File #229	Copy all files of an input tape onto 1 to 16 output tapes, strip labels from tape files, add new labels to tape files, correct label records, initialize tape volumes.
DSSDUMP	CBT File #860	Create a backup sequential dataset, ADRDSSU format, containing the contents of one or more DASD datasets.
DSSREST	CBT File #860	Restore one or more datasets from a backup set created by ADRDSSU or DSSDUMP.
FIXDSCB	CBT File #566	Modify and repair of dataset control blocks (DSCBs).
ICKDSF	IBM	Prepare an offline DASD volume so that it can be used in an MVS system.
IEBCOMPR	IBM	Compare two sequential datasets or two partitioned datasets at the logical record level to verify a backup copy.
IEBCOPY	IBM	Copy one or more partitioned datasets or to merge partitioned datasets.
IEBGENER	IBM	Copy a sequential dataset, or a member of a partitioned dataset, to a sequential dataset or a member of a partitioned dataset.
IEBISAM	IBM	Copy an indexed sequential (ISAM) dataset from one DASD volume to another, or into a sequential dataset on DASD or tape; create an ISAM dataset from an unloaded dataset, or print an ISAM dataset.
IEBPTPCH	IBM	Print or punch: all/selected members/selected records from a sequential or partitioned dataset, or the directory of a partitioned dataset.
IEBUPDTE	IBM	Incorporate source language modifications into sequential or partitioned datasets.
IEHDASDR	IBM	Prepare DASD volumes for use, dump/restore the entire contents or portions of a DASD volume to a volume or volumes of the same DASD type, or to a tape volume or volumes.
IEHINITT	IBM	Prepare tape volumes for use by writing IBM volume label sets in EBCDIC onto any number of magnetic tapes mounted on one or more tape units.
IEHLIST	IBM	Print entries in a DASD Volume Table Of Contents or entries in the directory of one or more partitioned datasets.

IEHMAP	CBT V129 File#83	Print a formatted listing of a VTOC or a dataset.
IEHMOVE	IBM	Move or copy one or several datasets from one DASD volume to another, or to a tape.
IEHPROGM	IBM	Modify system control data and to maintain datasets at an organizational level.
LISTPDS	CBT File #316	Prints formatted listings of a PDS directory, the contents of processed members or sequential files, as well as (optionally) punching processed members and/or sequential datasets.
MINIUNZ	CBT File #135	Uncompress the contents of datasets created by compression utilities commonly used on PCs and by the companion program MINIZIP.
MINIZIP	CBT File #135	Compress the contents of: 1) one or more sequential datasets or members of partitioned datasets; or 2) all the members of a one or more partitioned datasets, to create a smaller, portable copy of the original data.
OFFLOAD	CBT File #093	Sequentialize a partitioned dataset.
PDSLOAD	CBT File #093	Recreate a partitioned dataset from a sequential dataset that contains control statements interspersed with data records.
PDSMATCH	CBT File #357	Compare two partitioned datasets.
PDSPRINT	CBT File #316	Print or punch members from a partitioned dataset.
PDSPROGM	CBT File #316	Perform maintenance on partitioned datasets.
PDSSCAN	CBT File #684	Scan all members of one or more partitioned datasets, searching for specified strings.
PDSUPDTE	CBT OFL FIle #065	Scan all members of one or more partitioned datasets, searching for specified strings and replacing those found with replacement strings.
PDSUR	CBT File #949	Copy partitioned datasets from DASD to tape (unload) and from tape to DASD (reload) in IEHMOVE format.
RECV370	CBT File #571	Process XMIT files produced by either the TSO/E TRANSMIT command processor, the XMIT370 batch program, or a similar facility.
RESETDS	NaSPA 1986	Reset to empty status partitioned or sequential datasets in preparation for a reload.
REVLMOD	CBT File #134	Reload load modules that have been offloaded with REVIEW's OFFLOAD command.
SORT	IBM	Sort data records from one or more sequential datasets.
SUPERLST	CBT File #134	Print DASD VTOC listing, with optional Partitioned dataset directory information and volume allocation map.
SYSREPRO	CBT File #316	Copy data records from a sequential dataset or member of a partitioned dataset to another sequential dataset or member.
TAPEMAP	CBT File #299	Read a tape volume and report everything it contains, including detailed information about file contents when the creation program is recognized.
TAPESCAN	CBT File #102	Read a tape volume and report an overview of the datasets on a tape, copy files and recover data past the first end of volume indicator.
UNUPDTE	CBT File #093	Sequentialize a partitioned dataset.
UPDTE	CBT File #093	Recreate a partitioned dataset from a sequential dataset that contains ADD control statements interspersed with data

		records.
VTOCLIST	CBT OFL File #343	Print formatted VTOC listing.
XMIT370	CBT File #571	Read partitioned datasets to produce XMIT files equivalent to the TSO/E TRANSMIT command processor or a similar facility.
ZAPDSCB	CBT File #163	Update DSCBs for specified datasets with values supplied on DD statements.
ZTDUMPTP	CBT File #316	Print data records from datasets on tape.
ZZRELINK	CBT File #860	Re-link-edit one or more existing load modules.

Index of Programs With Control Statements and Example JCL

Program Name	Control Statements	Examples (JCL)
AMASPZAP	ABSDUMP | ABSDUMPT DUMP | DUMPT IDRDATA REP VER | VERIFY SETSSI	1) Dump the CSECT from a load module 2) Verify/Replace instruction in a load module 3) Dump a record from a file 4) Inspect/Modify a data record
AMBLIST	LISTIDR LISTLOAD LISTLPA LISTOBJ	1) List Load Module 2) List Object Module 3) List Identification Records from one or more CSECTs 4) List Link Pack area
CMPRSEQ		1) Compare members of two partitioned datasets
COPYMODS	BLKCNT | NOBLKCNT BYTES CHGVOL | NOCHGVOL COPYLTM | SKIPLTM CORRBLKS CUMTOT | CUMSEP EOF1 | NOEOF1 EOF2 | NOEOF2 EOFS | NOEOFS EOV1 | NOEOV1 EOV2 | NOEOV2 EOV2EOF | NOEOVCHG EOVS | NOEOVS EXNULL | NOEXNULL FOOTAGE FILELIMIT HDR1 | NOHDR1 HDR2 | NOHDR2 HEXPRT IDRCOFF	1) Copy tape to 3 output tapes, SL output, change VOL1s 2) Simulate copy, print input tape information

	KEEPVOL LABADDIN LABELIMIT LABLDUMP | NOLABELD LBLFIX | NOLBLFIX LBLINFO | NOLABL MINMAX OUTVOLALL PRADDLBL | NOPRADDL PRINTRCDS READ | WRITE RECSIZE SECOFF SHOOVL STRIP SYSIN | NOSYSIN TAPEOWNER VOLLBL | NOVOLLBL	
DSSDUMP	DUMP EXCLUDE INCLUDE OPTIONS PREFIX RENAME STRIP	1) Run a test (simulation) dump 2) Dump a dataset, substituting new HLQ on backup set 3) Dump two groups of datasets contained on one volume
DSSREST		1) Read backup set, generate report and restore JCL 2) Restore a single dataset from backup set 3) Restore more than one, but not all, datasets from backup set

FIXDSCB	EXPIRE
	EXTEND
	NAME
	  BLKSIZE
	  DSORG
	  KEYL
	  LRECL
	  OPTCODE
	  RECFM
	  RKP
	PROTECT
	RENAME
	RENEW
	SCRATCH
	SETNOPWR
	UNLOCK
	ZEROEXPD
		1) Extend expiration date for dataset
		2) Modify RECFM, LRECL, BLKSIZE

ICKDSF	INIT	1) Initialize 2314, 3330, 3340, 3350 DASD 2) Initialize 3375, 3380, 3390 DASD
IEBCOMPR	COMPARE EXITS LABELS	1) Compare two sequential datasets 2) Compare two partitioned datasets

IEBCOPY	COPY EXCLUDE SELECT	1) Copy partitioned dataset from one DASD to another DASD, including all members 2) Copy partitioned dataset from one DASD to another DASD, selecting specific members 3) Copy members from two partitioned datasets, creating a new partitioned dataset, replacing any members with the same name from dataset 2 4) Compress partitioned dataset 5) Unload a partitioned dataset to tape 6) Reload some members of an unloaded partitioned dataset
IEBGENER	EXITS GENERATE LABELS MEMBER RECORD	1) Copying tape to tape 2) Tape to print 3) Card to tape
IEBISAM	COPY EXIT LOAD PRINTL UNLOAD	1) Copy ISAM dataset blocking data records 2) Copy ISAM dataset to portable copy (unloaded) 3) Reload ISAM dataset from unloaded copy 4) Print ISAM data records
IEBPTPCH	EXITS LABELS MEMBER PRINT PUNCH RECORD TITLE	1) Print sequential dataset with conversion to hex 2) Print partitioned dataset, 10 records from each member 3) Copy all members from PDS onto single file on tape 4) Print two members of PDS 5) Resequence card deck
IEBUPDTE	ADD ALIAS CHANGE DELETE ENDUP NUMBER REPL REPRO	1) Add two members to a partitioned dataset 2) Change member in partitioned dataset, update in place 3) Add records to member in partitioned dataset 4) Delete records from a member in a partitioned dataset 5) Create a partitioned dataset, input data contained solely in control dataset 6) Create a partitioned dataset, input data copied from existing dataset
IEHDASDR	ANALYZE DUMP LABEL RESTORE	1) Initialize 2314, 3330 (1 and 2), 3340, 3350 DASD volumes 2) Dump DASD volume to tape 3) Restore DASD volume from backup 4) Alter Volume Serial on offline 2314
IEHINITT	INITT	1) Initialize 3 tapes
IEHLIST	LISTCTLG LISTPDS LISTVTOC	1) List VTOC 2) List PDS
IEHMAP	ATTRIB AVAIL CCHHR DIR DSNAME DUMP DUMP456 FIXUP	1) List all datasets on volume All other functions are documented in the job above.

	FREE MAP MISSING NAME OVERLAP PDS SORT TRACKS TTR VERIFY	
IEHMOVE	COPY DSGROUP COPY DSNAME COPY PDS COPY VOLUME MOVE DSGROUP MOVE DSNAME MOVE PDS MOVE VOLUME	1) Copy partitioned dataset from one DASD to another DASD, recataloging 2) Copy all datasets from one DASD volume to another DASD volume 3) Move all datasets matching HLQ from one DASD to another DASD, recataloging 4) Unload all datasets matching HLQ from DASD to tape 5) Reload partitioned dataset from tape to DASD 6)
IEHPROGM	ADD CATLG DELETEP LIST RENAME REPLACE SCRATCH UNCATLG	1) Scratch all expired datasets, including SYS datasets 2) Remove all datasets on volume, regardless of expiration status 3) Scratch and uncatalog two datasets 4) Rename and recatalog a dataset 5) Add passwords to a dataset 6) List passwords for a dataset 7) Rename a member of a Partitioned dataset
LISTPDS	DECK | NODECK EJECT | NOEJECT EROPT HEXOUT | NOHEXOUT LINECNT LIST | NOLIST LISTDIR NUM | NONUM RITS | CRBE | CRJE SELECT | NOSEL | EXCLUDE SPF | NOSPF SSI | NOSSI TRUNC | NOTRUNC UPDTE | NOUPDTE XLATE | TEXT | NOXLATE | NOTEXT	1) Print the directory of a partitioned dataset 2) Print all members of a partitioned dataset 3) Punch selected members of a partitioned dataset 4) Punch selected members of a partitioned dataset with IEBUPDTE cards 5) Punch and print sequential dataset
MINIUNZ		1) List contents of archive 2) Extract entire contents of archive to a partitioned dataset 3) Extract a single member from an archive to a sequential dataset
MINIZIP		1) Compress all members of a partitioned dataset to archive 2) Compress all members of three partitioned datasets to archive 3) Compress three sequential datasets to archive 4) Compress four members of a partitioned dataset to archive
OFFLOAD	E (Exclude) O (Offload) S (Select)	1) Offload selected members of a partitioned dataset 2) Offload all but excluded member of a partitioned dataset 3) Offload all members of a partitioned dataset

PDSLOAD	CK3 CTL(xx) NAME={CHK|ASIS|IBM} NEW S(namemask) SPF UPDTE(x)	1) Load partitioned dataset from OFFLOADed dataset 2) Load partitioned dataset using mask to select members 3) Add members to partitioned dataset from OFFLOADed dataset
PDSMATCH		1) Compare two partitioned datasets, names only 2) Compare two partitioned datasets, names and userdata 3) Compare two partitioned datasets, names and actual data records
PDSPRINT	PRINT PUNCH	1) Punch all members with IEBUPDTE ADD cards 2) Print selected members 3) Punch long list of selected members 4) Print members selected with name pattern
PDSPROGM	ALIAS DELETE FILE OPTION RENAME	1) Partitioned dataset maintenance
PDSSCAN		1) Scan two partitioned datasets searching for strings
PDSUPDTE		1) Scan two partitioned datasets searching for strings 2) Scan a partitioned dataset, two matchings strings required, update if found
PDSUR	LIST MEMBER RELOAD UNLOAD	1) Unload two partitioned datasets 2) Reload a partitioned dataset 3) Unload and reload a partitioned dataset in one step 4) List three unloaded partitioned datasets 5) Unload with selects and excludes 6) Reload with selects and excludes
RECV370		1) Unpack XMIT to create PDS
RESETDS		1) Reset datasets to empty
REVLMOD		1) Reload load module from sequential dataset
SORT	SORT MERGE RECORD MODS END	1) Sort 1 tape dataset, creating a disk dataset 2) Sort 2 tape datasets (implicit merge), creating a disk dataset 3) Merge 2 disk datasets, creating a disk dataset 4) Sort disk dataset on split sequence field 5) Sort disk dataset on split sequence field, all CH data
SUPERLST		1) List VTOCs and PDS directories on two volumes
SYSREPRO		1) Print a card dataset 2) Copy a sequential dataset, disk to disk 3) Copy a tape dataset to disk 4) Add member to existing partitioned dataset from cards
TAPEMAP		1) Analyze tape and report on files and their contents
TAPESCAN		1) Print dataset VTOC for SL tape

		2) Copy 4th and 5th datasets to second tape
UNUPDTE		1) Offload members of a partitioned dataset to sequential dataset
UPDTE		1) Load partitioned dataset from sequential dataset
VTOCLIST		1) List VTOC for DASD volume 2) List VTOC for DASD volume with PDS directory information
XMIT370		1) Pack partitioned dataset into XMIT
ZAPDSCB		1) Modify DSCB fields for dataset
ZTDUMPTP	FILES MODE PRINT REC REDUN RUN SKPFIL SKPREC	1) Dump and print first 20 records from first 5 files on tape
ZZRELINK	LINK SELECT	1) Create (on SYSPUNCH) JCL for job to relink a load module 2) Re-link-edit all members of a load library to a new library 3) Re-link-edit a load module, renaming it into the input library 4) Re-link-edit a group of modules to a new library, replacing duplicates

AMASPZAP **[supplied by IBM, located in SYS1.LINKLIB]**

Used for inspecting and/or modifying a load module or data record stored on disk. AMASPZAP requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSLIB - defines the load library or dataset that will be accessed.

  DSNAME and DISP (either OLD or SHR) are required.
  The VOLUME and UNIT parameters are necessary only if the dataset is not cataloged.
  When the dataset is the VTOC, DSNAME=FORMAT4.DSCB must be specified.
  The SYSLIB DD statement cannot define a concatenated dataset.
- SYSIN - defines a sequential input dataset that contains AMASPZAP control statements.

Control Statements

```
NAME member

     csect
```

The NAME control statement specifies the CSECT in a load module that is to be the object of subsequent VERIFY, REP, IDRDATA or SETSSI operations. The parameters are:

member

Specifies the member name of the load module that contains the control section in which the data to be inspected and/or modified is contained. The load module must be a member of the partitioned dataset defined by the SYSLIB DD statement.

csect

Specifies the name of the particular control section that contains the data to be verified or replaced. When this parameter is omitted, it is assumed that the first CSECT contained in the load module is the one to be referenced. If there is only one CSECT in the load module, this parameter is not required.

```
CCHHR record address
```

The CCHHR control statement specifies the address of a physical record on a direct access device that is to be modified or verified. The record must be located within the dataset defined by the SYSLIB DD statement. Any immediately following REP or VERIFY control statements will reference the data in the specified record. The parameter is:

record address

Specifies the actual direct access device address of the record containing the data to be replaced or verified. It must be specified as a 10-digit hexadecimal number in the form ccccchhhhrr, where cccc is the cylinder, hhhh is the track, and rr is the record number. For example, 0001000A01 addresses record 1 of cylinder 1, track 10. A zero record number is invalid and will default to 1.

More than one CCHHR control statement can be supplied in the input stream, however, the VERIFY, REP and SETSSI control statements associated with each CCHHR control statement must immediately follow the specific CCHHR control statement to which they apply.

```
VER | VERIFY   offset

               expected content
```

The VERIFY control statement specifies that the contents at a specified location within a control section or physical record are to be compared with the data the user supplies in the control statement. VERIFY may be abbreviated to VER. If the two fields being compared are not in agreement, no succeeding REP or SETSSI operations will be performed until the next NAME or CCHHR control statement is encountered. A formatted dump is provided of each CSECT or record for which a VERIFY operation failed. The parameters are:

offset

Specifies the hexadecimal displacement of the data to be inspected in a CSECT or record. This displacement must be specified as a multiple of two hexadecimal digits (0D, 021C, 014682, etc.). If this offset value is outside the limits of the CSECT or data record defined by the preceding NAME or CCHHR control statement, the VERIFY control statement will be rejected. When inspecting a record with a key, the length of the key should be considered in the calculation of the displacement;

that is, offset zero is the first byte of the key.

expected content

Specifies the bytes of data that are expected at the specified location. As with the offset parameter, the number of bytes of data defined must be specified as a multiple of two hexadecimal digits. If desired, the data within the parameters may be separated by commas (never blanks), but again, the number of digits between commas must also be a multiple of two. For example, the data may look like this: 5840C032 (no commas), or this: 5840,C032 (commas).

If all the data will not fit into one VERIFY control statement (one 80-byte logical record), then another VERIFY control statement must be supplied.

```
REP  offset

     data
```

The REP control statement modifies data at a specified location in a CSECT or physical record that has been previously defined by a NAME or CCHHR control statement. The data specified on the REP control statement will replace the data at the record or CSECT location stipulated in the offset parameter field. (Note: You should always first use the VERIFY function to make sure you know what you are going to change with the REP function.) A message will be issued to record the contents of the specified location as they were before the change was made. The parameters are:

offset

Specifies the hexadecimal displacement of the data to be inspected in a CSECT or record. This displacement must be specified as a multiple of two hexadecimal digits (0D, 021C, 014682, etc.). If this offset value is outside the limits of the CSECT or data record defined by the preceding NAME or CCHHR control statement, the REP control statement will be rejected. When modifying a record with a key, the length of the key should be considered in the calculation of the displacement; that is, offset zero is the first byte of the key.

data

Specifies the bytes of data that are to be inserted at the specified location. As with the offset parameter, the number of bytes of data defined must be specified as multiples of two hexadecimal digits. If desired, the data within the parameter may be separated by commas (never blanks), but again, the number of digits between commas must also be a multiple of two. For example, the data may look like this: 4160BB20 (no commas), or this: 4160,BB20 (commas).

If all the data to be modified will not fit into one REP control statement (one 80-byte logical record), then another REP control statement must be supplied.

AMASPZAP automatically updates the system status index (SSI) when it successfully modifies the associated load module.

If multiple VERIFY and REP operations are to be performed on a CSECT, then all the VERIFY control statements should precede all the REP control statements. This will ensure that all the REP operations are ignored if any VERIFY reject occurs.

When a record in the VTOC (that is, a DSCB) is accessed for modification, a message is written to the console.

```
IDRDATA xxxxxxxx
```

The IDRDATA control statement places up to eight bytes of user data into the CSECT Identification Record of the load module. This is only done if a REP operation associated with a NAME control statement is performed and the load module has been processed by the Linkage Editor to include CSECT Identification Records. The parameter is:

xxxxxxxx

Specifies one to eight characters of user data (with no embedded blanks) that is to be placed in the user data field of the IDR of the load module. If more than eight characters are in the parameter field, only the first eight characters will be used.

The IDRDATA control statement is valid only when used in conjunction with the NAME control statement. It must follow its associated NAME control statement and precede any DUMP or ABSDUMP control statement. IDRDATA control statements associated with CCHHR control statements will be ignored.

```
SETSSI xxyynnnn
```

The SETSSI control statement places user supplied system status information in the partitioned dataset directory entry for the library member specified in the preceding NAME control statement. The SSI, however, must have been created when the load module was link edited. The parameter is:

xxyynnnn

Specifies the 4 bytes of system status information the user wishes to place in the SSI field for this member. Each byte is supplied as two hexadecimal digits signifying the following:

xx - change level
yy - flag byte
nnnn - modification serial number

If an error has been detected in any previous VERIFY or REP operation, the SETSSI function will not be performed.

Since all bits in the SSI entry are set (or reset) by the SETSSI control statement, extreme care should be exercised in its use to avoid altering information vital to the depiction of the maintenance status of the program being changed. A message will be issued to record the SSI as it was before the SETSSI operation was performed.

```
DUMP | DUMPT  member

             csect | ALL
```

The DUMP control statement dumps a specific control section or all control sections in a load module. The

format of the output of this dump is hexadecimal. The DUMPT control statement differs from the DUMP control statement in that it also gives the user an EBCDIC and instruction mnemonic translation of the hexadecimal data. The parameters are:

member

Specifies the member name of the load module that contains the control section(s) to be dumped. This module must be a member of a partitioned dataset that is defined by the SYSLIB DD statement.

csect

Specifies the name of the particular control section that is to be dumped. To dump all the CSECTs of a load module, code ALL instead of the CSECT name. If the CESCT parameter is omitted entirely, it is assumed that the user means to dump only the first control section contained in the load module.

```
ABSDUMP | ABSDUMPTstartaddr stopaddr | membername | ALL
```

The ABSDUMP control statement dumps a group of data records, a member of a partitioned dataset, or an entire dataset, as defined in the SYSLIB DD statement. If the key associated with each record is to be formatted, DCB=(KEYLEN=nn), when nn is the length of the record key, must also be specified by the SYSLIB DD statement. Note that when dumping a VTOC, DCB=(KEYLEN=44) should be specified; when dumping a PDS directory, DCB=(KEYLEN=8) should be specified.

ABSDUMP produces a hexadecimal printout only, while ABSDUMPT prints the hexadecimal data, the EBCDIC translation, ant the mnemonic equivalent of the data. The parameters are:

startaddr

Specifies the absolute direct access device address of the first record to be dumped. This address must be specified in hexadecimal in the form ccccchhhhrr (cylinder, track and record number).

stopaddr

Specifies the absolute direct access device address of the last record to be dumped, and it must be in the same format as the start address.

Both addresses must be specified when this method of dumping records is used, and both addresses must be within the limits of the dataset defined by the SYSLIB DD statement. The record number specified in the start address must be a valid record number. The record number specified as the stop address need not be a valid record number, but if it is not, the dump will continue until the last record on the track specified in the stop address has been dumped.

membername

Specifies the name of a member of a partitioned dataset. The member can be a group of data records or a load module. In either case, the entire member is dumped when this parameter is specified.

ALL

Specifies that the entire dataset defined by the SYSLIB DD statement is to be dumped. How much of the space allocated to the dataset is dumped depends on how the dataset is organized: for sequential datasets, the dump will continue until it reaches end of file; for indexed sequential and direct access datasets, the dump will contain all extents; for partitioned datasets, the dump will contain all extents, including all linkage editor control records, if any exist.

BASE xxxxxx

The BASE control statement is used to adjust offset values that are to be specified in any subsequent VERIFY and REP control statements. This control statement should be used when the offsets given in the VERIFY and REP control statements for a CSECT are to be obtained from an assembly listing in which the starting address of the CSECT is not location zero.

For example, assume that CSECT ABC begins at assembly listing location x'000400', and that the data to be replaced in this CSECT is at location x'000408'. The actual displacement of the data in the CSECT is x'08'. However, an offset of 'x0408' (obtained from the assembly listing location x'000408') can be specified in the REP control statement if a BASE statement specifying x'000400' is included prior to the REP control statement in the input stream. When the REP control statement is processed, the base value x'000400' will be subtracted from the offset x'0408' to determine the proper displacement of data within the CSECT. The parameter is:

xxxxxx

Specifies a 6-character hexadecimal offset that it to be used as a base for subsequent VERIFY and REP operations. This value should reflect the starting assembly listing address of the CSECT being inspected or modified.

The BASE control statement should be included in the input stream immediately following the NAME control statement that identifies the control section that is to be involved in the operations. The specified base value remains in effect until all VERIFY, REP, and SETSSI operations for the CSECT have been processed.

Examples

Dump the CSECT from a load module

```
//AMASPZAP JOB (SYS),'AMASPZAP DUMP',CLASS=A,MSGCLASS=X
//AMASPZAP EXEC PGM=AMASPZAP
//SYSPRINT DD  SYSOUT=*
//SYSLIB   DD  DISP=SHR,DSN=SYSC.LINKLIB
//SYSIN    DD  *
  NAME   IKFCBL01 IKF011
  DUMPT IKFCBL01 IKF011
/*
//
```

Verify/Replace instruction in a load module

```
//AMASPZAP JOB (SYS),'AMASPZAP REP ',CLASS=A,MSGCLASS=X
//AMASPZAP EXEC PGM=AMASPZAP
//SYSPRINT DD SYSOUT=*
//SYSLIB DD DISP=SHR,DSN=JAY01.LOAD
//SYSIN DD *
```

```
  NAME ISLOAD
  VER 0616 9640,1051
  REP 0616 966B,1051
  SETSSI 01212134
/*
//
```

Dump a record from a file

```
//AMASPZAP JOB (SYS),'AMASPZAP DUMP',CLASS=A,MSGCLASS=X
//AMASPZAP EXEC PGM=AMASPZAP
//SYSPRINT DD SYSOUT=*
//SYSLIB DD DISP=SHR,DSN=JAY01.ISAM.MASTER
//SYSIN DD *
  ABSDUMPT 007A000120 007A000120
/*
//
```

Inspect/Modify a data record

```
//AMASPZAP JOB (SYS),'AMASPZAP CH VTOC',CLASS=A,MSGCLASS=X
//AMASPZAP EXEC PGM=AMASPZAP
//SYSPRINT DD SYSOUT=*
//SYSLIB DD DISP=OLD,DSN=FORMAT4.DSCB,UNIT=3330,VOL=SER=111111,DCB=(KEYLEN=44)
//SYSIN DD *
  CCHHR   005000001
  VERIFY  2C 0504
  REP     2C 0A08
  REP     2E 0001,03000102
  ABSDUMPT ALL
/*
//
```

Source: GC28-0633-1 OS/VS Service AIDS

AMBLIST **[supplied by IBM, located in SYS1.LINKLIB]**

Used for producing a formatted listing of an object or load module, producing a map and cross-reference listing of a nucleus, producing a list of IDRDATA from a CSECT, or producing a map of all modules in the reenterable load module area (the Link Pack area). AMBLIST requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSLIB or any DD name specified on the control statement - defines the load or object library that will be accessed.
- SYSIN - defines a sequential input dataset that contains AMBLIST control statements.

Control Statements

```
LISTLOAD[OUTPUT={MODLIST | XREF | BOTH}]

        [,TITLE=('title',position)]

        [,DDN=ddname]

        [,MEMBER={membername | (membername[,membername]...)]
```

```
      [,RELOC=nnnnnn]
```

The LISTLOAD control statement produces a listing of the load module along with its module map and cross-reference listing. The parameters are:

OUTPUT=type

Specifies the type of load module listing to be produced. The choices for type are:

- MODLIST - Requests a formatted listing of the control and text records, including its External Symbol Dictionary and Relocation Dictionary Records.
- XREF - Requests a module map and cross-reference listing.
- BOTH - Requests both MODLIST and XREF.

If omitted, BOTH is assumed.

TITLE=('title',position)

Specifies a title, one to forty characters, to be printed below the heading line on each page of the output. The optional position specifies the number of characters from the left margin the user's title is to be indented. There may be no commas included in the 'title' text, as commas are recognized as delimiters; any characters beyond the first comma encountered will be ignored.

DDN=ddname

Identifies the DD statement that defines the dataset containing the input module. If omitted, SYSLIB is the default DD name.

MEMBER={membername | (membername[,membername]...)

Identifies the input load module(s) by member name or alias name. To specify more than one load module, enclose the list of names in parentheses and separate the names with commas. If you omit the MEMBER= parameter, all modules in the dataset will be printed.

RELOC=nnnnnn

Specifies a relocation or base address in hexadecimal of up to six characters. When the relocation address is added to each relative map and cross-reference address, it gives the absolute main storage adress for each item on the output listing. If omitted, no relocation is performed.

```
LISTOBJ[TITLE=('title',position)]

      [,DDN=ddname]

      [,MEMBER={membername | (membername[,membername]...)}]
```

The LISTOBJ control statement produces a listing that contains the External Symbol Dictionary, the Relocation Dictionary, and the text of the program containing instructions and data. The parameters are:

TITLE=('title',position)

Specifies a title, one to forty characters, to be printed below the heading line on each page of the output. The optional position specifies the number of characters from the left margin the user's title is to be indented. There may be no commas included in the 'title' text, as commas are recognized as delimiters; any characters beyond the first comma encountered will be ignored.

DDN=ddname

Identifies the DD statement that defines the dataset containing the input module. If omitted, SYSLIB is the default DD name.

MEMBER={ membername | (membername[,membername]...)

Identifies the input load module(s) by member name or alias name. To specify more than one load module, enclose the list of names in parentheses and separate the names with commas. If you omit the MEMBER= parameter, all modules in the dataset will be printed.

```
LISTIDR[OUTPUT={IDENT | ALL}]

       [,TITLE=('title',position)]

       [,DDN=ddname]

       [,MEMBER={membername | (membername[,membername]...)}]
```

The LISTIDR control statement produces a formatted listing of all information in a load module's CSECT identification records, including:

- the version and modification level of the language translator and the date that each CSECT was translated,
- the version and modification level of the linkage editor that built the load module and gives the date the load module was created,
- identifies by date modifications to the load module that may have been performed by AMASPZAP.

The parameters are:

OUTPUT={IDENT | ALL }

Specifies whether the listing should include all CSECT identification records or only those containing AMASPZAP data and user data. If the parameter is omitted or OUTPUT=ALL is specified, all IDRs associated with the module will be printed. If OUTPUT=IDENT is specified, only those IDRs that contain AMASPZAP data or user-supplied data will be printed.

TITLE=('title',position)

Specifies a title, one to forty characters, to be printed below the heading line on each page of the output. The optional position specifies the number of characters from the left margin the user's title is to be indented. There may be no commas

included in the 'title' text, as commas are recognized as delimiters; any characters beyond the first comma encountered will be ignored.

DDN=ddname

Identifies the DD statement that defines the dataset containing the input module. If omitted, SYSLIB is the default DD name.

MEMBER={ membername | (membername[,membername]...)

Identifies the input load module(s) by member name or alias name. To specify more than one load module, enclose the list of names in parentheses and separate the names with commas. If you omit the MEMBER= parameter, all modules in the dataset will be printed.

`LISTLPA`

The LISTLPA control statement produces a map of all modules in the reenterable load module area (the Link Pack area). There are no parameters for this control statement.

Examples

List Load Module

```
//AMBLIST  JOB (SYS),AMBLIST,CLASS=A,MSGCLASS=X
//AMBLIST  EXEC PGM=AMBLIST
//SYSPRINT DD  SYSOUT=*
//SYSLIB   DD  DISP=SHR,DSN=SYSC.LINKLIB
//SYSIN    DD  *
  LISTLOAD OUTPUT=BOTH,MEMBER=IKFCBL01
//
```

List Object Module

```
//AMBLIST  JOB (SYS),AMBLIST,CLASS=A,MSGCLASS=X
//AMBLIST  EXEC PGM=AMBLIST
//SYSPRINT DD  SYSOUT=*
//SYSLIB   DD  DISP=SHR,DSN=MVTSRC.COBOL.OBJECT,
//             UNIT=SYSDA,VOL=SER=MVTSRC
//SYSIN    DD  *
  LISTOBJ  MEMBER=IKFCBL01
//
```

List Identification Records from one or more CSECTs

```
//AMBLIST  JOB (SYS),AMBLIST,CLASS=A,MSGCLASS=X
//AMBLIST  EXEC PGM=AMBLIST
//SYSPRINT DD  SYSOUT=*
//SYSLIB   DD  DISP=SHR,DSN=SYSC.LINKLIB
//SYSIN    DD  *
  LISTIDR  OUTPUT=IDENT,MEMBER=IKFCBL01
//
```

List Link Pack area

```
//AMBLIST  JOB (SYS),AMBLIST,CLASS=A,MSGCLASS=X
//AMBLIST  EXEC PGM=AMBLIST
```

```
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  LISTLPA
//
```

Source: GC28-0633-1 OS/VS Service AIDS

CMPRSEQ **[written by David B. Cole, located in SYSC.LINKLIB]**

Used to compare two sequential datasets or members of two partitioned datasets at the logical record level. CMPRSEQ was written by David B. Cole and the source may be found on CBT Tape File #226 (Version 437).

The CMPRSEQ program compares two sequential card image files and reports precisely the differences between the two. Such differences can consist of insertions, deletions, and replacements (of either equal or unequal sizes). CMPRSEQ accomplishes this by reading the two input files alternately and comparing the card images as it goes along. When it encounters two cards that mismatch, it saves them and continues to read the two files alternately. Each time it reads a card from one file, it compares it to all cards that it has read from the other file since the start of the mismatch. If no match is found, then it saves that card and proceeds to read the next card from the other file. CMPRSEQ continues alternating back and forth in this manner until it has read a card that does match one of the saved cards from the other file. The mismatch then consists of all cards in the two saved stacks that are below the two matching cards.

CMPRSEQ requires the following DD statements:

- SYSPRINT - a sequential output message dataset.
- OLD - sequential input: first of the input datasets to be compared.
- NEW - sequential input: second of the input datasets to be compared.
- OLDLIST - sequential output dataset where the records read from OLD are reproduced, along with the locations of mismatches.
- NEWLIST - sequential output dataset where the records read from NEW are reproduced, along with the locations of mismatches.
- IGNORE - sequential input dataset that can contain images of records that appear repeatedly and are to be ignored.

  The ignore file is optional. If it is available, then it is used to help control the possible desynchronization problems. The ignore file should contain copies of card images that appear repeated ly throughout the files being compared, and especially appearing in areas affected by insertions and deletions. Copies of these cards, when encountered during a mismatch resolution process, will not be used to resolve the mismatch

- SYNC - sequential input dataset that can be used to manage desynchronization problems.

  This file should contain copies of one or more cards each of which appears exactly once in both the old and new files. The cards in the SYNC file should appear in the same order by which they appear in the OLD and NEW files. They should represent points in the OLD and NEW files at which you wish to force comparison synchronization. Generally, such points will be following those area where otherwise unrecoverable desynchronization has occurred.

Completion Codes

0 - processing has completed successfully. No mismatches have been found.
4 - processing has completed successfully. At least one mismatch has been found.
12 - processing has failed. A memory shortage has occurred.
16 - processing has aborted. One of the comparison files (OLD or NEW) is not available.

Program Parameter (PARM= on EXEC statement)

By default CMPRSEQ compares the records of the two input datasets on columns 1 through 72. By specifying a PARM='FULL' on the EXEC statement, CMPRSEQ may be directed to perform the comparison on columns 1 through 80.

Control Statements

There are no control statements for CMPRSEQ.

Examples

Compare members of two partitioned datasets

```
//CMPRSEQ  JOB (1),'COMPARE MEMBERS',CLASS=A,MSGCLASS=X
//CMPRSEQ  EXEC PGM=CMPRSEQ,REGION=1024K PARM=FULL  DEFAULT=1,72
//SYSPRINT DD  SYSOUT=*
//OLD      DD  DISP=SHR,DSN=SYS1.MACLIB(YREGS)
//NEW      DD  DISP=SHR,DSN=SYSP.MACLIB(YREGS)
//OLDLIST  DD  SYSOUT=*
//NEWLIST  DD  SYSOUT=*
//IGNORE   DD  *
//SYNC     DD  *
//
```

COPYMODS **[written by Paul Tokheim, located in SYSC.LINKLIB]**

Used to copy all files of an input tape onto 1 to 16 output tapes. The original purpose of this program was for copying the SHARE MVS MODs tape, but if you have any need for copying (or recovering data from) a tape, you probably should see if COPYMODS will solve your problem. COPYMODS was written by Paul Tokheim, but has been heavily modified and is now maintained by Sam Golob. The source may be found on CBT Tape File #229. COPYMODS requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines a sequential input dataset that contains COPYMODS option statements.
- IN - defines the input tape volume.
- OUT1 through OUT16 - defines the output tape volume(s).
- LABLDUMP - (optional) defines the sequential output dataset to hold label sets copied from IN.

- LABADDIN - (optional) defines the sequential input dataset holding label sets to insert in OUT1 (through OUT16).
- PARMREPT - (optional) defines the sequential output message dataset to receive diagnostic output from PARMCHEK.

Program Parameter (PARM= on EXEC statement)

COPYMODS will accept many of its options as keywords in the PARM parameter, but if you code SYSIN as the only keyword, all options will be read from the SYSIN DD. If an option is supplied both by PARM and SYSIN, the SYSIN version takes precedence.

PARM may be omitted if a SYSIN DD is provided; the presence of the SYSIN DD sets the PARM=SYSIN option on.

Control Statements

```
COPYLTM | SKIPLTM
```

If COPYLTM is coded, leading tape marks on the input tape are copied to the output tape(s). If SKIPLTM is coded, leading tape marks on the input tape are skipped. The default is COPYLTM. LTMCOPY is a synonym for COPYLTM.

```
LBLINFO | NOLABL
```

If LBLINFO is coded, information from all standard labels will be printed. If NOLABL is coded, no label information will be printed. The default is NOLABL. PRINTL and LABELS are synonyms for LBLINFO; NOLABEL is a synonym for NOLABL.

```
HDR1 | NOHDR1
```

If HDR1 is coded, HDR1 labels will be printed. If NOHDR1 is coded, HDR1 labels will not be printed.

```
HDR2 | NOHDR2
```

If HDR2 is coded, HDR2 labels will be printed. If NOHDR2 is coded, HDR2 labels will not be printed.

```
EOF1 | NOEOF1
```

If EOF1 is coded, EOF1 labels will be printed. If NOEOF1 is coded, EOF1 labels will not be printed.

```
EOF2 | NOEOF2
```

If EOF2 is coded, EOF2 labels will be printed. If NOEOF2 is coded, EOF2 labels will not be printed.

`EOV1 | NOEOV1`

If EOV1 is coded, EOV1 labels will be printed. If NOEOV1 is coded, EOV1 labels will not be printed.

`EOV2 | NOEOV2`

If EOV2 is coded, EOV2 labels will be printed. If NOEOV2 is coded, EOV2 labels will not be printed.

`HDRS | NOHDRS`

If HDRS is coded, all HDR labels will be printed. If NOHDRS1 is coded, HDR labels will not be printed.

`EOFS | NOEOFS`

If EOFS is coded, all EOF labels will be printed. If NOEOFS1 is coded, EOF labels will not be printed.

`EOVS | NOEOVS`

If EOVS is coded, all EOV labels will be printed. If NOEOVS1 is coded, EOV labels will not be printed.

`CHGVOL | NOCHGVOL`

If CHGVOL is coded, the VOLSER on the output tape(s) will be changed to the serial coded in JCL. If NOCHGVOL is coded, the VOLSER on the output tape(s) will not be changed. JCLVOL and NEWVOL are synonyms for CHGVOL; NOJCLVOL and NONEWVOL are synonyms for NOCHGVOL.

`VOLLBL | NOVOLLBL`

If VOLLBL is coded, the VOL1 label will be printed. If NOVOLLBL is coded, the VOL1 label will not be printed. The default is NOVOLLBL.

`EOV2EOF | NOEOVCHG`

If EOV2EOF is coded, the EOV label will be changed to EOF. If NOEOVCHG is coded, the EOV label is not converted. The default is NOEOVCHG. EOVTOEOF and EOVCHG are synonyms for EOV2EOF.

`READ | WRITE`

If READ is coded, the input tape will be read, but no output tape(s) will be written. If WRITE is coded, the output tape(s) will be written. The default is WRITE. READONLY and NOWRITE are synonyms for

READ.

`LABLDUMP | NOLABELD`

If LABLDUMP is coded, all standard labels will be written to LABLDUMP DD. If NOLABELD is coded, standard labels will not be written to LABLDUMP. The default is NOLABELD.

`SYSIN | NOSYSIN`

If SYSIN is coded, other options will be read from SYSIN DD. If NOSYSIN is coded, options are not read from SYSIN. The default is NOSYSIN, except if a SYSIN DD card is supplied, SYSIN becomes the default.

`OUTVOLALL=volser`

If OUTVOLALL is coded, the VOLSER on the output tape(s) will be changed to volser.

`TAPEOWNER=ownerid`

If TAPEOWNER is coded, the owner field on the output tape(s) will be changed to ownerid.

`LABADDIN`

If LABADDIN is coded, label sets will be read from LABADDIN DD and merged with the data files read from an unlabeled input tape to create standard labels on the output tape(s).

`BLKCNT | NOBLKCNT`

If BLKCNT is coded with LABADDIN, EOF1/EOV1 block counts will be corrected on the output tape(s). If NOBLKCNT is coded, block counts in EOF1/EOV1 labels will not be changed. The default is BLKCNT.

`PRADDLBL | NOPRADDL`

If PRADDLBL is coded, labels created with LABADDIN will be printed. If NOPRADDL is coded, labels created with LABADDIN will not be printed.

`CORRBLKS`

If CORRBLKS is coded, EOF1/EOV1 block counts will be corrected on the output tape(s).

`EXNULL | NOEXNULL`

If EXNULL is coded, with LABADDIN, null standard labels sets will be added at end of tape if the input unlabeled tape runs out of files. If NOEXNULL is coded, adding labels will stop at end of the input tape. SLNULL is a synonum for EXNULL. NOSLNULL is a synonum for NOEXNULL.

`LBLFIX | NOLBLFIX`

If LBLFIX is coded, with LABADDIN, fix missing EOF labels. If NOLBLFIX is coded, missing EOF labels will not be fixed.

`INITVOLS`

If INITVOLS is coded, the output tape volumes will be initialized without an input tape volume. Code OUTVOLALL=serial and TAPEOWNER=owner and all output tapes will be initialized identically.

`BYTES`

If BYTES is coded, bytes read from all the files on the input tape will be counted.

`CUMTOT | CUMSEP`

If CUMTOT is coded, a running total of bytes counted and reported for all the files read from the input tape. At the end of the input tape, total bytes are broken into: grand total, bytes read from labels, and bytes from files. If you code CUMSEP the total label bytes are cumulatively reported separately from the cumulative data bytes.

`FILELIMIT=nnn`

If FILELIMIT is coded, the number of files copied from the input tape is limited to the number specified in nnn. For a standard label input tape, the number entered for nnn will be multiplied by 3 to compensate for the labels. The adjustment of nnn may be forced by coding option SLLIM for standard label volumes and NLLIM for unlabled volumes.

`LABELIMIT=nnn`

If LABELIMIT is coded, the number of label sets copied from the LABADDIN DD is limited to the number specified in nnn.

`RECSIZE`

If RECSIZE is coded, the detailed block size of every block in every file read from the input tape volume will be displayed. Coding RECSIZE automatically produces the MINMAX option.

`MINMAX | NOMINMAX`

If MINMAX is coded, the minimum and maximum block size of every file read from the input tape volume will be displayed. Coding NOMINMAX turns off this display.

`PRINTRCDS=nnn`

If PRINTRCDS is coded, the number of records to print at the beginning of each tape file, turned on by the HEXPRT option, is specified by nnn. The default for n is 4.

`HEXPRT`

If HEXPRT is coded, the first 132 bytes of the first n records in each file are printed in HEX and EBCDIC. The value of n is set by the PRINTRCDS option and the default value is 4.

`STRIP`

If STRIP is coded, standard labels are not copied from the input tape file onto the output tape(s); the output tape(s) are unlabeled.

`IDRCOFF`

If IDRCOFF is coded, a virtual device on AWS format tapes is handled as non-compressed.

`SECOFF`

If SECOFF is coded, the security indicators in tape labels for the output tape(s) are turned off.

`FOOTAGE`

If FOOTAGE is coded, the estimate tape size, at 6250 BPI, is calculated and reported.

`KEEPVOL`

If KEEPVOL is coded, an existing VOL1 label on an output tape will be retained.

`SHOOVL`

If SHOOVL is coded, in combination with READ, COPYMODS will do a dry run to show you the results of a future copy.

Examples

Copy tape to 3 output tapes, SL output, change VOL1s

```
//COPYMODS JOB (001),'COPYMODS-1',CLASS=A,MSGCLASS=X
//COPYMODS EXEC PGM=COPYMODS,REGION=4096K,PARM='SYSIN'
//SYSPRINT DD  SYSOUT=*
//IN    DD VOL=SER=157806,DISP=SHR,UNIT=TAPE,LABEL=(,BLP,EXPDT=98000)
//OUT1  DD VOL=SER=022101,DISP=OLD,UNIT=TAPE,LABEL=(,SL,EXPDT=98000)
//OUT2  DD VOL=SER=022102,DISP=OLD,UNIT=TAPE,LABEL=(,SL,EXPDT=98000)
//OUT3  DD VOL=SER=022103,DISP=OLD,UNIT=TAPE,LABEL=(,SL,EXPDT=98000)
//SYSIN DD *
CHGVOL
CUMTOT
LABELS
TAPEOWNER=HERCULES
BYTES
MINMAX
SHOOVL
FOOTAGE
//
```

Simulate copy, print input tape information

```
//COPYMODS JOB (001),'COPYMODS-1',CLASS=A,MSGCLASS=X
//COPYMODS EXEC PGM=COPYMODS,REGION=4096K,PARM='SYSIN'
//SYSPRINT DD  SYSOUT=*
//IN    DD VOL=SER=22013C,DISP=SHR,UNIT=TAPE,LABEL=(,BLP,EXPDT=98000)
//OUT1  DD VOL=SER=022301,DISP=OLD,UNIT=TAPE,LABEL=(,SL,EXPDT=98000)
//SYSIN DD *
CHGVOL
CUMTOT
LABELS
TAPEOWNER=HERCULES
BYTES
MINMAX
READ
FOOTAGE
//
```

DSSDUMP **[written by Gerhard Postpischil, located in SYSC.LINKLIB]**

Used for creating a backup sequential dataset containing the contents of one or more DASD datasets. The output dataset may be written to a tape image or to a DASD dataset. DSSDUMP backup datasets are equivalent to those produced by ADRDSSU, which is a licensed program used under z/OS. DSSDUMP was written by Gerhard Postpischil and the source may be found on CBT Tape File #860. DSSDUMP requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines a sequential input dataset that contains DSSDUMP control statements.
- TAPE - defines the sequential output dataset that will contain the backup set, in DSS format, uncompressed.

RECFM=U
For tape device output: 65520 is the default BLKSIZE; 7892<block size<65520 is supported, RECFM=V is supported, with blocks 7900<block size<32760.
For DASD device output: the default BLKSIZE=track size, if >=7892 or the half-track size for modulo devices; the supported range is 7892<data size<32760.

Control Statements

```
OPTIONS [{ENQ | NOENQ}]

        [,TEST]

        [,{ALLDATA | ALLEXCP}]

        [,EXPORT]

        [,{LIST | SHOW}]
```

The OPTIONS control statement specifies one or more options, separated by commas. The OPTIONS statement may appear anywhere in the stream of control statements. The parameters are:

ENQ | NOENQ

If ENQ is specified, DSSDUMP issues an exclusive ENQ TEST for each dataset. Dump continues if the dataset is not available, and issues RC=4. Applies from prior DUMP until end-of-job. If NOENQ is specified, which is the default if neither ENQ or NOENQ is specified, each dataset is written to the output dataset as is.

TEST

If TEST is specified, DSSDUMP bypasses all TAPE output. Note that file has been opened already, and will be empty.

ALLDATA | ALLEXCP

If ALLDATA or ALLEXCP, which are synonyms, is specified, DSSDUMP writes all tracks, including unused ones, to be written to the output dataset. This option may be needed to recover from invalid, non-zero LSTAR. This option also preserves LABEL=SUL data.

EXPORT

If EXPORT is specified, DSSDUMP modifies the output DSCB1 by removing any expiration date and password flags.

LIST | SHOW

If LIST or SHOW, which are synonyms, is specified, DSSDUMP will write the current OPTION settings to the message dataset.

```
INCLUDE | DUMP  mask

                [VOLUME(serial)]
```

The INCLUDE or DUMP control statement - DUMP and INCLUDE are synonyms - specifies which datasets are selected to be written to the backup set. The INCLUDE or DUMP control statement occurs

one or more times for each execution of DSSDUMP. The parameters are:

mask

Specifies either a single dataset name, unquoted, or provides a mask which may match one or more datasets. A mask ending in a period (.) is treated as a mask followed by an implied double asterisk (**). In MVS (VSAM catalog), the first index level may not contain a mask character. Also note that VSAM catalog processing does not support detection or processing of alias entries.

A percent sign (%) is treated as a positional mask (one to one correspondence of characters/mask to dataset name). Any number of DUMP cards may be used in a run but only about 700 datasets will fit in the name table (which is an assembly time option).

Masking bytes are valid in any position in the mask.

VOLUME(serial)

Specifies inclusion of matching datasets on the specified VOLUME SERIAL only. If this results in duplicate dataset names, a .D#nnnnnn is appended to duplicates on higher volume serials (i.e., the cataloged entry may be the one that gets renamed).

`EXCLUDE mask`

The EXCLUDE control statement allows for excluding matching datasets which have been selected by a preceding DUMP or INCLUDE control statement. The parameter is:

mask

Specifies either a single dataset name, unquoted, or provides a mask which may match one or more datasets. A mask ending in a period (.) is treated as a mask followed by an implied double asterisk (**). In MVS (VSAM catalog), the first index level may not contain a mask character. Also note that VSAM catalog processing does not support detection or processing of alias entries.

A percent sign (%) is treated as a positional mask (one to one correspondence of characters/mask to dataset name). Any number of DUMP cards may be used in a run but only about 700 datasets will fit in the name table (which is an assembly time option).

Masking bytes are valid in any position in the mask.

`PREFIX string`

The PREFIX control statement causes all dataset names to be prefixed by the specified text string. It is not required to be an index level (e.g., SYS9.), but if not, generated names may be syntactically invalid. Result names are truncated to 44 characters, and a trailing period is blanked. Only one PREFIX card may be used per execution, and it is mutually exclusive of RENAME and STRIP options.

```
STRIP string
```

The STRIP control statement specifies that the specified string is removed from any dataset name where it is found. Multiple STRIP and RENAME requests are supported (up to 16; which is an assembly time option).

```
RENAME oldname

       newname
```

The RENAME control statement specifies that wherever oldname is found in a selected dataset name it is to be replaced by newname. Up to 16 combined RENAME and STRIP requests are legal. All strings in PREFIX/RENAME/STRIP are limited to 23 characters (which is an assembly time option).

Examples

Run a test (simulation) dump

```
//DSSDUMP  JOB (SYS),'DSSDUMP',CLASS=S,MSGCLASS=X
//DSSDUMP  EXEC PGM=DSSDUMP,REGION=4096K
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  OPTIONS TEST
  DUMP HMVS01.**
  DUMP HMVS02.CNTL
//TAPE      DD DSN='DUMP.HMVS.USERS.DATA',
//             UNIT=(TAPE,,DEFER),DISP=(NEW,KEEP),
//             DCB=(LRECL=18448,BLKSIZE=18452,RECFM=V)
//
```

Dump a dataset, substituting new HLQ on backup set

```
//DSSDUMP  JOB (SYS),'DSSDUMP',CLASS=S,MSGCLASS=X
//DSSDUMP  EXEC PGM=DSSDUMP,REGION=4096K
//STEPLIB   DD DSN=SYSC.LINKLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  DUMP HMVS01.STUDENT.DATA
  RENAME HMVS01 HMVS02
//TAPE      DD DSN='DUMP.HMVS.USERS.DATA',
//             UNIT=(TAPE,,DEFER),DISP=(NEW,KEEP),
//             DCB=(LRECL=18448,BLKSIZE=18452,RECFM=V)
//
```

Dump two groups of datasets contained on one volume

```
//DSSDUMP  JOB (SYS),'DSSDUMP',CLASS=S,MSGCLASS=X
//DSSDUMP  EXEC PGM=DSSDUMP,REGION=4096K
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  DUMP GCC.** VOLUME(SYSCPK)
  DUMP PDPCLIB.** VOLUME(SYSCPK)
//TAPE      DD DSN='DUMP.GCCMVS',
//             UNIT=(TAPE,,DEFER),DISP=(NEW,KEEP)
//
```

DSSREST **[written by Charlie Brint, located in SYSC.LINKLIB]**

Used for restoring one or more datasets from a backup set created by DFDSS (ADRDSSU) or DSSDUMP. DSSREST was written by Charlie Brint and the source may be found on CBT Tape File #860. DSSREST requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSUT1 - defines a sequential input dataset that contains a backup set in DSS format, uncompressed.
- REPORT - defines a sequental output dataset that will contain the details of the contents of the backup set.
- JCLOUT - defines a sequential output dataset that will contain skeleton JCL that, after customization, may be used to restore one or more datasets from the backup set. This DD is optional and, if omitted, a message will be written to the output message dataset and the skeleton JCL will not be produced.
- SYSUT2 - defines a model DD that will be used for each dataset restored.

Control Statements

There are no control statements for DSSREST.

Program Parameter (PARM= on EXEC statement)

If no PARM is specified, DSSREST will read the backup set and produce a report of the datasets contained in the backup set, and, if the JCLOUT DD is supplied, will write to the JCLOUT DD a set of skeleton JCL that may be used to restore one or more of the datasets contentained in the backup set. The parameters allowed are:

DYNALLOC | DYNALLOC,prefix

If DYNALLOC is specified alone, DSSREST will attempt to dynamically restore all datasets from the backup set. If DYNALLOC is specified with a prefix, DSSREST will attempt to dynamically restore all datasets from the backup set, but substitute the supplied prefix for the high level qualifier of each dataset in the backup set.

* | single dataset name

If the parameter supplied is a single asterisk (*) or a single dataset name, the JCL is expected to include the set of model SYSUT2 DDs generated by DSSREST executed with no PARM.

If the parameter is an asterisk (*), DSSREST will attempt to restore all datasets from the backup set. There should be a SYSUT2 included for each dataset in the backup set, which DSSREST will use, in sequence, to allocate the output dataset to restore. Because the SYSUT2 DD statements are expected to be in the same sequence as the datasets contained in the backup set, it is recommended that you always use the generated skeleton JCL to restore an entire backup set. If you wish to restore only some of the datasets contained in the backup set, you should edit the skeleton JCL and comment out the SYSUT2 statements that correspond to the datasets in the backup set that you **do not wish to be restored**.

If the parameter is the name of a single dataset in the backup set, DSSREST will use the model SYSUT2 DD to restore only the single dataset from the backup set.

Examples

Read backup set, generate report and restore JCL

```
//DSSREST  JOB (SYS),'DSSREST',CLASS=S,MSGCLASS=X
//DSSREST  EXEC PGM=DSSREST,REGION=2048K,PARM=''
//SYSPRINT  DD SYSOUT=*
//REPORT    DD SYSOUT=*
//JCLOUT    DD SYSOUT=*
//SYSUT1    DD DSN='DUMP.JAY01',
//             UNIT=TAPE,VOL=SER=21121A,DISP=OLD
//
```

Restore a single dataset from backup set

```
//DSSREST  JOB (SYS),'DSSREST',CLASS=S,MSGCLASS=X
//RESTORE EXEC PGM=DSSREST,REGION=8000K,TIME=1440,
//             PARM='JAY01.CNTL'
//SYSPRINT DD  SYSOUT=*
//REPORT   DD  SYSOUT=*
//SYSUT1   DD  DISP=OLD,DSN=DUMP.JAY01,
//             UNIT=TAPE,VOL=SER=(21121A),
//             LABEL=(1,SL)
//*
//SYSUT2 DD DSN=JAY01.CLIST,                                    /*50*/
//          DISP=(,CATLG,DELETE),FREE=CLOSE,DCB=(DSORG=PO,
//          RECFM=FB,LRECL=80,BLKSIZE=19040),
//          SPACE=(CYL,(1,1,1)),
//          UNIT=3380 VOL=SER=PUB000
//SYSUT2 DD DSN=JAY01.CNTL,                                     /*51*/
//          DISP=(,CATLG,DELETE),FREE=CLOSE,DCB=(DSORG=PO,
//          RECFM=FB,LRECL=80,BLKSIZE=23440),
//          SPACE=(CYL,(12,1,1)),
//          UNIT=3380,VOL=SER=PUB000
//SYSUT2 DD DSN=JAY01.EXEC,                                     /*52*/
//          DISP=(,CATLG,DELETE),FREE=CLOSE,DCB=(DSORG=PO,
//          RECFM=VB,LRECL=255,BLKSIZE=23460),
//          SPACE=(CYL,(3,1,1)),
//          UNIT=3380 VOL=SER=PUB000
//SYSUT2 DD DSN=JAY01.LOAD,                                     /*54*/
//          DISP=(,CATLG,DELETE),FREE=CLOSE,DCB=(DSORG=PO,
//          RECFM=U,BLKSIZE=19069),
//          SPACE=(CYL,(3,1,1)),
//          UNIT=3380 VOL=SER=PUB000
//SYSUT2 DD DSN=JAY01.SOURCE,                                   /*59*/
//          DISP=(,CATLG,DELETE),FREE=CLOSE,DCB=(DSORG=PO,
//          RECFM=FB,LRECL=80,BLKSIZE=19040),
//          SPACE=(CYL,(6,1,1)),
//          UNIT=3380 VOL=SER=PUB000
//
```

Restore more than one, but not all, datasets from backup set

```
//DSSREST  JOB (SYS),'DSSREST',CLASS=S,MSGCLASS=X
//RESTORE EXEC PGM=DSSREST,REGION=8000K,TIME=1440,
//             PARM='*'
//SYSPRINT DD  SYSOUT=*
//REPORT   DD  SYSOUT=*
//SYSUT1   DD  DISP=OLD,DSN=DUMP.JAY01,
//             UNIT=TAPE,VOL=SER=(21121A),
//             LABEL=(1,SL)
//*
//*SYSUT2 DD DSN=JAY01.CLIST,                                   /*50*/
//          DISP=(,CATLG,DELETE),FREE=CLOSE,DCB=(DSORG=PO,
```

```
//          RECFM=FB,LRECL=80,BLKSIZE=19040),
//          SPACE=(CYL,(1,1,1)),
//          UNIT=3380,VOL=SER=PUB000
//SYSUT2 DD DSN=JAY01.CNTL,                                    /*51*/
//          DISP=(,CATLG,DELETE),FREE=CLOSE,DCB=(DSORG=PO,
//          RECFM=FB,LRECL=80,BLKSIZE=23440),
//          SPACE=(CYL,(12,1,1)),
//          UNIT=3380,VOL=SER=PUB000
//*SYSUT2 DD DSN=JAY01.EXEC,                                   /*52*/
//          DISP=(,CATLG,DELETE),FREE=CLOSE,DCB=(DSORG=PO,
//          RECFM=VB,LRECL=255,BLKSIZE=23460),
//          SPACE=(CYL,(3,1,1)),
//          UNIT=3380,VOL=SER=PUB000
//*SYSUT2 DD DSN=JAY01.LOAD,                                   /*54*/
//          DISP=(,CATLG,DELETE),FREE=CLOSE,DCB=(DSORG=PO,
//          RECFM=U,BLKSIZE=19069),
//          SPACE=(CYL,(3,1,1)),
//          UNIT=3380,VOL=SER=PUB000
//SYSUT2 DD DSN=JAY01.SOURCE,                                  /*59*/
//          DISP=(,CATLG,DELETE),FREE=CLOSE,DCB=(DSORG=PO,
//          RECFM=FB,LRECL=80,BLKSIZE=19040),
//          SPACE=(CYL,(6,1,1)),
//          UNIT=3380,VOL=SER=PUB000
//
```

FIXDSCB **[written by David Alan Weaver, located in SYSC.LINKLIB]**

Used for modification and repair of dataset control blocks (DSCBs). FIXDSCB was written by David Alan Weaver and the source may be found on CBT Tape File #566. FIXDSCB requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines a dataset input that contains FIXDSCB control statements.
- any DD name - defines the volume on which to find the dataset(s) for which the control blocks are to be modified.

Control Statements

```
RENAME VOLUME=volid

       ,DSNAME=current

       ,NEWNAME=new
```

The RENAME control statement alters the dataset name to the specified new name. The parameters are:

VOLUME=volid

Identifies the DASD Volume Serial number on which the dataset to be renamed exists. VOLUME may be abbreviated VOL.

DSNAME=current

Specifies the one to forty-four character name of the dataset to be modified. DSNAME may be abbreviated DSN.

NEWNAME=new

Specifies the one to forty-four character name which will be the new name of the dataset.

Note: When using this command, I have found that the resulting dataset is no longer catalogued. There are other methods available to rename a dataset, **see IEHPROGM and PDSPROGM**.

```
PROTECT VOLUME=volid

        ,DSNAME=name
```

The PROTECT control statement sets the dataset's password bits ON. The parameters are:

VOLUME=volid

Identifies the DASD Volume Serial number on which the dataset exists. VOLUME may be abbreviated VOL.

DSNAME=name

Specifies the one to forty-four character name of the dataset to be modified. DSNAME may be abbreviated DSN.

```
 SETNOPWR VOLUME=volid

          ,DSNAME=name
```

The SETHOPWR control statement sets the dataset's no password read enable bits ON. The parameters are:

VOLUME=volid

Identifies the DASD Volume Serial number on which the dataset exists. VOLUME may be abbreviated VOL.

DSNAME=name

Specifies the one to forty-four character name of the dataset to be modified. DSNAME may be abbreviated DSN.

```
UNLOCK VOLUME=volid

       ,DSNAME=name
```

The UNLOCK control statement sets the dataset's password bits OFF. The parameters are:

VOLUME=volid

Identifies the DASD Volume Serial number on which the dataset exists. VOLUME may be abbreviated VOL.

DSNAME=name

Specifies the one to forty-four character name of the dataset to be modified. DSNAME may be abbreviated DSN.

```
RENEW VOLUME=volid

      ,DSNAME=name
```

The RENEW control statement resets the creation date for the specified dataset to the current date. The parameters are:

VOLUME=volid

Identifies the DASD Volume Serial number on which the dataset exists. VOLUME may be abbreviated VOL.

DSNAME=name

Specifies the one to forty-four character name of the dataset to be modified. DSNAME may be abbreviated DSN.

```
EXPIRE VOLUME=volid

       ,DSNAME=name
```

The EXPIRE control statement swaps creation date and expiration date for the specified dataset. The parameters are:

VOLUME=volid

Identifies the DASD Volume Serial number on which the dataset exists. VOLUME may be abbreviated VOL.

DSNAME=name

Specifies the one to forty-four character name of the dataset to be modified. DSNAME may be abbreviated DSN.

```
EXTEND VOLUME=volid

       ,DSNAME=name
```

The EXTEND control statement set the expiration date to 99:365 for the specified dataset. The parameters are:

VOLUME=volid

Identifies the DASD Volume Serial number on which the dataset exists. VOLUME may be abbreviated VOL.

DSNAME=name

Specifies the one to forty-four character name of the dataset to be modified. DSNAME may be abbreviated DSN.

```
ZEROEXPD VOLUME=volid

         ,DSNAME=name
```

The ZEROEXPD control stateement sets the expiration date to 00:000 for the specified dataset. The parameters are:

VOLUME=volid

Identifies the DASD Volume Serial number on which the dataset exists. VOLUME may be abbreviated VOL.

DSNAME=name

Specifies the one to forty-four character name of the dataset to be modified. DSNAME may be abbreviated DSN.

```
SCRATCH VOLUME=volid

        ,DSNAME=name
```

The SCRATCH control statement deletes the specified dataset. The parameters are:

VOLUME=volid

Identifies the DASD Volume Serial number on which the dataset exists. VOLUME may be abbreviated VOL.

DSNAME=name

Specifies the one to forty-four character name of the dataset to be deleted. DSNAME may be abbreviated DSN.

Note: When using this command, I have found that the scratched dataset remains in the catalog. There are other methods available to scratch a dataset, **see IEHPROGM and PDSPROGM**.

```
NAME VOLUME=volid

     ,DSNAME=name
```

The NAME control statement provides the dataset information for immediately following subcommand(s). The parameters are:

VOLUME=volid

Identifies the DASD Volume Serial number on which the dataset exists. VOLUME may be abbreviated VOL.

DSNAME=name

Specifies the one to forty-four character name of the dataset to be modified. DSNAME may be abbreviated DSN.

RECFM=xxxxx

The RECFM subcommand sets the RECFM to the value specified as xxxxx. The value specified must be one of: UT, UA, UM, UTA, UTM, F, FB, FS, FT, FBS, FBT, FBST, FA, FBA, FSA, FTA, FBSA, FBTA, FBSTA, FM, FBM, FSM, FTM, FBSM, FBTM, FBSTM, V, VB, VS, VT, VBS, VBT, VBST, VA, VBA, VSA, VTA, VBSA, VBTA, VBSTA, VM, VBM, VSM, VTM, VBSM, VBTM, or VBSTM.

LRECL=xxxxx

The LRECL subcommand sets the LRECL to the value specified as xxxxx.

BLKSIZE=xxxxx

The BLKSIZE subcommand sets the BLKSIZE to the value specified as xxxxx.

DSORG=xxx

The DSORG subcommand sets the DSORG to the value specified as xxx. The value specified must be one of: PSU, DA, DAU, IS, ISU, PO, or POU.

KEYL=xxx

The KEYL subcommand sets the key length to the value specified as xxx.

RKP=xxx

The RKP subcommand sets the relative key position to the value specified as xxx.

OPTCODE=x

The OPTCODE subcommand sets the OPTCODE to the value specified as x. Consult JCL manual for possible values.

If a particular subcommand is entered more than once, the last occurrence of the subcommand will be the one used. If an error occurs during processing of any subcommand, the entire subcommand set for the current NAME command will be discarded.

Examples

Extend expiration date for dataset

```
//FIXDSCB  JOB (1),'FIXDSCB',CLASS=A,MSGCLASS=X
//FIXDSCB  EXEC PGM=FIXDSCB
//DD01     DD  UNIT=SYSDA,VOL=SER=MVS381,DISP=OLD
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  EXTEND VOL=MVS381,DSN=HMVS01.SOURCE.COBOL
//
```

Modify RECFM, LRECL, BLKSIZE

```
//FIXDSCB  JOB (1),'FIXDSCB',CLASS=A,MSGCLASS=X
//FIXDSCB  EXEC PGM=FIXDSCB
//DD01     DD  UNIT=SYSDA,VOL=SER=MVS381,DISP=OLD
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  NAME VOL=MVS381,DSN=HMVS01.COBOL.LISTINGS
    RECFM=FBA
    LRECL=121
    BLKSIZE=23474
//
```

ICKDSF **[supplied by IBM, located in SYS1.LINKLIB]**

Used to perform various operations on direct-access storage devices. For the scope of this page, I will only be covering the use of ICKDSF to initialize an offline direct-access storage volume so that it can be used in an MVS system.

Note that on MVS 3.8j systems built following my instructions/tutorial, there are two versions of ICKDSF installed and available for use. The version installed with the base Operating System resides in SYS1.LINKLIB and is Version 6. The version installed in SYS2.LINKLIB is Version 13. ICKDSF requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines a sequential input dataset that contains ICKDSF control statements.
- any DD name - defines the input dataset containing IPL text records; required only when the IPLDD parameter is used.

Control Statements

```
INIT UNITADDRESS(address) -
```

```
{VERIFY(volser) | NOVERIFY} -

{PURGE | NOPURGE} -

[OWNERID(owner)] -

VOLID(volser) -

VTOC(cylinder,track[,extent]) -

[IPLDD(ddname)]
```

The INIT control statement prepares a DASD volume for use by examining the mounted (offline) volume to determine available space, optionally writes IPL bootstrap records as records 1 and 2, writes a volume label as record 3, and reserves tracks for the Volume Table of Contents at the location specified by the user and for the number of tracks specified. If no location is specified, tracks are reserved at the default location. Alternate tracks are checked for defective-track flags. The parameters are:

UNITADDRESS(address)

Specifies the physical device address of the DASD to be initialized. The keyword UNITADDRESS may be abbreviated to UNIT. The device must be on a channel that is online; if the channel is offline the program may enter a nonterminating wait state. The address may be specified as either a three or four digit hexadecimal address.

VERIFY(volser) | NOVERIFY

VERIFY specifies the current Volume Serial number of the DASD mounted at the address specified, which will be verified before ICKDSF proceeds. NOVERIFY specifies that no checking will be performed. The keyword VERIFY may be abbreviated to VFY; NOVERIFY may be abbreviated to NVFY.

Note: It is a sound practice to create new DASD volumes with the Hercules' dasdinit utility giving a new volume a known Volume Serial number, such as 111111, which may be specified in the VERIFY parameter to ensure that the proper address has been given for ICKDSF to initialize.

PURGE | NOPURGE

PURGE specifies that unexpired, or VSAM, or password-protected, or RACF-protected datasets found on the volume are to be written over. NOPURGE specifies that if these types of datasets exist on the volume, you do not want them written over. The keyword PURGE may be abbreviated to PRG; NOPURGE may be abbreviated to NPRG.

OWNERID(owner)

Specifies an optional one to fourteen character string that will be written in the volume label. The keyword OWNERID may be abbreviated OWNER.

VOLID(volser)

Specifies the volume serial number to be written in the volume label. For volser, substitute one to six alphameric characters for the volume serial number. If fewer than six characters are specified, the serial is left-justified, and the remainder of the field is padded with blanks (X'40'). The VOLID of each DASD placed online for MVS to access must be unique.

VTOC(cylinder,track[,extent])

Specifies the location and size for the Volume Table of Contents. For cylinder,track, specify one to four decimal (n) or hexadecimal (X'n') numbers to identify the cylinder and one to four decimal (n) or hexadecimal (X'n') numbers to identify the track where the volume table of contents is to be placed. For extent, specify one to five decimal (n) or hexadecimal (X'n') numbers for the number of tracks that are to be reserved the volume table of contents.

Defaults: The default for cylinder,track is 0,1. The default for extent is one track. A VTOC cannot be placed at cylinder 0, track 0.

IPLDD(ddname)

An optional parameter which allows you to supply an IPL program to be written on the volume during initialization. ICKDSF supplies an IPL bootstrap which is written on the volume during initialization together with the IPL text you supply.

Examples

Initialize 2314, 3330, 3340, 3350 DASD

```
//ICKDSF   JOB 1,ICKDSF,CLASS=A,MSGCLASS=X
//* ************************************************************ *
//* * USE FOR 3350 | 3330 | 3340 | 2314                          *
//* ************************************************************ *
//*
//ICKDSF   EXEC PGM=ICKDSF,REGION=4096K
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  INIT UNITADDRESS(254) VERIFY(111111) -
       VOLID(134134) OWNER(HERCULES) -
       VTOC(0,1,15)
//
```

Initialize 3375, 3380, 3390 DASD

```
////ICKDSF13 JOB 1,ICKDSF,CLASS=S,MSGCLASS=X
//* ************************************************************ *
//* * USE FOR 3390 | 3380 | 3375                                  *
//* ************************************************************ *
//ICKDSF13 EXEC PGM=ICKDSF13,REGION=4096K
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  INIT UNIT(187) -
       VERIFY(SVD001) -
       OWNER(HERCULES) -
       VOLID(SVD001) -
       VTOC(0,1,15)
//
```

Source: GC35-0033-5 Device Support Facilities User's Guide and Reference 4.0

IEBCOMPR [supplied by IBM, located in SYS1.LINKLIB]

Used to compare two sequential datasets or two partitioned datasets at the logical record level to verify a backup copy. Fixed, variable, or undefined records from blocked or unblocked datasets or members can also be compared. **Note: This is a very old utility and of limited functionality. I would suggest taking a look at CMPRSEQ and PDSMATCH.**

Two sequential datasets are considered to be identical, if: 1) the datasets contain the same number of records, and, 2) corresponding records and keys are identical. If these conditions are not met, an unequal comparison results. If records are unequal, the record and block numbers, the names of the DD statements that define the datasets, and the unequal records are listed in a message dataset. Ten successive unequal comparisons terminate the job step unless a user routine is provided to handle error conditions.

Two partitioned datasets are considered to be identical if: 1) corresponding members contain the same number of records, 2) note lists are in the same position within corresponding members, and 3) corresponding records and keys are identical. If these conditions are not met, an unequal comparison results. If records are unequal, the record and block numbers, the names of the DD statements that define the datasets, and the unequal records are listed in a message dataset. After ten successive unequal comparisons, processing continues with the next member unless a user routine is provided to handle error conditions. Partitioned datasets can be compared only if all the names in one or both of the directories have counterpart entries in the other directory. The comparison is made on members identified by these entries and corresponding user data.

IEBCOMPR requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines a sequential input dataset that contains IEBCOMPR control statements.
- SYSUT1 - defines the first of the input datasets to be compared
- SYSUT2 - defines the second of the input datasets to be compared

Control Statements

```
COMPARE TYPEORG={PS | PO}
```

The COMPARE control statement specifies the organization of the two datasets to be compared. The COMPARE control statement, if included, must be the first utility control statement. A COMPARE control statement is required if the EXITS or LABELS statement is used or if the input datasets are partitioned datasets. The values that may be specified are:

- PS - specifies that the input datasets to be compared are Physical Sequential organization.
- PO - specifies that the input datasets to be compared are Partitioned organization.

```
LABELS DATA={YES | NO | ALL | ONLY}
```

The LABELS control statement specifies whether user labels are to be treated as data by IEBCOMPR.

Note: LABELS DATA=NO must be specified to make standard user label (SUL) exits inactive when input/output datasets with nonstandard labels (NSL) are the be processed. If more than one valid LABELS statement is included, all but the last LABEL statement are ignored. The values that may be specified are:

- YES - specifies that any user labels that are not rejected by a user's label processing routine are to be treated as data. Processing of labels as data stops in compliance with standard return codes.
- NO - specifies that user labels are not to be treated as data.
- ALL - specifies that user labels are to be treated as data regardless of any return code. A return code of 16 causes IEBCOMPR to complete processing of the remainder of the group of user labels and to terminate the job step.
- ONLY - specifies that only user header labels are to be treated as data. User header labels are processed as data regardless of any return code. The job terminates upon return from the OPEN routine.

```
EXITS [INHDR=routinename]

      [,INTLR=routinename]

      [,ERROR=routinename]

      [,PRECOMP=PCRTN]
```

The EXITS control statement specifies the name of the routine to receive control after each error condition. The parameters are:

ERROR=routinename

Specifies the symbolic name of a routine that is to receive control after each unequal comparison for error handling. If this parameter is omitted and ten consecutive unequal comparisons occur while IEBCOMPR is comparing sequential datasets, processing is terminated; if the input datasets are partitioned, processing continues with the next member.

INHDR=routinename

Specifies the symbolic name of a routine that processes user input header labels.

INTLR=routinename

Specifies the symbolic name of a routine that processes user input trailer labels.

PRECOMP=routinename

Specifies the symbolic name of a routine that processes logical records (physical blocks in the case of VS or VBS records longer than 32K bytes) from either or both of the input datasets before they are
compared.

Examples

Compare two sequential datasets

```
//IEBCOMPR JOB (1),IEBCOMPR,CLASS=A,MSGCLASS=X
//IEBCOMPR EXEC PGM=IEBCOMPR
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=DATASET1,UNIT=3330,VOL=SER=D10040,
//DISP=SHR
//SYSUT2   DD DSN=DATASET2,UNIT=3330,VOL=SER=D10040,
//  DISP=SHR
//SYSIN    DD *
COMPARE TYPORG=PS
/*
//
```

Compare two partitioned datasets

```
//IEBCOMPR JOB (1),IEBCOMPR,CLASS=A,MSGCLASS=X
//IEBCOMPR EXEC PGM=IEBCOMPR
//SYSPRINT DD SYSOUT=*
//SYSUTl   DD DSN=PDSSET1,UNIT=DISK,DISP=SHR,
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000),
//            VOLUME=SER=111112
//SYSUT2   DD DSN=PDSSET2,UNIT=DISK,DISP=SHR,
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000),
//            VOLUME=SER=111113
//SYSIN    DD *
  COMPARE TYPORG=PO
/*
//
```

Source: GC26-3902-1 OS/VS2 MVS Utilities Release 3.8

IEBCOPY **[supplied by IBM, located in SYS1.LINKLIB]**

Used to copy one or more partitioned datasets or to merge partitioned datasets. A partitioned dataset which is copied to a sequential dataset is said to be unloaded. The sequential dataset created by an unload operation can be copied to any direct access device. When one or more datasets created by an unload operation are used to re-create a partitioned dataset, this is called a load operation. An individual member, or members, of a partitioned or unloaded dataset can be selected for, or excluded from, a copy, unload, or load process.

IEBCOPY can be used to:

- Create a backup copy of a partitioned dataset.
- Copy one or more datasets per copy operation.
- Copy one partitioned dataset to a sequential dataset (unload).
- Copy one or more datasets created by an unload operation to any direct access device (load).
- Select members from a dataset to be copied, unloaded, or loaded.
- Replace identically named members on datasets (except when unloading).
- Replace selected dataset members.
- Rename selected members.
- Exclude members from a dataset to be copied, unloaded, or loaded.
- Compress partitioned datasets in place (except when the dataset is an unloaded dataset).
- Merge datasets (except when unloading).
- Re-create a dataset that has exhausted its primary, secondary, or directory space allocation.

A compressed dataset is one that does not contain embedded, unused space. After copying or loading one or more input partitioned datasets to a new output partitioned dataset (by means of a selective, exclusive, or full copy or load that does not involve replacing members), the output partitioned dataset contains no embedded, unused space. To make

unused space available, either the entire dataset must be scratched or it must be compressed in place. A compressed version can be created by specifying the same dataset for both the input and the output parameters in a full copy step. **A backup copy of the partitioned dataset to be compressed in place should be kept until successful completion of an in-place compression is indicated (by an end-of-job message and a return code of 00).** An in-place compression does not release extents assigned to the dataset. Selection, exclusion, or renaming of selected members cannot be done during the compression of a partitioned dataset.

In addition, IEBCOPY automatically lists the number of unused directory blocks and the number of unused tracks available for member records in the output partitioned dataset. If LIST=NO is coded the names of copied, unloaded, or loaded members listed by the input dataset are suppressed.

IEBCOPY requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines a sequential input dataset that contains IEBCOPY control statements.
- SYSUT1 - defines the first of the input datasets to be compared
- SYSUT2 - defines the second of the input datasets to be compared
- SYSUT3 - defines a temporary work file
- SYSUT4 - defines a temporary work file

Control Statements

All keywords on the control statements may be abbreviated to a single letter: COPY to C; INDD to I; OUTDD to O; SELECT to S; EXCLUDE to E.

```
COPYOUTDD=ddname

    ,INDD={ddname | (ddname1[,ddname2]...)}

    [,LIST=NO]
```

The COPY control statement specifies the DD names to be used for a copy operation and allows suppression of list. The parameters are:

OUTDD=ddname,

Specifies the DD name for the output dataset.

INDD=ddname | INDD=(ddname1[,ddname2]...

Specifies one or more DD names for the input dataset(s). If more than one ddname is specified, they must be enclosed in parentheses and separated by commas; the input datasets specified in the list are processed in the order specified in the list.

In the case of multiple input datasets, there can arise the event of duplicate member names. If this happens, a subsequent member encountered with the same name as a member already copied to the output dataset will not replace the member that has already been copied to the output dataset, unless the replace option is specified. To replace any duplicate members encountered, enclose the ddname for which the replace option is to be used in an additional pair of parentheses and follow the

ddname with a comma followed by R:

INDD=(ddname1,(ddname2,R),ddname3)

will result in any members in ddname2 that have duplicates in ddname1 to replace the members already copied from ddname1 in the output dataset.

[,LIST=NO]

Specifies that the list of copied members, as well as unused tracks and directory blocks in the output dataset, is to be suppressed.

```
SELECTMEMBER={name | (name1[,name2]...) | ((name,newname)[,(name2,newname2)...)}
```

The SELECT control statement specifies specific members to be copied from the input dataset(s). If more than one member name is specified, they must be enclosed in parentheses and separated by commas. Multiple SELECT statements may be specified, in which case the subsequent statements are processed as a continuation of the first. If the optional newname is specified for a member, the original member is copied to the output dataset using the newname supplied.

A selected member will not replace an identically named member already copied into the output dataset unless the replace option is specified. To cause any selected member to replace an identically named member, enclose the member name in an additional pair of parentheses and follow the member name with two commas followed by R:

SELECT MEMBER=(name1,(name2,,R),name3,(name4,newname4,R))

will result in name2, and name4 after renaming to newname4, replacing any duplicate already copied into the output dataset.

```
EXCLUDE MEMBER={name | (name1[,name2]...)}
```

The EXCLUDE control statement specifies specific members to be excluded from being copied to the output dataset.

Note: SELECT and EXCLUDE statements may not appear together in the same COPY operation.

Examples

Copy partitioned dataset from one DASD to another DASD, including all members

```
//IEBCOPY JOB (1),IEBCOPY,CLASS=A,MSGCLASS=X
//IEBCOPY EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//FROM     DD DSN=FROMLIB,UNIT=3330,VOL=SER=D10040,
// DISP=OLD
//TO       DD DSN=TOLIB,UNIT=3330,VOL=SER=D10080,
// DISP=(NEW,KEEP),SPACE=(TRK,(400,,50))
//SYSUT3   DD UNIT=3330,SPACE=(TRK,5)
//SYSUT4   DD UNIT=3330,SPACE=(TRK,5)
```

```
//SYSIN    DD *
COPY INDD=FROM,OUTDD=TO
/*
//
```

Copy partitioned dataset from one DASD to another DASD, selecting specific members

```
//IEBCOPY JOB (1),IEBCOPY,CLASS=A,MSGCLASS=X
//IEBCOPY EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//FROM     DD DSN=FROMLIB,UNIT=3330,VOL=SER=D10040,
// DISP=OLD
//TO       DD DSN=TOLIB,UNIT=3330,VOL=SER=D10080,
// DISP=(NEW,KEEP),SPACE=(TRK,(400,,50))
//SYSUT3   DD UNIT=3330,SPACE=(TRK,5)
//SYSUT4   DD UNIT=3330,SPACE=(TRK,5)
//SYSIN    DD *
COPY INDD=FROM,OUTDD=TO
  SELECT MEMBER=(PROGA,PROGB)
/*
//
```

Copy members from two partitioned datasets, creating a new partitioned dataset, replacing any members with the same name from dataset 2

```
//IEBCOPY JOB (1),IEBCOPY,CLASS=A,MSGCLASS=X
//IEBCOPY EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//FROM1    DD DSN=FROMLIB1,UNIT=3330,VOL=SER=D10040,
// DISP=OLD
//FROM2    DD DSN=FROMLIB2,UNIT=3330,VOL=SER=D10041,
// DISP=OLD
//TO       DD DSN=TOLIB,UNIT=3330,VOL=SER=D10080,
// DISP=(NEW,KEEP),SPACE=(TRK,(400,,50))
//SYSUT3   DD UNIT=3330,SPACE=(TRK,5)
//SYSUT4   DD UNIT=3330,SPACE=(TRK,5)
//SYSIN    DD *
COPY INDD=(FROM1,(FROM2,R)),OUTDD=TO
/*
//
```

Compress partitioned dataset

```
//COMPRESS JOB (1),COMPRESS,CLASS=A,MSGCLASS=X
//IEBCOPY  EXEC PGM=IEBCOPY,REGION=4M
//SYSPRINT  DD SYSOUT=*
//LIB       DD DSN=LIBFILE,UNIT=3330,VOL=SER=D10040
// DISP=OLD
//SYSUT3    DD UNIT=3330,SPACE=(TRK,5)
//SYSUT4    DD UNIT=3330,SPACE=(TRK,5)
//SYSIN     DD *
COPY INDD=LIB,OUTDD=LIB
/*
//
```

Unload a partitioned dataset to tape

```
//UNLOAD   JOB (1),UNLOAD,CLASS=A,MSGCLASS=X
//IEBCOPY  EXEC PGM=IEBCOPY
//SYSPRINT  DD SYSOUT=*
//LIB       DD DSN=LIBFILE,UNIT=3330,VOL=SER=D10040
// DISP=OLD
//TAPE      DD DSN=LIBFILE.BACKUP,UNIT=TAPE,VOL=SER=T10002,
//             LABEL=(1,SL),DISP=(NEW,KEEP)
//SYSUT3    DD UNIT=3330,SPACE=(TRK,5)
//SYSUT4    DD UNIT=3330,SPACE=(TRK,5)
```

```
//SYSIN     DD *
COPY INDD=LIB,OUTDD=TAPE
/*
//
```

Reload some members of an unloaded partitioned dataset

```
//RELOAD   JOB (1),RELOAD,CLASS=A,MSGCLASS=X
//IEBCOPY  EXEC PGM=IEBCOPY,REGION=1M
//SYSPRINT  DD SYSOUT=*
//TAPE      DD DISP=(OLD,KEEP),UNIT=TAPE,LABEL=(1,SL),
//             VOL=SER=T10002,DSN=LIBFILE.BACKUP
//LIB       DD DISP=(,CATLG,DELETE),DSN=SYSP.SOURCE.COBOL,
//             UNIT=SYSDA,VOL=SER=SYSP01,
//             SPACE=(TRK,(15,15,5))
//SYSUT3    DD UNIT=3330,SPACE=(TRK,5)
//SYSUT4    DD UNIT=3330,SPACE=(TRK,5)
//SYSIN    DD  *
  COPY INDD=TAPE,OUTDD=LIB
  SELECT MEMBER=(PROGV,PROGL)
/*
//
```

Source: GC26-3902-1 OS/VS2 MVS Utilities Release 3.8

IEBGENER **[supplied by IBM, located in SYS1.LINKLIB]**

Used to copy a sequential dataset, or a member of a partitioned dataset, to a sequential dataset or a member of a partitioned dataset.

> Note: At the January 2022 rewrite, for this utility, as with several others, I was using IBM manual GC26-3092-1 OS/VS2 MVS Utilities Release 3.8 for the primary source. That manual includes the definitions of control statements for using IEBGENER for much more advanced dataset manipulation than simply copying. However, in many decades of using IEBGENER, I have only once seen the use of SYSIN control statements to do any manipulation of the data moved between datasets. The vast majority of cases where I have seen IEBGENER used was simply for copying a dataset or member from one location to another, with the SYSIN DD provided as DUMMY. IEBGENER is excellent at performing that task. **I would also suggest taking a look at SYSREPRO.**

IEBGENER requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines an optional sequential input dataset that may contains IEBGENER control statements.
- SYSUT1 - defines the input dataset to be copied
- SYSUT2 - defines the output dataset

Control Statements

```
GENERATE [MAXNAME=n]

         [,MAXFLDS=n]

         [,MAXGPS=n]
```

```
      [,MAXLITS=n]
```

The GENERATE control statement specifies the number of member names and alias names, record identifiers, literals, and editing information contained in the control dataset. The parameters are:

MAXNAME=n

Specifies a number that is no less than the total number of member names as aliases appearing in subsequent MEMBER statements. MAXNAME is required if there are one or more MEMBER statements.

MAXFLDS=n

Specifies a number that is no less than the total number of FIELD parameter appearing in subsequent RECORD statements. MAXFLDS is required if there are any FIELD parameters in subsequent RECORD statements.

MAXGPS=n

Specifies a number that is no less than the total number of IDENT parameters appearing in subsequent RECORD statements. MAXGPS is required if there are any IDENT parameters in subsequent RECORD statements.

MAXLITS=n

Specifies a number that is no less than the total number of characters contained in the FIELD literals of subsequent RECORD statements. MAXLITS is required if the FIELD parameters of subsequent RECORD statements contain literals. MAXLITS does not pertain to literals used in IDENT parameters.

```
EXITS [INHDR=routinename]

      [,OUTHDR=routinename]

      [,INTLR=routinename]

      [,OUTTLR=routinename]

      [,KEY=routinename]

      [,DATA=routinename]

      [,IOERROR=routinename]

      [,TOTAL=(routinename, size)]
```

The EXITS control statement specifies that user routines are provided. The parameters are:

INHDR=routinename

Specifies the symbolic name of a routine that processes user input header labels.

OUTHDR=routinename

Specifies the symbolic name of a routine that creates user output header labels. OUTHDR is ignored if the output dataset is partitioned.

INTLR=routinename

Specifies the symbolic name of a routine that processes user input trailer labels.

OUTTLR=routinename

Specifies the symbolic name of a routine that processes user output trailer labels. OUTTLR is ignored if the output dataset is partitioned.

KEY=routinename

Specifies the symbolic name of a routine that creates the output record key. (This routine does not receive control when a dataset consisting of VS or VBS type records is processed because no processing of keys is permitted for this type of data.)

DATA=routinename

Specifies the symbolic name of a routine that modifies the physical record (logical record for VS or VBS type records) before its processed by IEBGENER.

IOERROR=routinename

Specifies the symbolic name of a routine that handles permanent input/output error conditions.

TOTAL=(routinename,size)

Specifies that exits to a user's routine are to be provided prior to writing each record. The keyword OPTCD=T must be specified for the SYSUT2 DD statement. TOTAL is valid only when the utility is used to process sequential datasets. These values must be coded:

- routinename - Specifies the name of a user-supplied totaling routine.
- size - Specifies the number of bytes needed to contain totals, counters, pointers, etc.

```
LABELS[DATA={YES | NO | ALL | ONLY | INPUT}]
```

The LABELS control statement specifies whether user labels are to be treated as data. The values that may be specified for the DATA operand are:

- YES - Specifies that any user labels that are not rejected by a user's label processing routine are to be treated as data. Processing of labels as data ends in compliance with standard return codes.
- NO - Specifies that user labels are not to be treated as data.
- ALL - Specifies that user labels in the group currently being processed are to be treated as data regardless of any return code. A return code of 16 causes IEBGENER to complete processing the

remainder of the group of user labels and to terminate the job step.

- ONLY - Specifies that only user header labels are to be treated as data. User header labels are processed as data regardless of any return code. The job terminates upon return from the OPEN routine.
- INPUT - Specifies that user labels for the output dataset are supplied as 80-byte input records in the data portion of SYSIN. The number of input records that should be treated as user labels must be identified by a RECORD statement.

```
MEMBER NAME={(name[,alias]...)}
```

The MEMBER control statement specifies the member name and alias(es) of a member of a partitioned dataset to be created. The parameter is:

NAME=(name[,alias]...)

Specifies a member name followed by a list of its aliases. If only one name appears in the statement, it need not be enclosed in parentheses.

```
RECORD[IDENT=(length,'name',input-location)]

      [,FIELD=([length],[{input-location | 'literal'}]

      ,[conversion]

      [,output-location])

      [,FIELD=...]]

      [,LABELS=n]
```

The RECORD control statement defines a record group to be processed and supplies editing information. The parameters are:

IDENT=(length,'name',input-location)

Identifies that last record of the input group to which the FIELD parameters of MEMBER statement applies. If the RECORD control statement is not followed by additional RECORD or MEMBER control statements, IDENT also defines the last record to be processed. These values can be coded:

- length - Specifies the length (in bytes) of the identifying name. The length cannot exceed eight characters.
- 'name' - Specifies the exact literal that identifies the last input record of a record group. Default: If no match for name is found, the remainder of the input data considered to be in one record group; subsequent RECORD and MEMBER statements are ignored.
- input-location - Specifies the starting location of the field that contains the identifying name in the input records.

Default: If IDENT is omitted, the remainder of the input data is considered to be in one record group; subsequent RECORD and MEMBER control statements are ignored.

FIELD=([length],[{input-location | 'literal'}],[conversion],[output-location])[,FIELD=...]

Specifies field-processing and editing information. Only the contents of specified fields in the input record is copied to the output record, that is, any field in the output record that is not specified will contain meaningless information. The values that can be coded are:

- length - Specifies the length (in bytes) of the input field or literal to be processed. If length is not specified, a length of 80 bytes is assumed. If a literal is to be processed, a length of 40 bytes or less must be specified.
- input-location - Specifies the starting byte of the field to be processed. Default: Byte 1 is assumed.
- 'literal' - Specifies a literal (maximum length of 40 bytes) to be placed in the specified output location. If a literal contains apostrophes, each apostrophe must be written as two consecutive apostrophes.
- conversion - Specifies a two-byte code that indicates the type of conversion to be performed on this field. If no conversion is specified, the field is moved to the output area without change. The values that can be coded are:
    - PZ - Specifies that data (packed decimal) is to be converted to unpacked decimal data.
    - ZP - Specifies that data (unpacked decimal) is to be converted to packed decimal data.
    - HE - Specifies that data (H-set BCD) is to be converted to EBCDIC.
- output-location - Specifies the starting location of this field in the output records. If conversion is specified in FIELD, the following restrictions apply:

    For PZ-type (packed-to-unpacked) conversion is impossible for packed decimal records longer than 16K bytes.

    For ZP-type (unpacked-to-packed) conversion, the normal 32K-type maximum applies.

    When the ZP parameter is specified, the conversion is performed in place. The original unpacked field is replaced by the new packed field. Therefore, the ZP parameter must be omitted from subsequent references to that field. If the field is needed in its original unpacked form, it must be referenced prior to the use of the ZP parameter.

    If conversion is specified in the FIELD parameter, the length of the output record can be calculated for each conversion specification. When L is equal to the length of the input record, the calculation is made, as follows:

        For a PZ (packed-to-unpacked) specification, 2L-1.

        For a ZP (unpacked-to-packed) specification, (L/2) + C. If L is an odd number, C is 1/2; if L is an even number, C is 1.

        For an HE (H-set BCDIC to EBCDIC) specification, L.

    Default: Byte 1 is assumed.

If both output header labels and output trailer labels are to be contained in the SYSIN dataset, the user must include one RECORD statement (including the LABELS parameter), indicating the number of input records to be treated as user labels, for header labels and one for trailer labels. The first such RECORD statement indicates the number of user header labels; the second indicates the number of user trailer labels. If only output trailer labels are included in the SYSIN dataset, a RECORD statement must be included to indicate that there are no output

header labels in the SYSIN dataset (LABELS=O). This statement must precede the RECORD LABELS=n statement which signals the start of trailer label input records.

LABELS=n

An optional parameter that indicates the number of records in the SYSIN dataset to be treated as user labels. The number n, which is a number from 1 to 8, must specify the exact number of label records that follow the RECORD statement. If this parameter is included, DATA=INPUT must be coded on a LABELS statement before it in the input stream.

Examples

Copying tape to tape

```
//DUPTAPE JOB (1),DUPTAPE,CLASS=A,MSGCLASS=X
//DUPTAPE EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=TAPE,DSN=TAPE.FILE,
//            VOL=SER=004000,DISP=OLD,LABEL=(1,SL)
//SYSUT2   DD UNIT=TAPE,DSN=TAPE.FILE,
//            DISP=(NEW,KEEP),LABEL=(1,SL),
//DCB=BLKSIZE=80
//SYSIN DD DUMMY
//
```

Tape to print

```
//PRNTTAPE JOB (1),PRNTTAPE,CLASS=A,MSGCLASS=X
//PRNTTAPE EXEC PGM=IEBGENER
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD UNIT=TAPE,DSN=TAPE.FILE,
//             VOL=SER=004000,LABEL=(1,SL),DISP=OLD
//SYSUT2 DD SYSOUT=A,DCB=BLKSIZE=80
//SYSIN DD DUMMY
//
```

Card to tape

```
//CARDTAPE JOB (1),CARDTAPE,CLASS=A,MSGCLASS=X
//CARDTAPE EXEC PGM=IEBGENER
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD *

 [card file to be put on tape goes here]

/*
//SYSUT2    DD UNIT=TAPE,DSN=TAPE.FILE,
//             DISP=(NEW,KEEP),LABEL=(1,SL),
// DCB=BLKSIZE=80
//SYSIN DD DUMMY
//
```

Source: GC26-3902-1 OS/VS2 MVS Utilities Release 3.8

IEBISAM **[supplied by IBM, located in SYS1.LINKLIB]**

Used to:

- copy an indexed sequential (ISAM) dataset from one DASD volume to another,
- copy an ISAM dataset into a sequential dataset on DASD or tape image (unloading),
- create an ISAM dataset from an unloaded dataset,
- print an ISAM dataset.

IEBISAM requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSUT1 - defines the input dataset
- SYSUT2 - defines the output dataset

Control Statements

There are no control statements.

Program Parameter (PARM= on EXEC statement)

The PARM parameter on the EXEC statement is used to control the execution of IEBISAM. The parameters are:

COPY

Specifies the input (ISAM) dataset is to be copied, creating an ISAM dataset. The SYSUT2 DD statement must include a primary space allocation that is sufficient to accommodate records that were contained in overflow areas in the original indexed sequential dataset. New overflow areas can be specified when the dataset is copied.

UNLOAD

Specifies the input (ISAM) dataset is to be copied to a sequential dataset, creating a portable copy of the data records that may subsequently be reloaded to recreate the ISAM dataset. Regardless of the data record size in the ISAM dataset, the LRECL of the output dataset is always 80 bytes. If a block size is specified for the output dataset, it must be a multiple of 80 bytes.

LOAD

Specifies the input dataset (created by IEBISAM with the UNLOAD function) is to be used to recreate an ISAM dataset. The input dataset, defined by the SYSUT1 DD statement, must have a record size of 80 bytes and the block size must be a multiple of 80 bytes. The SYSUT2 DD statement must include a primary space allocation that is sufficient to accommodate records that were contained in overflow areas in the original indexed sequential dataset. If new overflow areas are desired, they must be specified when the dataset is loaded.

PRINTL [,N]

Specifies the records of the input ISAM dataset are to be printed. If the device defined by the SYSUT2 DD statement is a printer, the specified BLKSIZE must be equal to or less than the physical printer size; that is 121, 133, or 145 bytes. If BLKSIZE is not specified, 121 bytes is assumed. LRECL (or BLKSIZE when no LRECL was specified) must be between 55 and 255 bytes. PRINTL specifies a print operation in which each record is converted to hexadecimal before printing. The N is an optional value that specifies that

records are not to be converted to hexadecimal before printing.

EXIT=routinename

Optionally specifies the name of an exit routine that is to receive control before each record is printed. If a user routine is supplied for a PRINTL operation, IEBISAM issues a LOAD macro instruction to make the user routine available. A BALR 14,15 instruction is subsequently used to give control to the routine. When the user routine receives control, register 0 contains a pointer to a record heading buffer; register 1 contains
a pointer to an input record buffer.

Examples

Copy ISAM dataset blocking data records

```
//IEBISAMC JOB (1),'IEBISAM-COPY',
//             CLASS=A,MSGCLASS=X,REGION=2M,NOTIFY=&SYSUID
//IEBISAM EXEC PGM=IEBISAM,PARM='COPY'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=OLD,DSN=JAY01.ISAM.MASTER,DCB=DSORG=IS
//SYSUT2   DD DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,VOL=SER=PUB000,
//            DSN=JAY01.ISAM.MASTER.TWO,SPACE=(CYL,(5,5)),
//            DCB=(DSORG=IS,RECFM=FB,LRECL=200,BLKSIZE=23400,
//            KEYLEN=7,RKP=1,CYLOFL=2,NTM=4,OPTCD=LMY)
//
```

Copy ISAM dataset to portable copy (unloaded)

```
//IEBISAMU JOB (1),'IEBISAM-UNLOAD',
//             CLASS=A,MSGCLASS=X,REGION=2M,NOTIFY=&SYSUID
//IEBISAM EXEC PGM=IEBISAM,PARM='UNLOAD'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=SHR,DSN=JAY01.ISAM.MASTER,DCB=DSORG=IS
//SYSUT2   DD DISP=(NEW,KEEP),UNIT=TAPE,VOL=SER=21305B,
//            DSN=JAY01.ISAM.MASTER.BACKUP,LABEL=(1,SL),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=23440)
//
```

Reload ISAM dataset from unloaded copy

```
//IEBISAML JOB (1),'IEBISAM-RELOAD',
//             CLASS=A,MSGCLASS=X,REGION=2M,NOTIFY=&SYSUID
//IEBISAM EXEC PGM=IEBISAM,PARM='LOAD'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=OLD,DSN=JAY01.ISAM.MASTER.BACKUP
//SYSUT2   DD DISP=(NEW,CATLG,DELETE),DSN=JAY01.ISAM.MASTER,
//            UNIT=SYSDA,VOL=SER=PUB000,SPACE=(CYL,(5,1)),
//            DCB=(DSORG=IS,RECFM=FB,LRECL=200,BLKSIZE=23400,
//            KEYLEN=7,RKP=1,CYLOFL=2,NTM=4,OPTCD=LMY)
//
```

Print ISAM data records

```
//IEBISAMP JOB (1),'IEBISAM-RELOAD',
//             CLASS=A,MSGCLASS=X,REGION=2M,NOTIFY=&SYSUID
//IEBISAM EXEC PGM=IEBISAM,PARM='PRINTL,N'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=JAY01.ISAM.MASTER,DISP=OLD,DCB=DSORG=IS
//SYSUT2   DD SYSOUT=*
```

```
//
```

Source: GC26-3902-1 OS/VS2 MVS Utilities Release 3.8

IEBPTPCH **[supplied by IBM, located in SYS1.LINKLIB]**

Used to:

- Print or punch a sequential or partitioned dataset in its entirety.
- Print or punch selected members from a partitioned dataset.
- Print or punch selected records from a sequential or partitioned dataset.
- Print or punch the directory of a partitioned dataset.
- Print or punch an edited version of a sequential or partitioned dataset.

IEBPTPCH requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines a sequential input dataset that contains IEBPTPCH control statements.
- SYSUT1 - defines the input dataset
- SYSUT2 - defines the output dataset

I would also suggest taking a look at LISTPDS and PDSPRINT.

Control Statements

```
PRINT [PREFORM={A | M}]

      [,TYPORG={PS | PO}]

      [,TOTCONV={XE | PZ}]

      [,CNTRL={n | 1}]

      [,STRTAFT=n]

      [,STOPAFT=n]

      [,SKIP=n]

      [,MAXNAME=n]

      [,MAXFLDS=n]

      [,MAXGPS=n]

      [,MAXLITS=n]

      [,INITPG=n]

      [,MAXLINE=n]
```

The PRINT control statement specifies that the data is to be printed. If this is a print operation, the PRINT statement must be the first statement in the control statements. The parameters are:

PREFORM = {A | M}

Specifies that a control character is provided as the first character of each record to be printed. The control characters are used to control the spacing, number of lines per page, and page ejection. That is, the output has been previously formatted, and the "standard specifications" are superseded. If an error occurs, the print operation is terminated. If PREFORM is coded, any additional PRINT operands other than TYPORG, and all other control statements except for LABELS, are ignored. Any ignored statement or operands are checked for correct syntax, however. PREFORM must not be used for printing datasets with VS or VBS records longer than 32K bytes. These values can be coded:

- A - Specifies that an ASA control character is provided as the first character of each record to be printed. If the input record length exceeds the output record length, the utility uses the ASA character for printing the first line, with a single space character on all subsequent lines of the record.
- M - Specifies that a machine-code control character is provided as the first character of each record to be printed. If the input record length exceeds the output record length, the utility prints all lines of the record with a print-skip-one-line character until the last line of the record, which will contain the actual character provided as input.

TYPORG={PS | PO}

Specifies the organization of the input dataset. These values can be coded:

- PS - Specifies that the input dataset is organized sequentially.
- PO - Specifies that the input dataset is partitioned.

TOTCONV = {XE | PZ}

Specifies the representation of data to be printed or punched. TOTCONV can be overridden by any user specifications (RECORD statements) that pertain to the same data. These values can be coded:

- XE - Specifies that data is to be punched in 2-character-per-byte hexadecimal representation (for example, C3 40 F4 F6). If XE is not specified, data is punched in 1-character per byte alphameric representation. The above example would appear as C 46.
- PZ - Specifies that data (packed decimal mode) is to be converted to unpacked decimal mode. IEBPTPCH does not check for packed decimal mode. The output is unpredictable when the input is other than packed decimal.

Default: If TOTCONV is omitted, data is not converted.

CNTRL={n | 1}

Specifies a control character for the output device that indicates line spacing, as follows:

- 1 indicates single spacing;
- 2 indicates double spacing; and
- 3 indicates triple spacing.

STRTAFT=n

Specifies, for sequential datasets, the number of logical records (physical blocks in the case of VS or VBS type records longer than 32K bytes) to be skipped before printing begins. For partitioned datasets, STRTAFT=n specifies the number of logical records to be skipped in each member before printing begins. The n value must not exceed 32,767. If STRTAFT is specified and RECORD control statements are present, the first RECORD control statement of a member describes the format of the first logical record to be printed.

STOPAFT=n

Specifies, for sequential datasets, the number of logical records (or physical blocks in the case of VS or VBS records longer than 32K bytes) to be printed. For partitioned datasets, this specifies the number of logical records (or physical blocks in the case of VS or VBS records longer than 32K bytes) to be printed in each member to be processed. The n value must not exceed 32,767. If STOPAFT is specified and RECORD control statements are present, the operation is terminated when the STOPAFT count is satisfied or at the end of the first record group, whichever occurs first.

SKIP=n

Specifies that every nth record (or physical block in the case of VS or VBS records longer than 32K bytes) is to be printed. Default: Successive logical records are printed.

MAXNAME=n

Specifies a number no less than the total number of subsequent MEMBER control statements. The value must not exceed 32,767. Default: If MAXNAME is omitted when there is a MEMBER statement present, the print or punch request is terminated.

MAXFLDS=n

Specifies a number no less than the total number of FIELD parameters appearing in subsequent RECORD control statements. The value must not exceed 32,767. Default: If MAXFLDS is omitted when there is a FIELD parameter present, the print or punch request is terminated.

MAXGPS=n

Specifies a number no less than the total number of IDENT parameters appearing in subsequent RECORD control statements. The value must not exceed 32,767. Default: If MAXGPS is omitted when there is an IDENT parameter present, the print request is terminated.

MAXLITS=n

Specifies a number no less than the total number of characters contained in the IDENT literals of subsequent RECORD control statements. The value must not exceed 32,767. Default: If MAXLITS is omitted when there is a literal present, the print request is terminated.

INITPG=n

Specifies the initial page number; the pages are numbered sequentially thereafter. The INITPG parameter must not exceed a value of 9999. Default: 1

MAXLINE=n

Specifies the maximum number of lines to a printed page. Spaces, titles, and subtitles are included in this number. Default: 60

```
PUNCH [PREFORM={A | M}]

      [,TYPORG={PS | PO}]

      [,TOTCONV={XE | PZ}]

      [,CNTRL={n | 1}]

      [,STRTAFT=n]

      [,STOPAFT=n]

      [,SKIP=n]

      [,MAXNAME=n]

      [,MAXFLDS=n]

      [,MAXGPS=n]

      [,MAXLITS=n]

      [,CDSEQ=n]

      [,CDINCR=n]
```

The PUNCH control statement specifies that the data is to be punched. If this is a punch operation, the PUNCH statement must be the first statement in the control statements. The parameters are:

PREFORM={A | M}

Specifies that a control character is provided as the first character of each record to be punched. The control characters are used to control selecting a stacker. That is, the output has been previously formatted, and the "standard specifications" are superseded. If an error occurs, the punch operation is terminated. If PREFORM is coded, any additional PUNCH operands other than TYPORG, and all other control statements except for LABELS, are ignored. Any ignored statement or operands are checked for correct syntax, however. PREFORM must not be used for punching datasets with VS or VBS records longer than 32K bytes. These values can be coded:

- A - Specifies that an ASA control character is provided as the first character of each record to be punched. If the input record length exceeds the output record length, the utility uses the ASA character for punching first record, and duplicates the ASA character on each output card of the record.

- M - Specifies that a machine-code control character is provided as the first character of each record to be punched. If the input record length exceeds the output record length, the utility duplicates the machine control character on each output card of the record.

TYPORG={PS | PO}

Specifies the organization of the input dataset. These values can be coded:

- PS - Specifies that the input dataset is organized sequentially.
- PO - Specifies that the input dataset is partitioned.

TOTCONV = {XE | PZ}

Specifies the representation of data to be printed or punched. TOTCONV can be overridden by any user specifications (RECORD statements) that pertain to the same data. These values can be coded:

- XE - Specifies that data is to be punched in 2-character-per-byte hexadecimal representation (for example, C3 40 F4 F6). If XE is not specified, data is punched in 1-character per byte alphameric representation. The above example would appear as C 46.
- PZ - Specifies that data (packed decimal mode) is to be converted to unpacked decimal mode. IEBPTPCH does not check for packed decimal mode. The output is unpredictable when the input is other than packed decimal.

Default: If TOTCONV is omitted, data is not converted.

CNTRL={n | 1}

Specifies a control character for the output device that is used to select the stacker, as follows:

- 1 indicates the first stacker and
- 2 indicates the second stacker.

STRTAFT=n

Specifies, for sequential datasets, the number of logical records (physical blocks in the case of VS or VBS type records longer than 32K bytes) to be skipped before punching begins. For partitioned datasets, STRTAFT=n specifies the number of logical records to be skipped in each member before punching begins. The n value must not exceed 32,767. If STRTAFT is specified and RECORD control statements are present, the first RECORD control statement of a member describes the format of the first logical record to be punched.

STOPAFT=n

Specifies, for sequential datasets, the number of logical records (or physical blocks in the case of VS or VBS records longer than 32K bytes) to be punched. For partitioned datasets, this specifies the number of logical records (or physical blocks in the case of VS or VBS records longer than 32K bytes) to be punched in each member to be processed. The n value must not exceed 32,767. If STOPAFT is

specified and RECORD control statements are present, the operation is terminated when the STOPAFT count is satisfied or at the end of the first record group, whichever occurs first.

SKIP=n

Specifies that every nth record (or physical block in the case of VS or VBS records longer than 32K bytes) is to be punched. Default: Successive logical records are punched.

MAXNAME=n

Specifies a number no less than the total number of subsequent MEMBER control statements. The value must not exceed 32,767. Default: If MAXNAME is omitted when there is a MEMBER control statement present, the punch request is terminated.

MAXFLDS=n

Specifies a number no less than the total number of FIELD parameters appearing in subsequent RECORD control statements. The value must not exceed 32,767. Default: If MAXFLDS is omitted when there is a FIELD parameter present, the punch request is terminated.

MAXGPS=n

Specifies a number no less than the total number of IDENT parameters appearing in subsequent RECORD control statements. The value must not exceed 32,767. Default: If MAXGPS is omitted when there is an IDENT parameter present, the punch request is terminated.

MAXLITS=n

Specifies a number no less than the total number of characters contained in the IDENT literals of subsequent RECORD control statements. The value must not exceed 32,767. Default: If MAXLITS is omitted when there is a literal present, the punch request is terminated.

CDSEQ=n

Specifies the initial sequence number of a deck of punched cards. This value must be contained in columns 73 through 80. Sequence numbering is initialized for each member of a partitioned dataset. If the value of n is zero, 00000000 is assumed as a starting sequence number. Default: Cards are not numbered.

CDINCR=n

Specifies the increment to be used in generating sequence numbers. Default: 10 is the increment value.

```
TITLE ITEM=('title'[,output-location])[,ITEM...]
```

The TITLE control statement specifies that a title is to precede the printed or punched data. The TITLE

control statement, if included, follows the PRINT or PUNCH control statement in the control dataset. The parameter is:

ITEM=('title'[,output-location])[,ITEM...]

Specifies title or subtitle information. The values that can be coded are:

- 'title' - Specifies the title or subtitle literal (maximum length of 40 bytes), enclosed in apostrophes. If the literal contains apostrophes, each apostrophe must be written as two consecutive apostrophes.
- output-location - Specifies the starting position at which the literal for this item is to be placed in the output record. The specified title may not exceed the output logical record length minus one. Default: 1

```
EXITS [INHDR=routinename]

      [,INTLR=routinename]

      [,INREC=routinename]

      [OUTREC=routinename]
```

The EXITS control statement specifies that user routines are provided. Exits to label processing routines are ignored if the input dataset is partitioned. The EXITS control statement statement, if included, must immediately follow any TITLE control statement or follow the PRINT or PUNCH control statement. The parameters are:

- INHDR=routinename - Specifies the symbolic name of a routine that processes user input header labels.
- INTLR=routinename - Specifies the symbolic name of a routine that processes user input trailer labels.
- INREC=routinename - Specifies the symbolic name of a routine that manipulates each logical record (or physical block in the case of VS or VBS records longer than 32K bytes) before it is processed.
- OUTREC=routinename - Specifies the symbolic name of a routine that manipulates each logical record (or physical block in the case of VS or VBS records longer than 32K bytes) before it is printed or punched.

```
MEMBER NAME={membername | aliasname}
```

The MEMBER control statement specifies that the input is a partitioned dataset and that a selected member is to be printed or punched. The parameter is:

NAME={membemame | aliasname}

Specifies a member to be printed or punched. These values can be coded:

- membername - Specifies a member by its member name.
- aliasname - Specifies a member by its alias.

```
RECORD[IDENT=(length,'name',input-location)]

      [,FIELD=(length,[input-location],[conversion]

      ,[output-location])[,FIELD=...]
```

The RECORD control statement specifies whether editing is to be performed, that is, records are to be printed or punched to nonstandard specifications. The parameters are:

IDENT=(length,'name',input-location)

Identifies the last record of the record group to which the FIELD parameters apply. The values that can be coded are:

- length - Specifies the length (in bytes) of the field that contains the identifying name in the input records. The length cannot exceed eight bytes.
- 'name' - Specifies the exact literal that identifies the last record of a record group. If the literal contains apostrophes, each must be written as two consecutive apostrophes.
- input-location - Specifies the starting location of the field that contains the identifying name in the input records.

Note: The sum of the length and the input location must be equal to or less than the input LRECL plus one. Default: If IDENT is omitted and STOPAFT is not included with the PRINT or PUNCH control statement, record processing halts after the last record in the dataset. If IDENT is omitted and STOPAFT is included with the PRINT or PUNCH control statement, record processing halts when the STOPAFT count is satisfied or after the last record of the dataset is processed, whichever occurs first.

FIELD=(length,[input-location],[conversion],[output-location])

Specifies field-processing and editing information. These values can be coded:

- length - Specifies the length (in bytes) of the input field to be processed. Note: The length must be equal to or less than the initial input LRECL.
- input-location - Specifies the starting byte of the input field to be processed. Default: 1 Note: The sum of the length and the input location must be equal to or less than the input LRECL plus one.
- conversion - Specifies a two-byte code that indicates the type of conversion to be performed on this field before it is printed or punched. The values that can be coded are:
  - PZ - Specifies that data (packed decimal) is to be converted to unpacked decimal data. The converted portion of the input record (length L) occupies 2L - 1 output characters when punching, and 2L output characters when printing.
  - XE - Specifies that data (alphameric) is to be converted to hexadecimal data. The converted portion of the input record (length L) occupies 2L output characters.

  Default: The field is moved to the output area without change.

- output-location - Specifies the starting location of this field in the output records. Unspecified fields in the output records appear as blanks in the printed or punched output. Data that exceeds the SYSUT2 printer or punch size is not printed or punched. The specified fields may not exceed the

logical output record length minus one. When specifying one or more FIELDs, the sum of all lengths and all extra characters needed for conversions must be equal to or less than the output LRECL minus one.

Default: 1

```
LABELS [CONV={PZ | XE}]

       [,DATA={YES | NO | ALL | ONLY}]
```

The LABELS control statement specifies whether user labels are to be treated as data. The parameters are:

CONV={PZ I XE}

Specifies a two-byte code that indicates the type of conversion to be performed on this field before it is printed or punched. The values that can be coded are:

- PZ - Specifies that data (packed decimal) is to be converted to unpacked decimal data. The converted portion of the input record (length L) occupies 2L - 1 output characters.
- XE - Specifies that data (alphameric) is to be converted to hexadecimal data. The converted portion of the input record (length L) occupies 2L output characters.

Default: The field is moved to the output area without change.

DATA={YES | NO | ALL | ONLY}

Specifies whether user labels are to be treated as data. The values that can be coded are:

- YES - Specifies that any user labels that are not rejected by a user's label processing routine are to be treated as data. Processing of labels as data stops in compliance with standard return codes.
- NO - Specifies that user labels are not to be treated as data.
- ALL - Specifies that user labels are to be treated as data regardless of any return code. A return code of 16 causes the utility to complete the processing of the remainder of the group of user labels and to terminate the job step.
- ONLY - Specifies that only user header labels are to be treated as data. User header labels are processed as data regardless of any return code. The job terminates upon return from the OPEN routine.

Examples

Print sequential dataset with conversion to hex

```
//IEBPTPCH JOB (001),'PRINT SEQ DS   ',CLASS=A,MSGCLASS=X
//IEBPTPCH EXEC PGM=IEBPTPCH,REGION=1M
//SYSPRINT  DD  SYSOUT=*
//SYSUT1    DD  DISP=SHR,DSN=JAY01.ISAMVSAM.DATA
//SYSUT2    DD  SYSOUT=*
```

```
//SYSIN     DD  *
  PRINT TOTCONV=XE
  TITLE ITEM=('PRINT SEQ DATASET WITH CONV TO HEX',10)
//
```

Print partitioned dataset, 10 records from each member

```
//IEBPTPCH JOB (001),'PRINT PDS 10 EA',CLASS=A,MSGCLASS=X
//IEBPTPCH EXEC PGM=IEBPTPCH,REGION=1M
//SYSPRINT  DD  SYSOUT=*
//SYSUT1    DD  DISP=SHR,DSN=JAY01.ALL.PTFS
//SYSUT2    DD  SYSOUT=*
//SYSIN     DD  *
  PRINT TYPORG=PO,STOPAFT=10
  TITLE ITEM=('PRINT PDS - 10 RECORDS EACH MEM',10)
//
```

Copy all members from PDS onto single file on tape

```
//IEBPTPCH JOB (001),'COPY PDS TO TAPE',CLASS=A,MSGCLASS=X
//IEBPTPCH EXEC PGM=IEBPTPCH,REGION=2M
//SYSPRINT  DD  SYSOUT=*
//SYSUT1    DD  DISP=SHR,DSN=JAY01.ALL.PTFS
//SYSUT2    DD  UNIT=TAPE,VOL=SER=PTFS21,DISP=(,KEEP,DELETE),
//              DSN=PTFS,DCB=(RECFM=FB,LRECL=80,BLKSIZE=7200)
//SYSIN     DD  *
  PUNCH TYPORG=PO,MAXFLDS=1
//
//
```

Print two members of PDS

```
//IEBPTPCH JOB (001),'PRINT 2 PDS MEMBERS',CLASS=A,MSGCLASS=X
//IEBPTPCH EXEC PGM=IEBPTPCH,REGION=2M
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DISP=SHR,DSN=JAY01.ISAM.COBOL
//SYSUT2   DD  SYSOUT=*              change to SYSOUT=B to produce cards
//SYSIN    DD  *
  PUNCH TYPORG=PO,MAXFLDS=1,MAXNAME=2
  MEMBER NAME=ISREPT1
  MEMBER NAME=ISREPT2
//
//
```

Resequence card deck

```
//SEQCARDS JOB (001),SEQCARDS,CLASS=A,MSGCLASS=X
//IEBPTPCH EXEC PGM=IEBPTPCH
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD *

 [card deck to be resequenced goes here]

/*
//SYSUT2   DD SYSOUT=B
//SYSIN    DD *
 PUNCH MAXFIELDS=1,CDSEQ=0,CDINCR=100
 RECORD FIELD=(72)
/*
//
```

Source: GC26-3902-1 OS/VS2 MVS Utilities Release 3.8

IEBUPDTE **[supplied by IBM, located in SYS1.LINKLIB]**

Used to incorporate IBM and user-generated source language modifications into sequential or partitioned datasets. Exits are provided for user routines that process user header and trailer labels.

IEBUPDTE can be used to:

- Create and update symbolic libraries.
- Incorporate changes to partitioned members or sequential datasets.
- Change the organization of a dataset from sequential to partitioned or vice versa.

IEBUPDTE requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines a sequential input control dataset.
- SYSUT1 - defines the input dataset
- SYSUT2 - defines the output dataset

IEBUPDTE originated in an age that preceded full screen editing, so much of its functionality is devoted to applying updates to partitioned datasets in a batch job. In the current era, IEBUPDTE is mainly used to reload or add members to partitioned datasets. I would suggest taking a look at OFFLOAD, PDSLOAD, UPDTE, UNUPDTE, PDSPRINT and REVLMOD, which were written to efficiently perform load, unload, and reload functions on partitioned datasets.

Control Statements

All control statements are read from the SYSIN DD with data statements interspersed. In order for IEBUPDTE to recognize the control statements, each control statement must contain a period in column 1, followed by a slash in column 2, followed by a space in column 3. An optional label may appear next on the control statement, followed by a blank, then the action word.

```
./ ADD | CHANGE | REPL | REPRO

  [LIST=ALL]

  [,SEQFLD={ }]

  [,NEW={PO | PS}]

  [,MEMBER=membername]

  [,COLUMN=dd]

  [,UPDATE=INPLACE]
  [,INHDR=routinename]

  [,INTLR=routinename]

  [OUTHDR=routinename]

  [OUTLR=routinename]

  [,TOTAL=routinename,size]
```

```
[,NAME=membername]

[,LEVEL=hh]

[,SOURCE=x]

[,SSI=hhhhhhhh]
```

The Function control statement is used to initiate an IEBUPDTE operation. At least one Function control statement must be provided for each member or dataset to be processed. The function actions are:

ADD

Specifies that a member or a dataset is to be added to an old master dataset. If a member is to be added and the member name already exists in the old master dataset, processing is terminated. If, however, PARM=NEW is specified on the EXEC statement, the member is replaced. For a sequential output master dataset, PARM=NEW must always be specified on the EXEC statement.

When a member replaces an identically named member on the old master dataset or a member is changed and rewritten on the old master, the alias (if any) of the original member still refers to the original member. However, if an identical alias is specified for the newly written member, the original alias entry in the directory is changed to refer to the newly written member.

CHANGE

Specifies that a member of a dataset is being entered in its entirety as a replacement for a sequential dataset or for a member of the old master dataset. The member name must already exist in the old master dataset.

REPL

Specifies that a member of a dataset is being entered in its entirety as a replacement for a sequential dataset or for a member of the old master dataset. The member name must already exist in the old master dataset.

REPRO

Specifies that a member or a dataset is to be copied in its entirety to a new master dataset.

The Function control statement parameters are:

LIST=ALL

Specifies that the SYSPRINT dataset is to contain the entire updated member or dataset and the control statements used in its creation. Default: For old datasets, if LIST is omitted, the SYSPRINT dataset contains modifications and control statements only. If UPDATE was specified, the entire updated member is listed only when renumbering has been done. For new datasets, the entire member or dataset and the control statements used in its creation are always written to the SYSPRINT dataset.

SEQFLD={ddl | (ddl,ddl)}

ddl

Specifies, in decimal, the starting column (up to column 80) and length (8 or less) of sequence numbers within existing logical records and subsequent data statements. Note that the starting column specification (dd) plus the length (l) cannot exceed the logical record length (LRECL) plus 1. Sequence numbers on incoming data statements and existing logical records must be padded to the left with enough zeros to fill the length of the sequence field.

(ddt, ddt)

May be used when an alphameric sequence number generation is required. The first ddt specifies the sequence number columns as above. The second ddt specifies, in decimal, the starting column (up to column 80) and length (8 or less) of the numeric portion of the sequence numbers in subsequent NUMBER statements. This information is used to determine which portion of the sequence number specified by the NEW1 parameter may be incremented and which portion(s) should be copied to generate a new sequence number for inserted or renumbered records. Note: The numeric columns must fall within the sequence number columns specified (or defaulted) by the first ddt. Acceptable alphameric characters are A-Z, 1-9, @, #, $, *. Default: 738 is assumed, that is, an eight-byte sequence number beginning in column 73. Therefore, if existing logical records and subsequent Data statements have sequence numbers in columns 73 through 80, this keyword need not be coded.

NEW={PO | PS}

Specifies the organization of the old master dataset and the organization of the updated output. NEW should not be specified unless the organization of the new master dataset is different from the organization of the old master. These values can be coded:

- PO - Specifies that the old master dataset is a sequential dataset, and that the updated output is to become a member of a partitioned dataset.
- PS - Specifies that the old master dataset is a partitioned dataset, and that a member of that dataset is to be converted into a sequential dataset.

MEMBER=membername

Specifies, a name to be assigned to the member placed in the partitioned dataset defined by the SYSUT2 DD statement. MEMBER is used only when SYSUT1 defines a sequential dataset, SYSUT2 defines a partitioned dataset, and NEW=PO is specified.

COLUMN=dd

Note: Used only with CHANGE. Specifies, in decimal, the starting column of a data field within a logical record image. The field extends to the end of the image. Within an existing logical record, the data in the defined field is replaced by data from a subsequent Data statement.

UPDATE=INPLACE

Note: Used only with CHANGE. Specifies that the old master dataset is to be updated within the space it actually occupies. The old master dataset must reside on a direct access device. UPDATE is valid only when coded with CHANGE. No other function statements (ADD, REPL, REPRO) may be in the same job step.

INHDR=routinename

Specifies the symbolic name of the user routine that handles any user input (SYSUTl) header labels. When used with UPDATE, this routine assumes a special function. This parameter is valid only when a sequential dataset is being processed.

INTLR=routinename

Specifies the symbolic name of the user routine that handles any user input (SYSUTl) trailer labels. INTLR is valid only when a sequential dataset is being processed, but not when UPDATE is coded.

OUTHDR=routinename

Specifies the symbolic name of the user routine that handles any user output (SYSUT2) header labels. OUTHDR is valid only when a sequential dataset is being processed, but not when UPDATE is coded.

OUTLR=routinename

Specifies the symbolic name of the user routine that handles any user output (SYSUT2) trailer labels. OUTTLR is valid only when a sequential dataset is being processed, but not when UPDATE is coded.

TOTAL=routinename,size

Specifies that exits to a user's routine are to be provided prior to writing each record. This parameter is valid only when a sequential dataset is being processed. These values are coded:

- routinename - Specifies the name of the user's totaling routine.
- size - Specifies the number of bytes required for the user's data. The size should not exceed 32K, nor be less than 2 bytes. In addition, the keyword OPTCD=T must be specified for the SYSUT2 (output) DD statement.

NAME=membername

Specifies the name of the member placed into the partitioned dataset. The member name need not be specified in the DD statement itself. NAME must be provided to identify each input member.

This parameter is valid only when a member of a partitioned dataset is being processed.

LEVEL=hh

Specifies the change (update) level in hexadecimal (00-FF). The level number is

recorded in the directory entry of the output member. This parameter is valid only when a member of a partitioned dataset is being processed. This parameter has no effect when SSI is specifed.

SOURCE=x

Specifies user modifications when the x value is 0, or IBM modifications when the x value is 1. The source is recorded in the directory entry of the output member. This parameter is valid only when a member of a partitioned dataset is being processed. This parameter has no effect when SSI is specified.

SSI=hhhhhhhh

Specifies eight hexadecimal characters of system status information (SSI) to be placed in the directory of the new master dataset as four packed hexadecimal bytes of user data. This parameter is valid only when a member of a partitioned dataset is being processed. SSI overrides any LEVEL or SOURCE data given on the same Function statement.

```
./ NUMBER | DELETE

  [SEQ1={cccccccc | ALL}]

  [,SEQ2=cccccccc]

  [,NEW1=cccccccc]

  [,INCR=nnnnnnnn]

  [,INSERT=YES]
```

A Detail control statement is used with a Function control statement for certain applications, such as deleting or renumbering selected logical records. NUMBER specifies, when coded with a CHANGE Function control statement, that the sequence number of one or more logical records is to be changed. It specifies, when coded with an ADD or REPL Function control statement, the sequence numbers to be assigned to the records within new or replacement members or datasets. When used with an ADD or REPL Function control statement, no more than one NUMBER Detail control statement is permitted for each ADD or REPL Function control statement. If NUMBER is coded, it must be preceded and followed by at least one blank. DELETE specifies, when coded with a CHANGE Function control statement, that one or more logical records are to be deleted from a member or dataset. If DELETE is coded, it must be preceded and followed by at least one blank.

When INSERT is coded on the Function control statement:

- The SEQ1 parameter specifies the existing logical record after which the insertion is to be made. The SEQ2 parameter need not be coded; SEQ1=ALL cannot be coded.
- The NEWl parameter assigns a sequence number to the first logical record to be inserted. If the parameter is alphameric, the SEQFLD=(ddl,ddl) parameter should be coded .
- The INCR parameter is used to renumber as much as is necessary of the member or dataset from the point of the first insertion; the member or dataset is renumbered until an existing logical record is found whose sequence number is equal to or greater than the next sequence number to be assigned. If no such logical record is found, the entire member or dataset is renumbered.
- Additional NUMBER Detail statements, if any, must specify INSERT. If a prior numbering

operation renumbers the logical record specified in the SEQ1 parameter of a subsequent NUMBER Detail statement, any NEWl or INCR parameter specifications in the latter NUMBER statement are overridden. The prior increment value is used to assign the next successive sequence numbers. If a prior numbering operation does not renumber the logical record specified in the SEQ1 parameter of a subsequent NUMBER Detail statement, the latter statement must contain NEWl and INCR specifications.

- The block of Data statements to be inserted must contain blank sequence numbers.
- The insert operation is terminated when a Function statement, a Detail control statement, and end-of-file indication, or a Data statement containing a sequence number is encountered.
- The SEQ1, SEQ2, and NEW1 parameters (with the exception of SEQ1=ALL) specify eight (maximum) alphameric characters. The INCR parameter specifies eight (maximum) numeric characters. Only the significant part of a numeric sequence number need be coded; for example, SEQ1=00000010 can be shortened to SEQ1=10. If, however, the numbers are alphameric, the alphabetic characters must be specified; for example, SEQ1=00ABCO10 can be shortened to SEQ1=ABCO10.

The parameters are:

SEQ1={cccccccc | ALL}

Specifies records to be renumbered, deleted, or assigned sequence numbers. These values can be coded:

- cccccccc - Specifies the sequence number of the first logical record to be renumbered or deleted. This value is not coded in a NUMBER Detail control statement that is used with an ADD or REPL Function control statement. When this value is used in an insert operation, it specifies the existing logical record after which an insert is to be made. It must not equal the number of a statement just replaced or added. Refer to the INSERT parameter for additional discussion.
- ALL - Specifies a renumbering operation for the entire member or dataset. ALL is used only when a CHANGE Function control statement and a NUMBER Detail control statement are used. ALL must be coded if sequence numbers are to be assigned to existing logical records having blank sequence numbers. If ALL is not coded, all existing logical records having blank sequence numbers are copied directly to the output master dataset. When ALL is coded, SEQ2 need not be coded and one NUMBER detail statement is permitted per function statement. When ALL is coded:
  - SEQ2 need not be coded and
  - one NUMBER Detail control statement is permitted per Function control statement.

SEQ2=cccccccc

Specifies the sequence number of the last logical record to be renumbered or deleted. SEQ2 is required on all DELETE Detail control statements. If only one record is to be deleted, the SEQ1 and SEQ2 specifications must be identical. SEQ2 is not coded in a NUMBER Detail control statement that is used with an ADD or REPL Function control statement.

NEW1=cccccccc

Specifies the first sequence number assigned to new or replacement data, or specifies the first sequence number assigned in a renumbering operation. A value

specified in NEW1 must be greater than a value specified in SEQ1 (unless SEQ1=ALL is specified, in which case this rule does not apply).

INCR=nnnnnnnn

Specifies an increment value used for assigning successive sequence numbers to new or replacement logical records, or specifies an increment value used for renumbering existing logical records.

INSERT=YES

Specifies the insertion of a block of logical records. The records, which are Data statements containing blank sequence numbers, are numbered and inserted in the output master dataset. INSERT is valid only when coded with both a CHANGE Function control statement and a NUMBER Detail control statement. SEQl, NEWl, and INCR are required on the first NUMBER Detail control statement.

Data statement

A Data Statement is used with a Function control statement, or with a Function control statement and a Detail control statement. It contains a logical record used as replacement data for an existing logical record, or new data to be incorporated in the output master dataset. Each Data statement contains one logical record, which begins in the first column of the Data statement. The length of the logical record is equal to the logical record length (LRECL) specified for the output master dataset. Each logical record contains a sequence number to determine where the data is to be placed in the output master dataset. When used with a CHANGE Function control statement, a Data statement contains new or replacement data, as follows:

- If the sequence number in the Data statement is identical with a sequence number in an existing logical record, the Data statement replaces the existing logical record in the output master dataset.
- If no corresponding sequence number is found within the existing records, the Data statement is inserted in the proper collating sequence within the output master dataset. (For proper execution of this function, all records in the old master dataset must have a sequence number.)
- If a Data statement with a sequence number is used and INSERT= YES was specified, the insert operation is terminated. IEBUPDTE will continue processing if this sequence number is at least equal to the next old master record (record following the referred to sequence record).

When used with an ADD or REPL Function control statement, a Data statement contains new data to be placed in the output master dataset.

Sequence numbers within the old master dataset are assumed to be in ascending order. No validity checking of sequence numbers is performed for data statements or existing records.

Sequence numbers in Data statements must be in the same relative position as sequence numbers in existing logical records. (Sequence numbers are assumed to be in columns 73 through 80; if the numbers are in columns other than these, the length and relative position must be specified in a SEQFLD parameter within a preceding Function control statement.)

```
./ ALIAS | DELETE

  NAME=cccccccc
```

The ALIAS control statement is used to create or retain an alias in an output (partitioned) master directory. The ALIAS control statement can be used with any of the Function control statements. Multiple aliases can be assigned to each member up to a maximum of 16 aliases. Note: If an ALIAS statement is specifying a name which already exists on the dataset, the original TTR of that directory entry will be destroyed. ALIAS must be preceded and followed by at least one blank. If multiple ALIAS control statements are used, they must follow the data records.

The parameter is:

NAME=cccccccc

Specifies a one to eight character alias.

```
./ ENDUP
```

An ENDUP control statement can be used to indicate the end of SYSIN input to this job step. It serves as an end-of-data indication if there is no other preceding delimiter statement. The ENDUP control statement follows the last group of SYSIN control statements.

Program Parameter (PARM= on EXEC statement)

Additional information can be coded in the PARM parameter of the EXEC statement, as follows:

PARM={NEW | MOD},[inhdr],[intlr]

The action of the values which may be supplied are:

NEW

Specifies that the input consists solely of the control dataset. The input dataset (SYSUT2) is not defined if NEW is specified.

MOD

Specifies that the input consists of both the control dataset and the input dataset. If neither NEW nor MOD is coded, MOD is assumed.

inhdr

Specifies the symbolic name of a routine that processes the user header label on the volume containing the control dataset.

intlr

Specifies the symbolic name of a routine that processes the user trailer label on the volume containing the control dataset.

Examples

Add two members to a partitioned dataset

```
//IEBUPDTE JOB (001),'IEBUPDTE-01',CLASS=S,MSGCLASS=X
//IEBUPDTE EXEC PGM=IEBUPDTE,PARM=NEW
//SYSUT2   DD  DSN=SYS2.PROCLIB,DISP=MOD
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DATA       when statements to be added contain JCL statements, code DATA on SYS
./ ADD NAME=CLEARDMP
./ NUMBER NEW1=10,INCR=10
//CLEARDMP PROC DD=00         SPECIFY DD={00|01|02}
//* ************************************************************ *
//* CLEAR DUMP DATASET                                           *
//* ************************************************************ *
//EMPTY    EXEC PGM=IEBGENER
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DUMMY
//SYSUT1   DD  DUMMY,DCB=(RECFM=U,LRECL=100,BLKSIZE=100)
//SYSUT2   DD  DISP=SHR,DSN=SYS1.DUMP&DD
./ ADD NAME=CLEARERP
./ NUMBER NEW1=10,INCR=10
//CLEARERP PROC
//* ************************************************************ *
//* CLEAR ENVIRONMENTAL ERROR RECORDER DATASET                   *
//* ************************************************************ *
//EREP     EXEC PGM=IFCDIP00
//SERERDS  DD  DISP=SHR,DSN=SYS1.LOGREC
./ ENDUP
//
```

Change member in partitioned dataset, update in place

```
//IEBUPDTE JOB (001),'IEBUPDTE-02',CLASS=A,MSGCLASS=X
//IEBUPDTE EXEC PGM=IEBUPDTE
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=SYS2.PROCLIB,DISP=OLD
//SYSUT2   DD DSN=SYS2.PROCLIB,DISP=OLD
//SYSIN    DD DATA
./ CHANGE NAME=CLEARDMP,UPDATE=INPLACE
//SYSUT1   DD  DUMMY,DCB=(RECFM=U,LRECL=10,BLKSIZE=10)                  0000080
//STEP2 EXEC PGM=EXAMPLE2 00000010
0000080
./ ENDUP
/*
//
```

Add records to member in partitioned dataset

```
//IEBUPDTE JOB (001),'IEBUPDTE-03',CLASS=A,MSGCLASS=X
//IEBUPDTE EXEC PGM=IEBUPDTE
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=SYS1.PARMLIB,DISP=OLD
//SYSUT2   DD DSN=SYS1.PARMLIB,DISP=OLD
//SYSIN    DD DATA
./ CHANGE NAME=SAMPLE
//EXTRADD  DD SYSOUT=A                                  00000015
./ ENDUP
/*
//
```

Delete records from a member in a partitioned dataset

```
//IEBUPDTE JOB (001),'IEBUPDTE-04',CLASS=A,MSGCLASS=X
```

```
//IEBUPDTE EXEC PGM=IEBUPDTE
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=SYS1.PARMLIB,DISP=OLD
//SYSUT2   DD DSN=SYS1.PARMLIB,DISP=OLD
//SYSIN    DD *
./ CHANGE NAME=SAMPLE
./ DELETE SEQ1=15,SEQ2=20
/*
//
```

Create a partitioned dataset, input data contained solely in control dataset

```
//IEBUPDTE JOB (001),'IEBUPDTE-05',CLASS=A,MSGCLASS=X
//IEBUPDTE EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD SYSOUT=*
//SYSUT2   DD DSN=OUTLIB,UNIT=DISK,DISP=(NEW,KEEP),
//            VOL=SER=111112,SPACE=(TRK,(50,,10)),
//            DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD DATA
./ ADD NAME=MEMB1,LEVEL=00,SOURCE=0
 (data statements with sequence numbers in columns 73 through 80)
./ ADD NAME=MEMB2,LEVEL=00,SOURCE=0
 (data statements with sequence numbers in columns 73 through 80)
./ ADD NAME=MEMB3,LEVEL=00,SOURCE=0
 (data statements with sequence numbers in columns 73 through 80)
./ ENDUP
/*
//
```

Create a partitioned dataset, input data copied from existing dataset

```
//IEBUPDTE JOB (001),'IEBUPDTE-06',CLASS=A,MSGCLASS=X
//IEBUPDTE EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=SYS1.MACLIB,DISP=SHR
//SYSUT2   DD DSN=SYS9.MACLIB,UNIT=DISK,DISP=(NEW,KEEP),
//            VOL=SER=111112,SPACE=(TRK,(50,,10)),
//            DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD DATA
./ REPRO NAME=ATTACH,LEVEL=00,SOURCE=1,LIST=ALL
./ REPRO NAME=DETACH,LEVEL=00,SOURCE=1,LIST=ALL
./ ADD NAME=EXIT,LEVEL=00,SOURCE=1,LIST=ALL
./ NUMBER NEW1=10,INCR=100
 (data statements for new EXIT member)
./ ENDUP
/*
//
```

Source: GC26-3902-1 OS/VS2 MVS Utilities Release 3.8

IEHDASDR **[supplied by IBM, located in SYS1.LINKLIB]**

Warning: If you have installed Jim Morrison's modifications to provide 3375/3380/3390 support under MVS, be advised that IEHDASDR does not recognize these device types and will not function correctly, if at all, with volumes of these types. **As an alternative, I would suggest taking a look at ICKDSF for initializing DASD volumes and DSSDUMP/DSSREST for backing up DASD datasets.**

Used to prepare direct access volumes for use. In addition, IEHDASDR can be used to dump the entire contents or portions of a direct access volume to a volume or volumes of the same direct access device type, or to a tape volume or volumes. Data that is dumped to a tape volume is arranged so that it can subsequently be restored to its original

organization by IEHDASDR or IBCDMPRS (stand-alone restore program).

IEHDASDR requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines a sequential input dataset that contains IEHDASDR control statements.
- anyname - defines DASD volume to be used as input (DUMP) or output (RESTORE).
- anyname - defines tape volume to be used as input (RESTORE) or output (DUMP)
- anyname - defines source for IPL loader program

Control Statements

```
ANALYZE TODD=cuu

        ,VTOC=xxxxx

        ,EXTENT=xxxxx

        ,NEWVOLID=serial

        [,IPLDD=ddname]

        [,OWNERID=name]

        [,PURGE={YES | NO}
```

The ANALYZE control statement prepares the volume to make it ready for use; writes label track and creates Volume Table of Contents. The parameters are:

TODD=cuu

Specifies the channel and unit address of a direct access device containing an offline volume to be initialized.

VTOC=xxxxx

Specifies a one to five byte decimal relative track address representing a primary track on which the volume table of contents is to begin. The VTOC cannot occupy track 0.

EXTENT=xxxxx

Specifies the decimal length of the VTOC in tracks.

NEWVOLID=serial

Specifies a one to six character serial number.

IPLDD=ddname

Specifies,the ddname of a DD statement defining the dataset containing the IPL program. The IPL program can be defined as a sequential dataset or a member of a partitioned dataset.

OWNERID=name

Specifies a one to ten character name or other identifying information to be placed in the volume label record. OWNERID is specified as a character string of any alphameric, national character, hyphen (-), slash (/), or period (.).

PURGE={YES | NO}

Specifies whether the ANALYZE operation is be terminated when an unexpired dataset is encountered. These values can be coded:

- YES - Indicates that all unexpired datasets on the volume can be overwritten provided that the operator signals his concurrence when the first unexpired dataset is encountered. If PURGE=YES is coded and an unexpired dataset is encountered, the operator is prompted. The operator replies are:

  U - Which indicates that all unexpired datasets on this volume can be overwritten.

  T - Which indicates that this volume contains unexpired datasets that must not be overwritten.

- NO - Specifies that the various operations are to be terminated if an unexpired dataset is encountered.

The PURGE parameter does not apply to password-protected datasets; the operator must always respond with the proper password for each password-protected dataset encountered. If he is unable to do so, the operation is terminated.

```
LABEL TODD=cuu

      ,NEWVOLID=serial

      [,OWNERID=name]
```

The LABEL control statement changes the volume serial number of a direct access volume and, optionally, updates the owner field. The parameters are:

TODD=cuu

Specifies the channel and unit address of a direct access device containing an offline volume to be labeled.

NEWVOLID=serial

Specifies a one to six character serial number.

OWNERID=name

Specifies a one to ten character name or other identifying information to be placed in the volume label record. OWNERID is specified as a character string of any alphameric, national character, hyphen (-), slash (/), or period (.).

```
DUMP FROMDD=ddname

     ,TODD=ddname

     [,CPYVOLID={YES | NO}]

     [,BEGIN=ccccchhhh]

     [,END=ccccchhhh]
```

The DUMP control statement dumps a single track, a group of tracks, or an entire direct access volume. The parameters are:

FROMDD=ddname

Specifies the ddname of the DD statement defining the device containing the direct access volume from which a copy is to be made.

TODD=ddname

Specifies the ddnames of the DD statement defining the device containing the tape volume on which a copy is to be made.

CPYVOLID={YES | NO}

Specifies whether receiving direct access volumes are to be assigned the serial number of the dumped volume. These values can be coded:

- YES - Specifies that all receiving direct access volumes are to be assigned the serial number of the dumped volume.
- NO - Specifies that receiving volumes are to keep their own serial numbers.

BEGIN=ccccchhhh

Specifies in hexadecimal a cylinder number, cccc and head number, hhhh, that identify the first track to be dumped. If BEGIN is omitted, the dump operation begins with track 0.

END=ccccchhhh

Specifies in hexadecimal a cylinder number, cccc, and head number,hhhh, that identify the last track to be dumped. If only one track is to be dumped, both BEGIN and END specify that track address. Default: The last primary track of the volume is the last track to be copied.

```
RESTORE TODD=ddname

        ,FROMDD=ddname

        [,CPYVOLID={YES | NO}]

        [,PURGE={YES | NO}]
```

The RESTORE control statement restores a previously dumped direct access volume to a direct access device. The parameters are:

TODD=ddname

Specifies the ddname of the DD statement that identifies the volume serial number of the output volume.

FROMDD=ddname

Specifies the ddname of the DD statement that defines the tape volume containing the data to be restored.

CPYVOLID={YES | NO}

Specifies whether receiving direct access volumes are to be assigned the serial number of the dumped volume. These values can be coded:

- YES - Specifies that all receiving direct access volumes are to be assigned the serial number of the dumped volume.
- NO - Specifies that receiving volumes are to keep their own serial numbers.

PURGE={YES | NO}

Specifies whether the ANALYZE operation is be terminated when an unexpired dataset is encountered. These values can be coded:

- YES - Indicates that all unexpired datasets on the volume can be overwritten provided that the operator signals his concurrence when the first unexpired dataset is encountered. If PURGE=YES is coded and an unexpired dataset is encountered, the operator is prompted. The operator replies are:

  U - Which indicates that all unexpired datasets on this volume can be overwritten.

  T - Which indicates that this volume contains unexpired datasets that must not be overwritten.

- NO - Specifies that the various operations are to be terminated if an unexpired dataset is encountered.

The PURGE parameter does not apply to password-protected datasets; the operator must always respond with the proper password for each password-protected dataset encountered. If he is unable to do so, the operation is terminated.

Examples

Initialize 2314, 3330 (1 and 2), 3340, 3350 DASD volumes

```
//IEHDASDR JOB 1,'IEHDASDR-INIT',CLASS=S,MSGCLASS=X
//* ************************************************************ *
```

```
//* * USE ONLY FOR 3350 | 3330 | 3340 | 2314                          *
//* ************************************************************** *
//*
//IEHDASDR EXEC PGM=IEHDASDR,REGION=4096K
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
       ANALYZE TODD=127,                                             C
               VTOC=50,EXTENT=10,                                    C
               NEWVOLID=111111,                                      C
               OWNERID=HERCULES
/*
//
```

Dump DASD volume to tape

```
//IEHDASDR JOB 1,'IEHDASDR-DUMP',CLASS=S,MSGCLASS=X
//IEHDASDR EXEC PGM=IEHDASDR
//SYSPRINT DD SYSOUT=*
//DISK     DD UNIT=2314,VOL=SER=DLIB01,DISP=OLD
//TAPE     DD UNIT=TAPE,DSN=BACKUP.TAPE.DLIBS,DISP=(NEW,KEEP),
//            VOL=SER=DBKD01,LABEL=(2,SL)
//SYSIN    DD *
       DUMP FROMDD=DISK,TODD=TAPE
/*
//
```

Restore DASD volume from backup

```
//IEHDASDR JOB 1,'IEHDASDR-RESTORE',CLASS=S,MSGCLASS=X
//IEHDASDR EXEC PGM=IEHDASDR
//SYSPRINT DD SYSOUT=*
//TAPE     DD UNIT=BACKUP.TAPE.DLIBS,DSN=TAPE,DISP=OLD,
//            VOL=SER=DBKD01,LABEL=(2,SL)
//DISK     DD UNIT=2314,VOL=SER=111111,DISP=OLD
//SYSIN    DD *
       RESTORE FROMDD=TAPE,TODD=DISK,PURGE=YES,CPYVOLID=YES
/*
//
```

Alter Volume Serial on offline 2314

```
//IEHDASDR JOB 1,'IEHDASDR-RELABEL',CLASS=S,MSGCLASS=X
//IEHDASDR EXEC PGM=IEHDASDR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
       LABEL TODD=123,NEWVOLID=134134,OWNERID=HERCULES
/*
//
```

Source: GC26-3902-1 OS/VS2 MVS Utilities Release 3.8

IEHINITT **[supplied by IBM, located in SYS1.LINKLIB]**

Note: As the majority of the audience for this page is running MVS under Hercules, I will point out that there is a Hercules' utility provided to initialize emulated tape images: hetinit. The hetinit utility may be run from a command prompt (Windows) or terminal (Linux) or by using the Hercules' shell command (sh) from the Hercules' command prompt. In addition, the COPYMODS program may be used to initialize multiple tapes per execution. There is little use for the IBM IEHINITT utility, except for history.

Used to place IBM volume label sets written in EBCDIC onto any number of magnetic tapes mounted on one or more

tape units. Each volume label set created by the program contains:

- A standard volume label with user-specified serial number and owner identification.
- An 80-byte dummy header label. For IBM standard labels, this record consists of HDR1 followed by zeros.
- A tape mark.

IEHINITT requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines a sequential input dataset that contains IEHINITT control statements.
- anyname - defines a tape unit to be used in a labeling operation; more than one tape unit can be identified.

Control Statements

```
ddname INITT SER=xxxxxx

             [,OWNER='ccccccccc']

             [,NUMBTAPE = {n | 1]

             ,DISP= {REWIND I UNLOAD}
```

The INITT control statement provides control information for the IEHINITT program. Any number of INITT utility control statements can be included for a given execution of the program. An identically named DD statement must exist for each utility control statement in the job step. The parameters are:

ddname

Specifies a name that is identical to a ddname in the name field of a DD statement defining a tape unit(s). This name must begin in column 1.

SER=xxxxxx

Specifies the volume serial number of the first, or only tape to be labeled. The serial number cannot contain blanks, commas, apostrophes, equal signs, or special characters other than periods or hyphens. A specified serial number is incremented by one for each additional tape to be labeled. (Serial number 999999 is incremented to 000000.) When processing multiple tapes, the volume serial number must be all numeric.

OWNER='ccccccccccccc'

Specifies the owner's name or similar identification. The information is specified as character constants, and can be up to 10 bytes in length. The delimiting apostrophes can be omitted, if no blanks, commas, apostrophes, equal signs, or other special characters (except periods or hyphens) are included. If an apostrophe is included within the OWNER name field, it must be written as two consecutive apostrophes.

NUMBTAPE={n | 1}

Specifies the number of tapes to be labeled according to the specifications made in this control statement. The value n represents a number from 1 to 255. If more than one tape is specified, the serial number must be numeric.

DISP={REWIND | UNLOAD}

Specifies whether a tape is to be rewound or unloaded. These values can be coded:

- REWIND - Specifies that a tape is to be rewound (but not unloaded) after the label has been written. If DISP=REWIND is not specified, the tape volume is rewound and unloaded.
- UNLOAD - Specifies that a tape is to be unloaded after the label has been written.

Message **IEC701D** will be displayed on the console to request the mounting of scratch tape(s) to be initialized. A response of: **r nn,M** informs IEHINITT that the tape(s) have been mounted, so it will proceed.

Examples

Initialize 3 tapes

```
//IEHINITT JOB (SYS),'LABEL TAPES',MSGCLASS=X,CLASS=S
//IEHINITT EXEC PGM=IEHINITT
//SYSPRINT DD  SYSOUT=*
//LABEL    DD  UNIT=(TAPE,,DEFER)
//SYSIN    DD  *
LABEL INITT SER=014020,NUMBTAPE=3,OWNER=HERCULES,DISP=UNLOAD
/*
//
```

Source: GC26-3902-1 OS/VS2 MVS Utilities Release 3.8

IEHLIST **[supplied by IBM, located in SYS1.LINKLIB]**

Used to list entries in a volume table of contents, entries in the directory of one or more partitioned datasets, or entries in an OS catalog. **I would also suggest taking a look at IEHMAP, SUPERLST, and VTOCLIST (for VTOCs) and LISTPDS and MAPDISK (for Partitioned datasets).**

IEHLIST requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines a sequential input dataset that contains IEHLIST control statements.
- anyname - defines a DASD volume for which the Volume Table of Contents is to be listed or on which a Partitioned dataset resides for which the directory is to be listed.

Control Statements

```
LISTCTLG [VOL=device=serial]

         [,NODE=name]
```

The LISTCTLG control statement requests a listing of all or part of an OS catalog (SYSCTLG). **Note:** This is for listing OS CVOL catalogs; MVS uses VSAM catalogs (IDCAMS is the utility to list VSAM

catalog contents). The parameters are:

VOL=device=serial

Specifies the device type and volume serial number of the volume on which the catalog resides.

NODE=name

Specifies a qualified name. All dataset entries whose names are qualified by this name are listed. The CVOL must be defined in the VSAM Master Catalog as: SYSTCTLG.VYYYYYY, where YYYYYY is the serial number of the CVOL. Default: All dataset entries are listed.

```
LISTPDSDSNAME=dsname | (dsname[,dsname]...)

       [,VOL=device=serial]

       [,{DUMP | FORMAT}]
```

The LISTPDS control statement requests a directory listing of one or more partitioned datasets. The parameters are:

DSNAME=dsname | (dsname[,dsname]...)

Specifies the fully qualified names of the partitioned datasets whose directories are to be listed. A maximum of ten names is allowed. If the list consists of a single name, the parentheses can be omitted. DSNAME may be abbreviated to DSN.

VOL=device=serial

Specifies the device type and volume serial number of the volume on which the partitioned dataset resides.

DUMP | FORMAT

DUMP

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

FORMAT

Specifies that the listing is to be edited for each directory entry.

Default: If both DUMP and FORMAT are omitted, DUMP is the default used.

```
LISTVTOC [{DUMP | FORMAT}]

         [,DATE=dddyy]

         [,VOL=device=serial]
```

```
[,DSNAME=(name[,name]...)]
```

The LISTVTOC control statement requests a listing of all or part of a volume table of contents. The parameters are:

DUMP | FORMAT

DUMP

Specifies that the listing is to be in unedited, hexadecimal form. Default: If both DUMP and FORMAT are omitted, an abbreviated edited format is generated.

FORMAT

Specifies that a comprehensive edited listing is to be generated.

Default: If both DUMP and FORMAT are omitted, an abbreviated edited format is generated

DATE=dddyy

Specifies that each entry that expires before this date is to be flagged with an asterisk (*) in the listing. This parameter applies only to the abbreviated edited format. The date is represented by ddd, the day of the year, and yy, the last two digits of the year. Default: No asterisks appear in the listing.

VOL=device=serial

Specifies the device type and volume serial number of the volume on which the VTOC resides.

DSNAME=(name[,name]...)

Specifies the fully qualified names of the datasets whose entries are to be listed. A maximum of ten names is allowed. If the list consists of a single name, the parentheses can be omitted. DSNAME may be abbreviated to DSN.

Program Parameter (PARM= on EXEC statement)

Additional information can be coded in the PARM parameter of the EXEC statement, as follows:

PARM='LINECNT=nn'

Specifies the number of lines, nn, to be printed per page; nn is a decimal number from 01 through 99. If LINECNT is not specified, 58 lines are printed per page. The PARM field cannot contain embedded blanks, zeros, or any other PARM keywords, or the default of 58 is used.

Examples

List VTOC

```
//IEHLIST  JOB (001),'IEHLIST-VTOC',CLASS=A,MSGCLASS=X
//IEHLIST  EXEC PGM=IEHLIST
//SYSPRINT DD  SYSOUT=*
//VOL      DD  DISP=OLD,UNIT=SYSDA,VOL=SER=PUB000
//SYSIN    DD  *
  LISTVTOC FORMAT,VOL=SYSDA=PUB000
//
//
```

List PDS

```
//IEHLIST  JOB (001),'IEHLIST-PDS',CLASS=A,MSGCLASS=X
//IEHLIST EXEC PGM=IEHLIST
//SYSPRINT DD  SYSOUT=*
//VOL      DD  DISP=OLD,UNIT=SYSDA,VOL=SER=SYSCPK
//SYSIN    DD  *
  LISTPDS DSN=SYSC.PROCLIB,VOL=SYSDA=SYSCPK,FORMAT
//
//
```

Source: GC26-3902-1 OS/VS2 MVS Utilities Release 3.8

IEHMAP **[origin unknown, located in SYSC.LINKLIB]**

Used to obtain a formatted listing of a VTOC, a dataset or a catalog. Even though the name of this utility suggests it is an IBM utility, it is not. The source for several versions of IEHMAP may be found on the CBT Tape; the version that runs under MVS 3.8 is located in File #83 on CBT Tape V129. Note: The catalog functions are for OS catalogs, not MVS catalogs, which are VSAM, so the catalog functions of IEHMAP are not documented here.

IEHMAP requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines a sequential input dataset that contains IEHMAP control statements.
- DDNAME1 [through DDNAMEn] - defines the volume(s) on which IEHMAP is to operate.

Control Statements

```
NAME VOL=volid

     [,OPT=NOPRINT]

     [,OPT=AQUEUE]
```

The NAME control statement requests a report on all datasets on the volume, in addition to information about the Volume Table Of Contents. The information reported is:

A) Information taken from the formats 4, 5, and 6:

description of the device type for DASD volume

description of the VTOC of volume
summary of available space on the volume
summary of split cylinder datasets (if applicable)

B) An alphabetical listing of each dataset on volume, including:

creation date as dd/mm/yy
expiration date as dd/mm/yy **or** change date as dd/mm/yy **if OPT=AQUEUE specified**
volume created on **or** creation date as dd/mm/yy **if OPT=AQUEUE specified**
dataset sequence number **or** reference date as dd/mm/yy **if OPT=AQUEUE specified**
DSORG, RECFM, BLKSIZE, LRECL, KEYLEN, OPTCD
tracks allocated
tracks actually used
number of extents
secondary allocation
allocation type

C) The number of datasets on the volume and the space they occupy.

D) A count of the DSCBï¿½s in the VTOC

E) Errors found in DSCBï¿½s counts

F) The number of different types of dataset organizations found.

G) The number of datasets by extents.

H) The allocation of datasets by tracks.

I) The number of contiguous available cylinders.

J) The number of contiguous available tracks.

The parameters for the NAME control statement are:

VOL=volid

Specifies the volume serial number for the DASD volume on which to report.

OPT=NOPRINT

Specifies that detail information for the dataset(s) on the volume is to be suppressed; only items A and C through J below are reported. This is provided to give a quick summary of the VTOC.

OPT=AQUEUE

Specifies that the creation, reference and change dates be formatted in place of the volume created on, the dataset sequence number, and the expiration date.

```
TRACKS VOL=volid

       [,OPT=NOPRINT]
```

The TRACKS control statement requests a report on all track allocations on the volume, in addition to information about the Volume Table Of Contents. The information reported is:

A) The allocation of each track on the volume in CCHHR order:

the starting cchh
the ending cchh
the name of the dataset it is assigned to
an indication that it is free space
an indication that it is assigned to "unaccounted space" (vtoc, etc.)
the extent number (if assigned to a dataset)
the number of tracks in the extent

B) A listing of any errors found in track allocation:

overlapping tracks
missing tracks
invalid extents

Note: If missing tracks are found, in place of the starting and ending CCHHï¿½s a hexadecimal number will appear; this is the value that would have to be placed in a format 5 to recover this space as available space.

C) A totaling of all tracks found

D) A totaling of all tracks missing (if applicable)

The parameters for the TRACKS control statement are:

VOL=volid

Specifies the volume serial number for the DASD volume on which to report.

OPT=NOPRINT

Specifies that allocation information for the dataset(s) on the volume are to be suppressed; only items B through D below are reported..

```
MAP  VOL=volid

     [,OPT=NOPRINT]
```

The MAP control statement requests a report that will include all of the information that is reported by the NAME and TRACKS control statements (above).

The parameters for the MAP control statement are:

VOL=volid

Specifies the volume serial number for the DASD volume on which to report.

OPT=NOPRINT

Specifies that detail information for the dataset(s) on the volume and allocation information are to be suppressed.

```
DUMP {VOL=volid | dsname}
```

The DUMP control statement requests a report that will include all of the information that is reported by the NAME and TRACKS control statements (above), plus a dump of the DSCBs associated with each dataset. When listing a volume, it will also provide a hex dump of each DSCB in the format 4, 5, and 6 chains, and the format 1 chain.

The parameters for the DUMP control statement are:

VOL=volid

Specifies the volume serial number for the DASD volume on which to report.

dsname

Specifies the name of the dataset on which to report.

```
DUMP456 {VOL=volid | dsname}
```

THe DUMP456 control statement requests a report that will include all of the information that is reported by the NAME control statement (above), plus a hex dump of the format 4, 5, and 6 chains, when listing a volume, and the format 1 chain, when listing a dataset.

The parameters for the DUMP456 control statement are:

VOL=volid

Specifies the volume serial number for the DASD volume on which to report.

dsname

Specifies the name of the dataset on which to report.

```
CCHHR {VOL=volid | dsname}
```

The CCHRR control statement requests a report that will include all of the information that is reported by the DUMP control statement (above), however, the DSCBs will be dumped in CCHHR order; that is, they will be dumped in the order they are in the VTOC, not alphabetically by dataset name. Also, the various chains of DSCBs (the format 1 chain, the format 5 chain, and the format 6 chain) will not be dumped in chain order. This is similar to the dump option of IEHLIST.

The parameters for the CCHRR control statement are:

VOL=volid

Specifies the volume serial number for the DASD volume on which to report.

```
DSNAME dsname
```

The DSNAME control statement will report the same information as a NAME control statement for a single dataset.

The parameters for the DSNAME control statement are:

dsname

Specifies the name of the dataset on which to report.

```
PDS  {VOL=volid | dsname}
```

The PDS control statement will report the same information as a DSNAME control statement (above), but will report only partitioned datasets (DSORG=PO). If a single dataset is requested, it will not be listed if it is not a Partitioned dataset. It will also report a listing of the members from the directory in alphabetical order, along with a dump of the directory information in hexadecimal, and a summary of the directory space. Note: An 'A' after a member name indicates that it is an alias.

The parameters for the PDS control statement are:

VOL=volid

Specifies the volume serial number for the DASD volume on which to report.

dsname

Specifies the name of the dataset on which to report.

```
TTR  {VOL=volid | dsname}
```

The TTR control statement will report the same information as a PDS control statement (above), however, the members from the directory will be listed in TTR order, not alphabetically. Alias names will be sorted alphabetically after the true member name.

The parameters for the TTR control statement are:

VOL=volid

Specifies the volume serial number for the DASD volume on which to report.

dsname

Specifies the name of the dataset on which to report.

```
ATTRIB {VOL=volid[,OPT=TTR] | dsname}
```

The ATTRIB control statement will report the same information as a DSNAME control statement (above), but will report only partitioned datasets (DSORG=PO). If a single dataset is requested, it will not be listed if it is not a Partitioned dataset. It will also give a listing of the members from the directory in alphabetical order, and if the members are load modules, it will format the linkage editor data from the directory as follows:

the member name
an indication that the member is an alias
the ttr where the member starts
the number of user ttrï¿½s
the ttr where the text records start
the os/vs linkage editor flags
the load module size (storage required)
the load module entry point
the setssi data (if present)
the setcode data (if present)
the true member name if an alias
the true entry point if an alias
the load modules linkage editor attributes:

REFR refreshable
RENT re-entrant
REUS reuseable
DC downward compatible
OL only loadable
OVLY overlay
NE not editable
NOTEX not executable

The parameters for the ATTRIB control statement are:

VOL=volid

Specifies the volume serial number for the DASD volume on which to report.

OPT=TTR

Specifies that the member names will be sorted in TTR order rather than alphabetical.

dsname

Specifies the name of the dataset on which to report.

```
DIR  {VOL=volid | dsname}
```

The DIR control statement will report the same informaiton as a NAME control statement, but will report only partitioned datasets (DSORG=PO). If a single dataset is requested, it will not be listed if it is not a Partitioned dataset. A summary of the directory space will also be reported.

The parameters for the DIR control statement are:

VOL=volid

Specifies the volume serial number for the DASD volume on which to report.

dsname

Specifies the name of the dataset on which to report.

```
VERIFY  VOL=volid
```

The VERIFY control statement will report the same information as a NAME control statement, plus a verification of the format 4, 5, 6, and 1 chains is performed:

- all formats 5 are validated to be chained from the first format 5
- all formats 6 are validated to be chained from the format 4
- all formats 2 are validated to be chained from the format 1
- all formats 3 are validated to be chained from a format 1 or 2

Any errors, such as dangling (ie, non-chained) formats 2, 3, 5 or 6 are listed. If a second format 4 is found, it will be listed. If the key of a format 0 DSCB is not 44 hex zeroes, it too will be listed. If an unknown DSCB type (with an id higher than 6) is found, it will also be listed.

The parameter for the VERIFY control statement is:

VOL=volid

Specifies the volume serial number for the DASD volume on which to report.

```
FREE VOL=volid
```

The FREE control statement will report a description of the VTOC and a summary of the available space.

The parameter for the FREE control statement is:

VOL=volid

Specifies the volume serial number for the DASD volume on which to report.

```
OVERLAP VOL=volid
```

The OVERLAP control statement will check the VTOC of the volume for overlapping extents and list any that are found.

The parameter for the OVERLAP control statement is:

VOL=volid

Specifies the volume serial number for the DASD volume on which to report.

```
MISSING VOL=volid
```

The MISSING control statement will check the VTOC of the volume for missing tracks, and list any that are found.

The parameter for the MISSING control statement is:

VOL=volid

Specifies the volume serial number for the DASD volume on which to report.

```
AVAIL VOL=volid
```

The AVAIL control statement will list all available space on the volume in CCHHR order.

The parameter for the MISSING control statement is:

VOL=volid

Specifies the volume serial number for the DASD volume on which to report.

The AVAIL Control statement will list all available space on the volume in CCHHR order.

```
FIXUP VOL=volid
```

The FIXUP control statement makes an attempt to recover missing tracks on the volume. This is done by adding missing extents to the last format 5 in the format 5 chain, and in addition, one extra format 5 will be built from a format 0. By this means, from 26 to 51 missing extents may be recovered.

The output of FIXUP is the same as two TRACKS control statements. The first output shows the missing tracks that were found, and the action that IEHMAP took (replaced or not replaced). The second is a normal TRACKS output showing what the VTOC looks like after the FIXUP operation.

The parameter for the FIXUP control statement is:

VOL=volid

Specifies the volume serial number for the DASD volume to attempt corrections.

```
SORT VOL=volid
```

The SORT control statement makes a more encompassing attempt to recover missing tracks than a FIXUP control statement. IEHMAP attempts to arrange the missing tracks in CCHHR order in the format 5 chain, then the format 5 chain is rewritten. The output of a SORT operation is the same as a FIXUP; however, all free space will be listed as missing, and the action IEHMAP took will be shown.

SORT should be used if the second output of a FIXUP shows that IEHMAP has created multiple contiguous extents, and/or the volume has more than three formats 5. By multiple contiguous free extents is meant two or more contiguous extents marked as 'AVAILABLE' on the second output of FIXUP.

The parameter for the SORT control statement is:

VOL=volid

Specifies the volume serial number for the DASD volume to attempt corrections.

Program Parameter (PARM= on EXEC statement)

Additional information can be coded in the PARM parameter of the EXEC statement, as follows:

PARM='[LINES=nnn][,MIN=mmm],[MAX=vvv][,TSO]'

The parameters are:

LINES=nnn

nnn specifies the number of lines to print per page. If specified, the number may range 10 through 999; if a number is submitted outside this range, the default will be used. The default line count is 56. LINES may be abbreviated L.

MIN=mmm

mmm specifies the minimum value (in k-bytes) for the variable conditional GETMAINï¿½s IEHMAP does to obtain the area for control blocks, reading in DSCBï¿½s, etc. This will be rouanded to the next higher 4-k boundary. The default is 64 (16 4-k pages).

MAX=vvv

vvv is the maximum value (in k-bytes) for the variable conditional GETMAINï¿½s IEHMAP does to obtain the area for control blocks, reading in DSCBï¿½s, etc. This will be rounded to the next higher 4-k boundary. The value must be at least 1 (one) higher than the value of the MIN= parameter (or its default); if it is not, it will be set to mmm+4. The default value is 128 (32 4-k pages).

TSO

notifies IEHMAP that it is running under TSO, and, therefore, it is not authorized. When running under TSO, it cannot ENQ upon the VTOC or catalog; also, the FIXUP and SORT Control statements cannot be specified.

Examples

List all datasets on volume

```
//IEHMAP  JOB  (001),'IEHMAP',CLASS=A,MSGCLASS=X
//IEHMAP  EXEC PGM=IEHMAP,REGION=4096K
//SYSPRINT DD  SYSOUT=*
```

```
//DD1      DD  UNIT=SYSALLDA,VOL=SER=SYSP01,DISP=OLD
//SYSIN    DD  *
  NAME   VOL=SYSP01,OPT=AQUEUE
*
* List all volume information and all datasets:
*      NAME VOL=volid[,OPT=NOPRINT][OPT=AQUEUE]
*                                      change/create/refer dates
*                          suppress dataset list
* List track summary and all allocations for datasets:
*      TRACKS VOL=volid[,OPT=NOPRINT]
*                           suppress allocations for datasets
* List combined NAME & TRACK information with single command:
*      MAP VOL=volid[,OPT=NOPRINT]
*                         suppress dataset & allocation detail
* List MAP information, plus hex dump of DSCBs:
*      DUMP VOL=volid   datasetname
* List NAME information, plus hex dump of DSCBs:
*      DUMP456 VOL=volid   datasetname
* List DUMP information,only in CCHHR order:
*      CCHHR VOL=volid
* List NAME information for single dataset:
*      DSNAME datasetname
* List NAME information, only include Partitioned datasets:
*      PDS VOL=volid   datasetname
* List PDS information, only in TTR order:
*      TTR VOL=volid   datasetname
* List DSNAME information, only include Partitioned datasets:
*      ATTRIB VOL=volid[OPT=TTR]
*                             sort member names in TTR order
* List NAME information, plus directory information, only include
* Partitioned datasets:
*      DIR VOL=volid   dsname
* List NAME information, plus errors in DSCBs:
*      VERIFY VOL=volid
* List VTOC and available space information:
*      FREE VOL=volid
* List overlapping extents:
*      OVERLAP VOL=volid
* List missing tracks:
*      MISSING VOL=volid
* List available space in CCHHR order:
*      AVAIL VOL=volid
/*
//
```

IEHMOVE **[supplied by IBM, located in SYS1.LINKLIB]**

Used to move or copy one or several datasets from one DASD volume to another. IEHMOVE can be used to move or copy:

- A dataset residing on from one to five volumes, with the exception of ISAM datasets, and VSAM data spaces.
- A group of cataloged datasets.
- An OS catalog (CVOL) or portions of a CVOL.
- A volume of datasets.

In an IEHMOVE action, if the target device type is not capable of receiving a copy of the source dataset (i.e., source is DASD and target is tape), IEHMOVE attempts to reorganize the data in order to create an unloaded copy of the source dataset; if a subsequent IEHMOVE specifies the unloaded copy as source and a target that is capable of receiving the original dataset format, IEHMOVE attempts to recreate the original format of the original source dataset.

IEHMOVE frequently gets bad reviews, and there are other utilities that may be used to do most of the functions it provides, but I still find it useful when I am moving datasets from one DASD volume to another. I will not be including Control statements that I have found not to work, or problematic, under MVS and VSAM catalogs. **I would also suggest taking a look at PDSUR, which is able to read (and process) IEHMOVE unloaded datasets as well as create unloaded datasets in IEHMOVE format.**

In the scope of this page, IEHMOVE's functions relating to OS catalogs (CVOLs) is not really applicable, since MVS uses VSAM catalogs, so those functions will not be covered.

IEHMOVE requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines a sequential input dataset that contains IEHMOVE control statements.
- SYSUT1 - defines a DASD volume on which three required work datasets will be allocated.
- anyname1 - defines the source dataset.
- anyname2 - defines the target dataset.

Control Statements

Control statements for IEHMOVE follow the same rules as JCL; i.e., if a statement requires continuation onto a following record, the operands on the record to be continued are stopped following a comma before reaching column 72, a non-blank character is placed in column 72, and the statement continues on the following record beginning in column 16.

```
MOVE DSNAME=name

     ,FROM=device=serial

     [,FROMDD=ddname]

     ,TO=device=serial

     [,TODD=ddname]

     [,UNCATLG]

     [,RENAME=newname]

     [,UNLOAD]

     [,COPYAUTH]
```

The MOVE DSNAME control statement is used to move a dataset. The source dataset is scratched. The parameters are:

DSNAME=name

Specifies the fully qualified name of the dataset to be moved.

FROM=device=serial

Specifies the device type and serial number of the volume on which the dataset resides if it is not cataloged. If the dataset is cataloged, FROM should not be specified.

FROMDD=ddname

Specifies the name of the DD statement from which DCB and LABEL information (except dataset sequence number), for input datasets on tape volumes, can be obtained. When FROMDD is used for a partitioned dataset, the tape dataset must be an unloaded version of a partitioned dataset. The FROMDD operand can be omitted, provided the dataset has standard labels
and resides on a 9-track tape volume.

TO=device=serial

Specifies the device type and volume serial number of the volume to which the dataset is to be moved.

TODD=ddname

Specifies the name of a DD statement from which DCB (except RECFM, BLKSIZE and LRECL) and LABEL (except dataset sequence number) information for output datasets on tape volumes, can be obtained.

UNCATLG

Specifies that the catalog entry pertaining to the source dataset is to be removed. This parameter should be used only if the source dataset is cataloged. If the volume is identified by FROM, UNCATLG is ignored. Alias entries in VSAM catalogs for the source datasets are lost and can be replaced with Access Method Services if the datasets are later cataloged. For a MOVE operation, UNCATLG inhibits cataloging of the output dataset.

RENAME=newname

Specifies that the dataset is to be renamed, and indicates the new name.

UNLOAD

Specifies that the dataset is to be unloaded to the receiving volume(s).

COPYAUTH

Specifies that the receiving dataset is to be given the same access list as the input dataset, if the input dataset is RACF-protected.

```
MOVE PDS=name

     ,FROM=device=serial
     [,FROMDD=ddname]

     ,TO=device=serial

     [,TODD=ddname]

     [,UNCATLG]
```

```
[,RENAME=newname]

[,UNLOAD]

[,EXPAND=nn]

[,COPYAUTH]
```

The MOVE PDS control statement is used to move partitioned datasets. When used in conjunction with INCLUDE, EXCLUDE, REPLACE, or SELECT statements, the MOVE PDS statement can be used to merge selected members of several partitioned datasets or to delete members. If IEHMOVE is used to allocate space for an output partitioned dataset, the MOVE PDS statement can be used to expand a partitioned directory. If the receiving volume contains a partitioned dataset with the same name, the two datasets are merged. The source dataset is scratched. The parameters are:

PDS=name

Specifies the fully qualified name of the partitioned dataset to be moved or copied.

FROM=device=serial

Specifies the device type and serial number of the volume on which the dataset resides if it is not cataloged. If the dataset is cataloged, FROM should not be specified.

FROMDD=ddname

Specifies the name of the DD statement from which DCB and LABEL information (except dataset sequence number), for input datasets on tape volumes, can be obtained. When FROMDD is used for a partitioned dataset, the tape dataset must be an unloaded version of a partitioned dataset. The FROMDD operand can be omitted, provided the dataset has standard labels
and resides on a 9-track tape volume.

TO=device=serial

Specifies the device type and volume serial number of the volume to which the partitioned dataset is to be moved.

TODD=ddname

Specifies the name of a DD statement from which DCB (except RECFM, BLKSIZE and LRECL) and LABEL (except dataset sequence number) information for output datasets on tape volumes, can be obtained.

UNCATLG

Specifies that the catalog entry pertaining to the source partitioned dataset is to be removed. This parameter should be used only if the source dataset is cataloged. If the volume is identified by FROM, UNCATLG is ignored. Alias entries in VSAM catalogs for the source datasets are lost and can be replaced with Access Method Services if the datasets are later cataloged. For a MOVE operation, UNCATLG inhibits cataloging of the output dataset.

RENAME=newname

Specifies that the dataset is to be renamed, and indicates the new name.

UNLOAD

Specifies that the dataset is to be unloaded to the receiving volume(s).

EXPAND=nn

Specifies the number of 256-byte records (up to 99 decimal) to be added to the directory of the specified partitioned dataset. EXPAND will be ignored if space is previously allocated.

COPYAUTH

Specifies that the receiving dataset is to be given the same access list as the input dataset, if the input dataset is RACF-protected.

```
MOVE DSGROUP=name

     ,TO=device=serial

     [,TODD=ddname]

     [,UNCATLG]

     [,UNLOAD]

     [,COPYAUTH]
```

The MOVE DSGROUP control statement is used to move groups of datasets whose names are partially qualified by one or more identical names. Source datasets are scratched. Dataset groups to be moved must reside on direct access volumes. Only datasets that could be moved by MOVE DSNAME or MOVE PDS can be moved by MOVE DSGROUP. Alias entries in VSAM catalogs for the datasets are lost and can be replaced with Access Method Services. The parameters are:

DSGROUP=name

Specifies the cataloged dataset(s) to be moved. If name is a fully qualified dataset name, that dataset is not moved. If name is one or more qualifiers, all datasets whose names are qualified by name are moved.

TO=device=serial

Specifies the device type and volume serial number of the volume to which the dataset(s) is to be moved.

TODD=ddname

Specifies the name of a DD statement from which DCB (except RECFM, BLKSIZE and LRECL) and LABEL (except dataset sequence number) information for output datasets on tape volumes, can be obtained.

UNCATLG

Specifies that the catalog entry pertaining to the source dataset(s) is to be removed. This parameter should be used only if the source dataset(s) is cataloged. If the volume is identified by FROM, UNCATLG is ignored. Alias entries in VSAM catalogs for the source dataset(s) are lost and can be replaced with Access Method Services if the dataset(s) are later cataloged. For a MOVE operation, UNCATLG inhibits cataloging of the output dataset(s).

UNLOAD

Specifies that the dataset is to be unloaded to the receiving volume(s).

COPYAUTH

Specifies that the receiving dataset is to be given the same access list as the input dataset, if the input dataset is RACF-protected.

```
MOVE VOLUME=device=serial

     ,TO=device=serial

     [,TODD=ddname]

     [,UNLOAD]

     [,COPYAUTH]
```

The MOVE VOLUME is used to move all the datasets residing on a specified volume. Datasets to be moved must reside on direct access volumes. The parameters are:

VOLUME=device=serial

Specifies the device type and volume serial number of the source volume.

TO=device=serial

Specifies the device type and volume serial number of the volume to which the datasets are to be moved.

TODD=ddname

Specifies the name of a DD statement from which DCB (except RECFM, BLKSIZE and LRECL) and LABEL (except dataset sequence number) information for output datasets on tape volumes, can be obtained.

UNLOAD

Specifies that the dataset is to be unloaded to the receiving volume(s).

COPYAUTH

Specifies that the receiving dataset is to be given the same access list as the input dataset, if the input dataset is RACF-protected.

```
COPY DSNAME=name

     ,FROM=device=serial

     [,FROMDD=ddname]

     ,TO=device=serial

     [,TODD=ddname]

     [,UNCATLG]

     [,CATLG]

     [,RENAME=newname]

     [,UNLOAD]

     [,COPYAUTH]
```

The COPY DSNAME is used to copy a dataset. The source dataset, if cataloged, remains cataloged unless UNCATLG or CATLG is specified. The parameters are:

DSNAME=name

Specifies the fully qualified name of the dataset to be copied.

FROM=device=serial

Specifies the device type and serial number of the volume on which the dataset resides if it is not cataloged. If the dataset is cataloged, FROM should not be specified.

FROMDD=ddname

Specifies the name of the DD statement from which DCB and LABEL information (except dataset sequence number), for input datasets on tape volumes, can be obtained. When FROMDD is used for a partitioned dataset, the tape dataset must be an unloaded version of a partitioned dataset. The FROMDD operand can be omitted, provided the dataset has standard labels and resides on a 9-track tape volume.

TO=device=serial

Specifies the device type and volume serial number of the volume to which the dataset is to be copied.

TODD=ddname

Specifies the name of a DD statement from which DCB (except RECFM, BLKSIZE and LRECL) and LABEL (except dataset sequence number) information for output datasets on tape volumes, can be obtained.

UNCATLG

Specifies that the catalog entry pertaining to the source dataset is to be removed.

This parameter should be used only if the source dataset is cataloged. If the volume is identified by FROM, UNCATLG is ignored. Alias entries in VSAM catalogs for the source datasets are lost and can be replaced with Access Method Services if the datasets are later cataloged.

CATLG

Specifies that the copied dataset(s) is to be cataloged. The cataloging is done in the VSAM master/JOBCAT/STEPCAT catalog. If the RENAME and FROM operands are omitted, the source dataset(s) is uncataloged to permit the copied dataset(s) to be cataloged.

RENAME=newname

Specifies that the dataset is to be renamed, and indicates the new name.

UNLOAD

Specifies that the dataset is to be unloaded to the receiving volume(s).

COPYAUTH

Specifies that the receiving dataset is to be given the same access list as the input dataset, if the input dataset is RACF-protected.

```
COPY PDS=name

     ,FROM=device=serial

     [,FROMDD=ddname]

     ,TO=device=serial

     [,TODD=ddname]

     [,UNCATLG]

     [,CATLG]

     [,RENAME=newname]

     [,EXPAND=nn]
     [,COPYAUTH]
```

The COPY PDS control statement is used to copy partitioned datasets. When used in conjunction with INCLUDE, EXCLUDE, REPLACE, or SELECT statements, the COpy PDS statement can be used to merge selected members of several partitioned datasets or to delete members. If IEHMOVE is used to allocate space for an output partitioned dataset, the COPY PDS statement can be used to expand a partitioned directory. If the receiving volume already contains a partitioned dataset with the same name, the two are merged. The parameters are:

PDS=name

Specifies the fully qualified name of the partitioned dataset to be moved or copied.

FROM=device=serial

Specifies the device type and serial number of the volume on which the dataset resides if it is not cataloged. If the dataset is cataloged, FROM should not be specified.

FROMDD=ddname

Specifies the name of the DD statement from which DCB and LABEL information (except dataset sequence number), for input datasets on tape volumes, can be obtained. When FROMDD is used for a partitioned dataset, the tape dataset must be an unloaded version of a partitioned dataset. The FROMDD operand can be omitted, provided the dataset has standard labels
and resides on a 9-track tape volume.

TO=device=serial

Specifies the device type and volume serial number of the volume to which the partitioned dataset is to be copied.

TODD=ddname

Specifies the name of a DD statement from which DCB (except RECFM, BLKSIZE and LRECL) and LABEL (except dataset sequence number) information for output datasets on tape volumes, can be obtained.

UNCATLG

Specifies that the catalog entry pertaining to the source partitioned dataset is to be removed. This parameter should be used only if the source dataset is cataloged. If the volume is identified by FROM, UNCATLG is ignored. Alias entries in VSAM catalogs for the source datasets are lost and can be replaced with Access Method Services if the datasets are later cataloged.

CATLG

Specifies that the copied dataset(s) is to be cataloged. The cataloging is done in the VSAM master/JOBCAT/STEPCAT catalog. If the RENAME and FROM operands are omitted, the source dataset(s) is uncataloged to permit the copied dataset(s) to be cataloged.

RENAME=newname

Specifies that the dataset is to be renamed, and indicates the new name.

EXPAND=nn

Specifies the number of 256-byte records (up to 99 decimal) to be added to the directory of the specified partitioned dataset. EXPAND cannot be specified if space is previously allocated.

COPYAUTH

Specifies that the receiving dataset is to be given the same access list as the input

dataset, if the input dataset is RACF-protected.

```
COPY DSGROUP=name

     ,TO=device=serial

     [,TODD=ddname]

     [,UNCATLG]

     [,CATLG]

     [,UNLOAD]

     [,COPYAUTH]
```

The COPY DSGROUP is used to copy groups of datasets whose names are partially qualified by one or more identical names. Only datasets that can be copied with COPY DSNAME or COPY PDS can be copied with COPY DSGROUP. Dataset groups to be copied must reside on DASD volumes. The parameters are:

DSGROUP=name

Specifies the cataloged dataset(s) to be copied. If name is a fully qualified dataset name, that dataset is not copied. If name is one or more qualifiers, all datasets whose names are qualified by name are copied.

TO=device=serial

Specifies the device type and volume serial number of the volume to which the dataset(s) is to be copied.

TODD=ddname

Specifies the name of a DD statement from which DCB (except RECFM, BLKSIZE and LRECL) and LABEL (except dataset sequence number) information for output datasets on tape volumes, can be obtained.

UNCATLG

Specifies that the catalog entry pertaining to the source dataset(s) is to be removed. This parameter should be used only if the source dataset(s) is cataloged. If the volume is identified by FROM, UNCATLG is ignored. Alias entries in VSAM catalogs for the source dataset(s) are lost and can be replaced with Access Method Services if the dataset(s) are later cataloged.

CATLG

Specifies that the copied dataset(s) is to be cataloged. The cataloging is done in the VSAM master/JOBCAT/STEPCAT catalog.

UNLOAD

Specifies that the dataset is to be unloaded to the receiving volume(s).

COPYAUTH

Specifies that the receiving dataset is to be given the same access list as the input dataset, if the input dataset is RACF-protected.

```
COPY VOLUME=device=serial

     ,TO=device=serial

     [,TODD=ddname]

     [,CATLG]

     [,UNLOAD]

     [,COPYAUTH]
```

The COPY VOLUME is used to copy all the datasets residing on a specified volume. Datasets to be copied must reside on direct access volumes. The parameters are:

VOLUME=device=serial

Specifies the device type and volume serial number of the source volume.

TO=device=serial

Specifies the device type and volume serial number of the volume to which the datasets are to be copied.

TODD=ddname

Specifies the name of a DD statement from which DCB (except RECFM, BLKSIZE and LRECL) and LABEL (except dataset sequence number) information for output datasets on tape volumes, can be obtained.

CATLG

Specifies that the copied dataset(s) is to be cataloged. The cataloging is done in the VSAM master/JOBCAT/STEPCAT catalog. In my experience, under MVS, CATLG will fail because the source dataset(s) are not uncataloged.

UNLOAD

Specifies that the dataset is to be unloaded to the receiving volume(s).

COPYAUTH

Specifies that the receiving dataset is to be given the same access list as the input dataset, if the input dataset is RACF-protected.

Program Parameter (PARM= on EXEC statement)

The optional PARM values which may be specified are PARM='[POWER=n][,LINECNT=xx]'. The values are:

POWER=n

Used to request that the normal amount of space allocated for work areas be increased n times (1 to 999). The POWER parameter is used when 750 or more members are being moved or copied. The progression for the value of n is:

- POWER=2 when 750 to 1,500 members are to be moved or copied.
- POWER=3 when 1,501 to 2,250 members are to be moved or copied.
- POWER=4 when 2,251 to 3,000 members are to be moved or copied.

If POWER = 2, the work space requirement on the SYSUT1 volume is two times the basic requirement; if POWER=3, work space requirement is three times the basic requirement, etc. For example, if POWER=2, three areas of 26, 26, and 52 contiguous tracks on a 2314 must be available. If a direct access device other than a 2314 is used, an equivalent amount of space must be available.

LINECNT=xx

Specifies the number of lines per page in the listing of the SYSPRINT dataset; xx is a two-digit number in the range 04 through 99.

Examples

Copy partitioned dataset from one DASD to another DASD, recataloging

```
//IEHMOVE JOB (001),'IEHMOVE-1',CLASS=A,MSGCLASS=X
//IEHMOVE EXEC PGM=IEHMOVE,REGION=4096K,PARM='POWER=9'
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=3390,VOL=SER=PUB001,DISP=OLD
//DDFROM   DD  UNIT=3380,VOL=SER=MVS380,DISP=OLD
//DDTO     DD  UNIT=3380,VOL=SER=MVS381,DISP=OLD
//SYSIN    DD  *
  COPY PDS=JAY01.FILE807.PDS,FROM=3380=MVS380,TO=3380=MVS381,          C
               UNCATLG,CATLG
//
```

Copy all datasets from one DASD volume to another DASD volume

```
//IEHMOVE JOB (001),'IEHMOVE-2',CLASS=A,MSGCLASS=X
//IEHMOVE EXEC PGM=IEHMOVE,REGION=4096K,PARM='POWER=9'
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=3390,VOL=SER=PUB001,DISP=OLD
//DDFROM   DD  UNIT=3380,VOL=SER=MVS381,DISP=OLD
//DDTO     DD  UNIT=3380,VOL=SER=SYSP01,DISP=OLD
//SYSIN    DD  *
  COPY VOLUME=3380=MVS380,TO=3380=SYSP01
//
```

Move all datasets matching HLQ from one DASD to another DASD, recataloging

```
//IEHMOVE JOB (001),'IEHMOVE-3',CLASS=A,MSGCLASS=X
//IEHMOVE EXEC PGM=IEHMOVE,REGION=4096K,PARM='POWER=9'
//SYSPRINT DD  SYSOUT=*
```

```
//SYSUT1   DD  UNIT=3390,VOL=SER=PUB001,DISP=OLD
//DDFROM   DD  UNIT=3380,VOL=SER=PUB000,DISP=OLD
//DDTO     DD  UNIT=3380,VOL=SER=MVS381,DISP=OLD
//SYSIN    DD  *
  MOVE DSGROUP=HMVS01.PROJECT1,                                        C
               FROM=3380=PUB000,TO=3380=MVS381
//
```

Unload all datasets matching HLQ from DASD to tape

```
//IEHMOVE JOB (001),'IEHMOVE-4',CLASS=A,MSGCLASS=X
//IEHMOVE EXEC PGM=IEHMOVE,REGION=4096K,PARM='POWER=9'
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=3390,VOL=SER=PUB001,DISP=OLD
//DDFROM   DD  UNIT=3380,VOL=SER=PUB000,DISP=OLD
//DDTO     DD  UNIT=TAPE,VOL=SER=22005M,DISP=(NEW,KEEP),LABEL=(1,SL)
//SYSIN    DD  *
  COPY DSGROUP=HMVS01,                                                 C
               FROM=3380=PUB000,TO=TAPE=22005M,UNLOAD
//
```

Reload partitioned dataset from tape to DASD

```
//IEHMOVE JOB (001),'IEHMOVE-5',CLASS=A,MSGCLASS=X
//IEHMOVE EXEC PGM=IEHMOVE,REGION=4096K,PARM='POWER=9'
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=3390,VOL=SER=PUB001,DISP=OLD
//DDIN     DD  UNIT=TAPE,DSN=JAY01.FILE046.PDS,DISP=OLD,
//             VOL=SER=22005M,LABEL=(4,SL)
//DDOUT    DD  UNIT=3380,VOL=SER=MVS380,DISP=OLD
//SYSIN    DD  *
  COPY DSNAME=HMVS01.FILE046.PDS,                                      C
               FROM=TAPE=(22005M,4),TO=3380=MVS380
//
```

Source: GC26-3902-1 OS/VS2 MVS Utilities Release 3.8

IEHPROGM **[supplied by IBM, located in SYS1.LINKLIB]**

Used to modify system control data and to maintain datasets at an organizational level. IEHPROGM can be used to:

- Scratch a dataset or a member.
- Rename a dataset or a member.
- Catalog or uncatalog a non-VSAM dataset.
- Maintain dataset passwords.

IEHPROGM requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines a sequential input dataset that contains IEHPROGM control statements.
- anyname1 - defines the DASD volume on which affected datasets reside.

Control Statements

Control statements for IEHPROGM follow the same rules as JCL; i.e., if a statement requires continuation onto a following record, the operands on the record to be continued are stopped following a comma before reaching column 72, a non-blank character is placed in column 72, and the statement continues on the following record beginning in

column 16.

```
SCRATCH   {DSNAME=name | VTOC}

          [,VOL=device=list]

          [,PURGE]

          [,MEMBER=name]

          [,SYS]
```

The SCRATCH control statement is used to scratch a dataset or member from a DASD volume. A dataset or member is scratched only from the volumes designated in the SCRATCH statement. This function does not uncatalog scratched datasets. The parameters are:

DSNAME=name | VTOC

DSNAME=name

Specifies the fully qualified name of either the dataset to be scratched or the partitioned dataset that contains the member to be scratched. The qualified name must not exceed 44 characters, including delimiters.

VTOC

Specifies that all datasets on the specified volume, except those protected by a password or those whose expiration dates have not expired, are to be scratched. Password-protected datasets are scratched if the correct password is provided. The effect of VTOC is modified when it is used with PURGE or SYS. **Extreme caution is advised when specifying VTOC and PURGE together; the result will be the removal of all datasets on the volume with no warning given nor confirmation required.**

VOL=device=list

Specifies the device type and serial number(s) of the volume(s), limited to 50, that contain the dataset(s).

If VTOC or MEMBER is specified, VOL cannot specify more than one volume. Caution should be used when specifying VTOC if VOL specifies the system residence volume.

PURGE

Specifies that each dataset specified by DSNAME or VTOC be scratched, even if its expiration date has not elapsed. Default: The specified datasets are scratched only if their expiration dates have elapsed.

MEMBER=name

Specifies a member name or alias of a member (in the named dataset) to be removed from the directory of a partitioned dataset. This name is not validity-checked because all members must be accessible, whether the name is valid or not.

SYS

Specifies that datasets that have names that begin with "AAAAAAAA.AAAAAAAA.AAAAAAAA.AAAAAAAA." or "SYSnnnnn.T" and "P" or "V" in position 19 are to be scratched. These are names assigned to datasets by the operating system. This parameter is valid only when VTOC is specified. If the name of the dataset to be scratched begins with SYS, nnnnn is the date.

```
RENAME   DSNAME=name

         ,VOL=device=list

         ,NEWNAME=name

         [,MEMBER=name]
```

The RENAME control statement is used to change the name or alias of a dataset or member residing on a DASD volume. The name is changed only on the designated volume(s). The rename operation does not update the catalog. The parameters are:

DSNAME=name

Specifies the fully qualified name of either the dataset to be renamed or the partitioned dataset that contains the member to be renamed. The qualified name must not exceed 44 characters, including delimiters.

VOL=device=list

Specifies the device type and serial number(s) of the volume(s), limited to 50, that contain the dataset(s).

If MEMBER is specified, VOL cannot specify more than one volume.

NEWNAME=name

Specifies the new fully qualified name for the dataset, or the new member or alias.

MEMBER=name

Specifies a member name or alias of a member (in the named dataset) to be renamed. This name is not validity-checked because all members must be accessible, whether the name is valid or not.

```
CATLG   DSNAME=name

        ,VOL=device=list
```

The CATLG control statement is used to add an entry for a non-VSAM dataset to a catalog. The parameters are:

DSNAME=name

Specifies the fully qualified name of either the dataset to be cataloged. The qualified name must not exceed 44 characters, including delimiters.

VOL=device=list

Specifies the device type and serial number(s) of the volume(s), limited to 50, that contain the dataset(s).

The volume serial numbers must appear in the same order in which they were originally encountered (in DD statements within the input stream) when the dataset was created.

```
UNCATLG DSNAME=name
```

The UNCATLG control statemetn is used to remove an entry for a non-VSAM dataset from a catalog. The parameter is:

DSNAME=name

Specifies the fully qualified name of either the dataset to be uncataloged. The qualified name must not exceed 44 characters, including delimiters.

```
ADD  DSNAME=name

     [,PASWORD2=new-password]

     [,CPASWORD=control-password]

     [,TYPE=code]

     [,VOL=device=list]

     [,DATA='user-data']
```

The ADD control statement is used to add a password entry in the PASSWORD dataset. When the control entry for a direct access, online dataset is added, the indicated protection status of the dataset is set in the DSCB; when a secondary entry is added, the protection status in the DSCB is not changed. The parameters are:

DSNAME=name

Specifies the fully qualified name of either the dataset whose password entry is to be assigned. The qualified name must not exceed 44 characters, including delimiters.

PASWORD1=current-password

PASWORD2=new-password

Specifies the new password to be added or assigned to the entry. If the password is not to be changed, the current password must also be specified as the new password. The password can consist of one to eight alphameric characters.

Default: The operator is prompted for a new password.

CPASWORD=control-password

Specifies the control password for the dataset. CPASWORD must be specified unless this is the first password assigned to the dataset, in which case PASWORD2 specifies the password to be added.

TYPE=code

Specifies the protection code of the password, and if a control password entry is to be changed for or assigned to a direct access, online dataset, specifies the protection status of the dataset. The values that can be specified for code are:

- 1 - Specifies that the password is to allow both read and write access to the dataset; if a control password is being assigned or changed, read/write protection is set in the DSCB.
- 2 - Specifies that the password is to allow only read access to the dataset; if control password is assigned or changed, read/write protection is set in the DSCB.
- 3 - Specifies that the password is to allow b0th read and write access to the dataset; if a control password is being assigned or changed, read-without-password protection is set in the DSCB.

Default: For ADD, if this parameter is omitted the new password is assigned the same protection code as the control password for the dataset. If a control password is being "added," TYPE=3 is the default.

VOL=device=list

Specifies the device type and serial number(s) of the volume(s), limited to 50, that contain the dataset(s).

If omitted, the protection status in the DSCB is not set or changed, unless the dataset is cataloged. This parameter is not necessary for secondary password entries, or if the desired protection status in the DSCB is already set or is not to be changed by ADD or REPLACE.

DATA='user-data'

Specifies the user data is to be placed in the password entry. The user data has a maximum length of 77 bytes and must be enclosed in apostrophes. If DATA is omitted from an ADD operation, 77 blanks are used. If DATA is omitted from a REPLACE operation, current user data is not changed.

```
REPLACE   DSNAME=name

          [,PASWORD1=current-password]

          [,PASWORD2= new-password]

          [,CPASWORD=control-password]
```

```
[,TYPE=code]

[,VOL=device=list]

[,DATA='user-data']
```

The REPLACE control statement is used to replace any or all of the following information in a password entry: the password name, protection mode (read/write or read only) of the password, and user data. When the control entry for a direct access, online dataset is replaced, the protection status of the dataset is changed in the DSCB if necessary; when a secondary entry is replaced, the protection status in the DSCB is not changed. The parameters are:

DSNAME=name

Specifies the fully qualified name of either the dataset whose password entry is to be changed. The qualified name must not exceed 44 characters, including delimiters.

PASWORD1=current-password

Specifies the password in the entry to be changed. Default: The operator is prompted for the current password.

PASWORD2=new-password

Specifies the new password to be added or assigned to the entry. If the password is not to be changed, the current password must also be specified as the new password. The password can consist of one to eight alphameric characters. Default: The operator is prompted for a new password.

CPASWORD=control-password

Specifies the control password for the dataset. CPASWORD must be specified unless the control entry is being changed or deleted, in which case PASWORD1 specifies the control password.

TYPE=code

Specifies the protection code of the password, and if a control password entry is to be changed for or assigned to a direct access, online dataset, specifies the protection status of the dataset. The values that can be specified for code are:

- 1 - Specifies that the password is to allow both read and write access to the dataset; if a control password is being assigned or changed, read/write protection is set in the DSCB.
- 2 - Specifies that the password is to allow only read access to the dataset; if control password is assigned or changed, read/write protection is set in the DSCB.
- 3 - Specifies that the password is to allow b0th read and write access to the dataset; if a control password is being assigned or changed, read-without-password protection is set in the DSCB.

Default: For REPLACE, the protection is not changed.

VOL=device=list

Specifies the device type and serial number(s) of the volume(s), limited to 50, that contain the dataset(s).

If omitted, the protection status in the DSCB is not set or changed, unless the dataset is cataloged. This parameter is not necessary for secondary password entries, or if the desired protection status in the DSCB is already set or is not to be changed by ADD or REPLACE.

DATA='user-data'

Specifies the user data is to be placed in the password entry. The user data has a maximum length of 77 bytes and must be enclosed in apostrophes. If DATA is omitted from a REPLACE operation, current user data is not changed.

```
DELETEP   DSNAME=name

          [,PASWORD1=current-password]

          [,CPASWORD=control-password]

          [,VOL=device=list]
```

The DELETEP control statement is used to delete an entry in the PASSWORD dataset. If a control entry is deleted, all the secondary entries for that dataset are also deleted. If a secondary entry is deleted, only that entry is deleted. When the control entry for a direct access, online dataset is deleted, the protection status in the DSCB is set to indicate that the dataset is no longer protected. The parameters are:

DSNAME=name

Specifies the fully qualified name of either the dataset whose password entry is to be deleted. The qualified name must not exceed 44 characters, including delimiters.

PASWORD1=current-password

Specifies the password in the entry to be deleted. Default: The operator is prompted for the current password.

CPASWORD=control-password

Specifies the control password for the dataset. CPASWORD must be specified unless the control entry is being changed or deleted, in which case PASWORD1 specifies the control password.

VOL=device=list

Specifies the device type and serial number(s) of the volume(s), limited to 50, that contain the dataset(s).

If omitted, the protection status in the DSCB is not set or changed, unless the dataset is cataloged. This parameter is not necessary for secondary password entries, or if the desired protection status in the DSCB is already set or is not to be

changed by ADD or REPLACE.

```
LIST   DSNAME=name

       ,PASWORD1=current-password
```

The LIST control statement is used to format and print information from a password entry. The parameters are:

DSNAME=name

Specifies the fully qualified name of the dataset whose password entry is to be listed. The qualified name must not exceed 44 characters, including delimiters.

PASWORD1=current-password

Specifies the password in the entry to be listed. Default: The operator is prompted for the current password.

Examples

Scratch all expired datasets, including SYS datasets

```
//IEHPROGM JOB (001),'IEHPROGM-1',CLASS=S,MSGCLASS=X
//PROGM    EXEC PGM=IEHPROGM
//SYSPRINT  DD SYSOUT=*
//DD1       DD UNIT=3350,VOL=SER=WORK00,DISP=OLD
//SYSIN     DD  *
   SCRATCH VTOC,VOL=3350=WORK00
//
```

Remove all datasets on volume, regardless of expiration status

```
//IEHPROGM JOB (001),'IEHPROGM-2',CLASS=S,MSGCLASS=X
//PROGM    EXEC PGM=IEHPROGM
//SYSPRINT  DD SYSOUT=*
//DD1       DD UNIT=3380,VOL=SER=MVS380,DISP=OLD
//SYSIN     DD  *
   SCRATCH VTOC,PURGE,VOL=3380=MVS380
//
```

Scratch and uncatalog two datasets

```
//IEHPROGM JOB (001),'IEHPROGM-3',CLASS=S,MSGCLASS=X
//PROGM    EXEC PGM=IEHPROGM
//SYSPRINT  DD SYSOUT=*
//DD1       DD UNIT=3380,VOL=SER=MVS380,DISP=OLD
//SYSIN     DD  *
   UNCATLG JAY01.CUSTOMER.MASTER
   SCRATCH JAY01.CUSTOMER.MASTER,VOL=3380=MVS380
   UNCATLG JAY01.FILE807.PDS
   SCRATCH JAY01.FILE807.PDS,VOL=3380=MVS380
//
```

Rename and recatalog a dataset

```
//IEHPROGM JOB (001),'IEHPROGM-4',CLASS=S,MSGCLASS=X
```

```
//PROGM    EXEC PGM=IEHPROGM
//SYSPRINT  DD SYSOUT=*
//DD1       DD UNIT=3380,VOL=SER=MVS380,DISP=OLD
//SYSIN     DD  *
   RENAME DSNAME=JAY01.SIMULAO.LINKLIB,NEWNAME=JAY01.SIMULAN.LINKLIB,  C
               VOL=3380=MVS380
   UNCATLG JAY01.SIMULAO.LINKLIB
   CATLG JAY01.SIMULAN.LINKLIB
//
```

Add passwords to a dataset

```
//IEHPROGM JOB (001),'IEHPROGM-5',CLASS=S,MSGCLASS=X
//PROGM    EXEC PGM=IEHPROGM
//SYSPRINT  DD SYSOUT=*
//DD1       DD UNIT=3380,VOL=SER=MVS380,DISP=OLD
//SYSIN     DD  *
   ADD DSNAME=JAY01.STUDENT.MASTER,PASWORD2=KEY,TYPE=1,                C
               DATA='SECONDARY IS READ'
   ADD DSNAME=JAY01.STUDENT.MASTER,CPASWORD=KEY,TYPE=2,                C
               PASWORD2=READ,DATA='ASSIGNED TO J.DOE'
//
```

List passwords for a dataset

```
//IEHPROGM JOB (001),'IEHPROGM-6',CLASS=S,MSGCLASS=X
//PROGM    EXEC PGM=IEHPROGM
//SYSPRINT  DD SYSOUT=*
//DD1       DD UNIT=3380,VOL=SER=MVS380,DISP=OLD
//SYSIN     DD  *
   LIST DSNAME=JAY01.STUDENT.MASTER,PASWORD1=KEY
//
```

Rename a member of a Partitioned dataset

```
//IEHPROGM JOB (001),'IEHPROGM-7',CLASS=S,MSGCLASS=X
//PROGM    EXEC PGM=IEHPROGM
//SYSPRINT  DD SYSOUT=*
//DD1       DD UNIT=3380,VOL=SER=MVS381,DISP=OLD
//SYSIN     DD  *
   RENAME DSNAME=JAY01.ISAM.COBOL,VOL=3380=MVS381,MEMBER=ISREPT,       C
               NEWNAME=ISREPT1
//
```

Source: GC26-3902-1 OS/VS2 MVS Utilities Release 3.8

LISTPDS **[written by Gene Czarcinski, located in SYSC.LINKLIB]**

Used for listing and/or punching source libraries or datasets. LISTPDS generates formatted listings of a PDS directory, the contents of processed members or sequential files, as well as (optionally) punching processed members and/or sequential datasets. Records may be generated which allow the output to be used in conjunction with the IBM utility IEBUPDTE.

LISTPDS is located in File #316 of the CBT tape and was developed by Gene Czarcinski at the NASA/Goddard Space Flight Center. LISTPDS requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSLIST - defines a sequential output dataset. Will contain the listing of the requested datasest. Required when the LIST option is used.

- SYSPUNCH - defines a sequential output dataset. Required when either DECK or UPDTE options are used.
- SYSLIB - defines the Partitioned or sequential dataset(s) to be processed.
- SYSIN - defines a sequential input dataset that contains statements used to specify individual members to process. Only the first partitioned dataset of SYSLIB will be used. When SYSIN is empty, DUMMY, or not included, all members of SYSLIB are processed. If the EXCLUDE option is specified, all members other than those specified in the SYSIN file will be listed. The EXCLUDE/SELECT option is ignored if SYSIN is a null dataset or if the NOSEL option is specified.

Program Parameter (PARM= on EXEC statement)

The execution of LISTPDS and which options are used are entered as keywords in the parameter field. The keywords, their function and defaults are:

LIST | NOLIST

- LIST - Produce a listing of the contents of the datasets in the SYSLIB file on SYSLIST. The DD statement SYSLIST is required. The default is LIST.
- NOLIST - Do not produce a listing of datasets and/or members.

DECK | NODECK

- DECK - Reproduce card images of the contents of the members in the SYSLIB file on SYSPUNCH. The SYSPUNCH DD statement is required.
- NODECK - Do not reproduce card images. The default is NODECK.

LISTDIR

- Produces only a listing of the directory entries for the members. No record count is given and less I/O time is required. The default is no directory listing.

UPDTE | NOUPDTE

- UPDTE - IEBUPDTE utility ADD statements will be produced on SYSPUNCH. When DECK is also specified they will precede each member. If a member contains IEBUPDTE statements, the './' statements within the member will be written with '><' in place of the './' characters. The SYSPUNCH DD statement is required.
- NOUPDTE - IEBUPDTE utility ADD statements will not be produced. The default is NOUPDTE.

SSI | NOSSI

- SSI - The system status information is included on the IEBUPDTE ADD statements.
- NOSSI - The system status information is not placed on the IEBUPDTE ADD statements. The default is NOSSI.

SPF | NOSPF

- SPF - If ISPF statistics exist, they are included on the IEBUPDTE ADD statements. The default is SPF.
- NOSPF - ISPF statistics are not included on the IEBUPDTE ADD statements.

TRUNC | NOTRUNC

- TRUNC - For partitioned datasets with records longer than 100 bytes, only the first 100 bytes are listed on SYSLIST. The default is TRUNC.
- NOTRUNC - For partitioned datasets with records longer than 100 bytes, the entire record is listed on SYSLIST. This will require more than on line per record listed.

LINECNT=nn

- The number of lines printed per page for SYSPRINT and SYSLIST is specified by nn. The default value is 60. This should be changed to 58 for devices which allow a maximum of 66 lines per page, such as a XEROX 1200 printer.

RITS | CRBE | CRJE

- Used to inform LISTPDS special processing is required due to 88-byte logical records. The default is to treat Partitioned datasets as not being either a RITS, CRBE or CRJE library.

HEXOUT | NOHEXOUT

- HEXOUT - Prints the records on SYSLIST in hexadecimal format. This is useful for non-source members. The default is NOHEXOUT.
- NOHEXOUT - Prints the records on SYSLIST as they exist; unprintable characters (not in the 64 character set) will be printed as blanks. The default is NOHEXOUT.

NUM | NONUM

- NUM - Prints the record number on the left margin followed by a space and then the remainder of the record image. Datasets with record format F, FB, or U are assumed to have the record number in the last 8 bytes of each record. Datasets with record format V or VB, are assumed to have the record number in the first 8 bytes of each record. This applies to both partitioned and sequential organization datasets.
- NONUM - Prints each record as they exist. The default for this option is NONUM.

XLATE | TEXT | NOXLATE | NOTEXT

- XLATE and TEXT - Lower case characters are translated to upper case before printing. The default is TEXT.
- NOXLATE and NOTEXT - Records are printed as they exist.

EJECT | NOEJECT

- EJECT - Each member begins on a new page. The default is EJECT.
- NOEJECT - Triple spacing is used to separate members.

SELECT | NOSEL | EXCLUDE

- SELECT - Only the member names specified in SYSIN will be processed. If a SYSIN DD is supplied, this option is assumed to be in effect, unless NOSEL is specified.
- NOSEL - All members are processed.
- EXCLUDE - The member names specified in SYSIN file will NOT be printed if this option is specified. All other members will be printed.

EROPT={TERM | ACC}

- TERM - Processing will be terminated if an I/O error occurs while reading SYSLIB.
- ACC - Processing will continue after an I/O error is encountered while reading SYSLIB. The bad records are ignored. The default is EROPT=ACC.

MAXLIST=nn

- The number specified will limit printing on SYSLIST to the first nn lines of each member. The default is print all records for each member.

Control Statements

There are no control statements. If specific members of a partitioned dataset are to be selected or excluded, their names are included in SYSIN. The names specified are separated by commas. As many statements as are required may be included in a single execution.

Examples

Print the directory of a partitioned dataset

```
//LISTPDS  JOB (1),'LISTPDS-1',CLASS=A,MSGCLASS=X
//LISTPDS  EXEC PGM=LISTPDS,PARM='NOLIST'
//SYSPRINT  DD  SYSOUT=*
//SYSLIST   DD  SYSOUT=*
//SYSLIB    DD  DISP=SHR,DSN=SYSC.PROCLIB
//SYSPUNCH  DD  SYSOUT=B
//
```

Print all members of a partitioned dataset

```
//LISTPDS  JOB (1),'LISTPDS-2',CLASS=A,MSGCLASS=X
//LISTPDS  EXEC PGM=LISTPDS,PARM='LIST'
//SYSPRINT  DD  SYSOUT=*
//SYSLIST   DD  SYSOUT=*
//SYSLIB    DD  DISP=SHR,DSN=SYSC.PROCLIB
//SYSPUNCH  DD  SYSOUT=B
//
```

Punch selected members of a partitioned dataset

```
//LISTPDS  JOB (1),'LISTPDS-3',CLASS=A,MSGCLASS=X
//LISTPDS  EXEC PGM=LISTPDS,PARM='NOLIST,DECK'
//SYSPRINT  DD  SYSOUT=*
//SYSLIST   DD  SYSOUT=*
//SYSLIB    DD  DISP=SHR,DSN=SYSC.PROCLIB
//SYSPUNCH  DD  SYSOUT=B
//SYSIN     DD  *
ASMGC,ASMGCL,ASMGCLG
COBUC,COBUCG,COBUCL,COBUCLG
//
```

Punch selected members of a partitioned dataset with IEBUPDTE cards

```
//LISTPDS  JOB (1),'LISTPDS-4',CLASS=A,MSGCLASS=X
//LISTPDS  EXEC PGM=LISTPDS,PARM='NOLIST,DECK,UPDTE,SPF'
//SYSPRINT  DD  SYSOUT=*
```

```
//SYSLIST   DD  SYSOUT=*
//SYSLIB    DD  DISP=SHR,DSN=JAY01.ISAM.COBOL
//SYSPUNCH  DD  SYSOUT=B
//SYSIN     DD  *
ISREPT1,ISREPT2
//
```

Punch and print sequential dataset

```
//LISTPDS  JOB (1),'LISTPDS-5',CLASS=A,MSGCLASS=X
//LISTPDS  EXEC PGM=LISTPDS,PARM='LIST,DECK'
//SYSPRINT  DD  SYSOUT=*
//SYSLIST   DD  SYSOUT=*
//SYSLIB    DD  DISP=SHR,DSN=JAY01.CBLPATCH.ZAPS
//SYSPUNCH  DD  SYSOUT=B
//
```

MINIUNZ **[distributed by Greg Price, located in SYSC.LINKLIB]**

Used for uncompressing the contents of datasets created by compression utilities commonly used on PCs and by the companion program MINIZIP.

MINIUNZ was written in the C-language and has been ported to MVS using the compiler JCC, from Jason Winter, which produces Assembler language output and requires no runtime libraries. This version is from File #135 of the CBT tape (version 502, 11/2021).

MINIUNZ requires the following DD statements:

- STDOUT - defines a sequential output message dataset.
- inputDD - defines the sequential input dataset containing the compressed data records.
- outputDD - defines the output partitioned or sequential dataset that will contain the extracted data records.

Program Parameter (PARM= on EXEC statement)

The parameter field is used to specify DD names to use for the extraction and may also be used to specify what data is to be extracted from the compressed archive. The format of the parameter field is:

[{-L | -V}] inputDD outputDD [name]

The values and their function are:

[{-L | -V}]

If one of these options is specified, a list of the archive contents is produced, but no datasets are extracted. The inputDD is required, but the outputDD may point to a DUMMY DD statement. Both options will list the contents of the archive, -V will also include the comment that is added when an archive is created under MVS, by the program MINIZIP; the comment indicates whether the data was stored as EBCDIC or converted to ASCII, plus the RECFM and LRECL of the data records.

inputDD

Specifies the name of the DD statement to be used to read the compressed archive.

outputDD

Specifies the name of the DD statement to be used to write the extracted records. If the -L or -V option is used, this DD may be specified as DUMMY. If the [name] parameter is omitted, the DD must specify a partitioned dataset, into which all members of the archive are extracted. If the [name] parameter is specified, the DD must specify a sequential dataset, into which only the single member is extracted.

[name]

Specifies the single member to be extracted from the archive.

Control Statements

There are no control statements for MINIUNZ.

Examples

List contents of archive

```
//UNZIP    JOB 1,'MINI UNZIP 1',CLASS=A,MSGCLASS=X
//MINIUNZ EXEC PGM=MINIUNZ,REGION=7M,
//             PARM='-L ZIPFILE PDSOUT'
//STDOUT    DD SYSOUT=*
//ZIPFILE   DD DISP=SHR,DSN=JAY01.ISAMCOB.ZIP
//PDSOUT    DD DUMMY
//
```

Extract entire contents of archive to a partitioned dataset

```
//UNZIP    JOB 1,'MINI UNZIP 2',CLASS=A,MSGCLASS=X
//MINIUNZ EXEC PGM=MINIUNZ,REGION=7M,
//             PARM='ZIPFILE PDSOUT'
//STDOUT    DD SYSOUT=*
//ZIPFILE   DD DISP=SHR,DSN=JAY01.ISAMCOB.ZIP
//PDSOUT    DD DISP=(,CATLG,DELETE),DSN=JAY01.ISAMCO.UNZIP2,
//             UNIT=SYSDA,VOL=SER=MVS381,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=23440),
//             SPACE=(CYL,(10,1,50),RLSE)
//
```

Extract a single member from an archive to a sequential dataset

```
//UNZIP    JOB 1,'MINI UNZIP 3',CLASS=A,MSGCLASS=X
//MINIUNZ EXEC PGM=MINIUNZ,REGION=7M,
//             PARM='ZIPFILE SEQOUT SORTNAME'
//STDOUT    DD SYSOUT=*
//ZIPFILE   DD DISP=SHR,DSN=JAY01.ISAMCOB.ZIP
//SEQOUT    DD DISP=(,CATLG,DELETE),DSN=JAY01.ISAMCO.SORTNAME.COBOL,
//             UNIT=SYSDA,VOL=SER=MVS381,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=23440),
//             SPACE=(CYL,(10,1),RLSE)
//
```

MINIZIP **[distributed by Greg Price, located in SYSC.LINKLIB]**

Used for compressing the contents of: 1) one or more sequential datasets or members of partitioned datasets; or 2) all the members of a one or more partitioned datasets, to create a smaller, portable copy of the original data. Datasets created by MINIZIP may be processed by compression utilities commonly used on PCs and by the companion program MINIUNZ to reconstruct the original datasets.

Note: Datasets created by MINIZIP may be viewed with REVIEW's Browse function, which will expand the compressed output an present it in a viewable format.

MINIZIP was written in the C-language and has been ported to MVS using the compiler JCC, from Jason Winter, which produces Assembler language output and requires no runtime libraries. This version is from File #135 of the CBT tape (version 502, 11/2021).

MINIZIP requires the following DD statements:

- STDOUT - defines a sequential output message dataset.
- SYSUT1 - defines a work dataset.
- inputDD - defines the partitioned or sequential input dataset containing the data records to be compressed.
- outputDD - defines the sequential output dataset that will contain the compressed data records.

Program Parameter (PARM= on EXEC statement)

The parameter field is used to specify DD names to use for the compression. The format of the parameter field is:

[-O] outputDD inputDD[inputDD...]

The values and their function are:

[-O]

Specifies that all inputDD statements refer to partitioned datasets and all members of each dataset are to be added to the archive.

outputDD

Specifies the name of the DD statement to be used to write the compressed archive.

inputDD[inputDD...]

Specifies the names of one or more DD statements from which data records are to be read and placed into the archive.

Control Statements

There are no control statements for MINIZIP.

Examples

Compress all members of a partitioned dataset to archive

```
//ZIP      JOB 1,'MINI ZIP 1',CLASS=A,MSGCLASS=X
//MINIZIP EXEC PGM=MINIZIP,REGION=7M,
//             PARM='-o ZIPFILE PDSIN'
//STDOUT    DD SYSOUT=*
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(30,30)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//ZIPFILE   DD DISP=(NEW,CATLG,DELETE),DSN=JAY01.ISAMCOB.ZIP,
//             UNIT=SYSDA,VOL=SER=MVS381,
//             SPACE=(TRK,(60,15),RLSE),
//             DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=27920)
//PDSIN     DD DISP=SHR,DSN=JAY01.ISAM.COBOL
//
```

Compress all members of three partitioned datasets to archive

```
//ZIP      JOB 1,'MINI ZIP 2',CLASS=A,MSGCLASS=X
//MINIZIP EXEC PGM=MINIZIP,REGION=7M,
//             PARM='-o ZIPFILE PDSIN1 PDSIN2 PDSIN3'
//STDOUT    DD SYSOUT=*
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(30,30)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//ZIPFILE   DD DISP=(NEW,CATLG,DELETE),DSN=JAY01.BASE.ZIP,
//             UNIT=SYSDA,VOL=SER=MVS381,
//             SPACE=(TRK,(60,15),RLSE),
//             DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=27920)
//PDSIN1    DD DISP=SHR,DSN=JAY01.CLIST
//PDSIN2    DD DISP=SHR,DSN=JAY01.CNTL
//PDSIN3    DD DISP=SHR,DSN=JAY01.EXEC
//
```

Compress three sequential datasets to archive

```
//ZIP      JOB 1,'MINI ZIP 3',CLASS=A,MSGCLASS=X
//MINIZIP EXEC PGM=MINIZIP,REGION=7M,
//             PARM='ZIPFILE SEQIN1 SEQIN2 SEQIN3'
//STDOUT    DD SYSOUT=*
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(30,30)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//ZIPFILE   DD DISP=(NEW,CATLG,DELETE),DSN=JAY01.IERAM1.ZIP,
//             UNIT=SYSDA,VOL=SER=MVS381,
//             SPACE=(TRK,(60,15),RLSE),
//             DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=27920)
//SEQIN1    DD DISP=SHR,DSN=SYSP.MVT.IERAM1.CNTL
//SEQIN2    DD DISP=SHR,DSN=SYSP.MVT.IERAM1.OBJECT
//SEQIN3    DD DISP=SHR,DSN=SYSP.MVT.IERCMZ.CNTL
//
```

Compress four members of a partitioned dataset to archive

```
//ZIP      JOB 1,'MINI ZIP 4',CLASS=A,MSGCLASS=X
//MINIZIP EXEC PGM=MINIZIP,REGION=7M,
//             PARM='ZIPFILE PDSIN1 PDSIN2 PDSIN3 PDSIN4'
//STDOUT    DD SYSOUT=*
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(30,30)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//ZIPFILE   DD DISP=(NEW,CATLG,DELETE),DSN=JAY01.RESTORE.ZIP,
//             UNIT=SYSDA,VOL=SER=MVS381,
//             SPACE=(TRK,(60,15),RLSE),
//             DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=27920)
//PDSIN1    DD DISP=SHR,DSN=SYSP.RESTORE.LOCAL.CNTL(UCSYSP01)
//PDSIN2    DD DISP=SHR,DSN=SYSP.RESTORE.LOCAL.CNTL(ADDUSERS)
//PDSIN3    DD DISP=SHR,DSN=SYSP.RESTORE.LOCAL.CNTL(S3LLIB)
```

```
//PDSIN4    DD DISP=SHR,DSN=SYSP.RESTORE.LOCAL.CNTL(RESTORE)
//
```

OFFLOAD **[written by David B. Cole, located in SYSC.LINKLIB]**

Used for sequentializing a partitioned dataset. Either all or a specified subset of the members of a partitioned dataset are used to create a sequential dataset; each member is preceded by an ADD control statement. The resulting file may then be input to PDSLOAD to construct a new partitioned dataset containing the offloaded members. This is a batch implementation of the Offload function provided in Greg Price's REVIEW.

The most recent version of OFFLOAD is located in File #093 of the CBT tape. The program was originally written by David B. Cole. OFFLOAD requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines a sequential input dataset that contains control statements.
- anyname1 - defines the input partitioned dataset.
- anyname2 - defines the output sequential dataset.

I would also suggest taking a look at UNUPDTE.

Control Statements

The control statements are modeled after the control statements for the IEBCOPY utility program. There are three control statements; the control statement function, as well as all of the operands, may identified by the minimal possible abbreviation, one character: O for Offload, S for Select and E for Exclude.

The Offload control statement has three sub-operands: I for Input ddname, O for Output ddname, and T for Type.

Both the Select and Exclude control statements each have a single sub-operand: M for Member.

```
O  I=ddname

   ,O=ddname

   ,T={IEBUPDTE | IEBUPDAT}
```

The Offload control statement specifies the input and output DD names and the type of ADD control statement to insert ahead of each members data records. The operands are:

I=ddname

Specifies the DD name from which the partitioned dataset is read.

O=ddname

Specifies the DD name to which the sequential dataset is written.

T={IEBUPDTE | IEBUPDAT}

Specifies that an ADD control statement is to be written preceding each members data records. If this sub-operand is omitted, no ADD statement will be written. The values that may be specified for T are:

- IEBUPDTE - Specifies the ADD control statements are to be in IEBUPDTE format.
- IEBUPDAT - Specifies the ADD control statements are to be in IEBUPDAT format.

```
SM={membername | (membername1[,membername2]...)}
```

The Select control statement specifies that only these named members are to be processed - Selected - from the partitioned dataset. As many Select control statements may be used as necessary; subsequent control statements are processed as a continuation of the initial control statement.

M={membername | (membername1[,membername2]...)}

Specifies one or a list of names of members to be processed. If only a single member name is used, it need not be enclosed in parentheses.

```
EM={membername | (membername1[,membername2]...)}
```

The Exclude control statement specifies that these named members are not to be processed - Excluded - and all other members are processed from the partitioned dataset. As many Exclude control statements may be used as necessary; subsequent control statements are processed as a continuation of the initial control statement.

M={membername | (membername1[,membername2]...)}

Specifies one or a list of names of members to be excluded. If only a single member name is used, it need not be enclosed in parentheses.

If no Select or Exclude control statements are provided, all members of the partitioned dataset are processed.

The Select and Exclude control statements are mutually exclusive; only one may be used in a single execution of OFFLOAD.

Examples

Offload selected members of a partitioned dataset

```
//OFFLOAD  JOB (01),'OFFLOAD PO',CLASS=S,MSGCLASS=X
//OFFLOAD EXEC PGM=OFFLOAD
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DISP=SHR,DSN=SYSC.VSAMIOP.SOURCE
//SYSUT2   DD  DISP=(,CATLG,DELETE),DSN=JAY01.VSAMIOP.OFFLOAD,
//             UNIT=SYSDA,SPACE=(CYL,(5,2),RLSE),VOL=SER=PUB001,
//             DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSIN    DD  *
  O I=SYSUT1,O=SYSUT2,T=IEBUPDTE
```

```
  S M=(ESDSADDT,ESDSLOAD,ESDSREAD,ESDSUPDT,KSDSLOAD,KSDSRAND)
  S M=(KSDSREAD,KSDSSSEQ,KSDSUPDT,RRDSLODR,RRDSLODS,RRDSRAND)
  S M=(RRDSREAD,RRDSSSEQ,RRDSUPDT)
/*
//
```

Offload all but excluded member of a partitioned dataset

```
//OFFLOAD  JOB (01),'OFFLOAD PO',CLASS=S,MSGCLASS=X
//OFFLOAD EXEC PGM=OFFLOAD
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DISP=SHR,DSN=SYSC.VSAMIOP.SOURCE
//SYSUT2   DD  DISP=(,CATLG,DELETE),DSN=JAY01.VSAMIOP.OFFLOAD,
//             UNIT=SYSDA,SPACE=(CYL,(5,2),RLSE),VOL=SER=PUB001,
//             DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSIN    DD  *
  O I=SYSUT1,O=SYSUT2,T=IEBUPDTE
  E M=$INDEX
/*
//
```

Offload all members of a partitioned dataset

```
//OFFLOAD  JOB (01),'OFFLOAD PO',CLASS=S,MSGCLASS=X
//OFFLOAD EXEC PGM=OFFLOAD
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DISP=SHR,DSN=JAY01.ISAM.COBOL
//SYSUT2   DD  DISP=(,CATLG,DELETE),DSN=JAY01.ISAMCOB.OFFLOAD,
//             UNIT=SYSDA,SPACE=(CYL,(5,2),RLSE),VOL=SER=PUB001,
//             DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSIN    DD  *
  O I=SYSUT1,O=SYSUT2,T=IEBUPDTE
/*
//
```

PDSLOAD **[written by Bill Godfrey, located in SYSC.LINKLIB]**

Used for recreating a partitioned dataset from a sequential dataset that contains control statements interspersed with data records.

The most recent version of PDSLOAD is located in File #093 of the CBT tape. The program was originally written by Bill Godfrey. PDSLOAD requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines a sequential input dataset that contains control statements and data when PARM=NEW.
- SYSUT1 - defines the input sequential dataset.
- SYSUT2 - defines the output partitioned dataset.

I would also suggest taking a look at UPDTE.

Program Parameter (PARM= on EXEC statement)

The operation of PDSLOAD is controlled by options which are entered as keywords in the parameter field. The keywords and their function are:

NEW

If specified, SYSUT1 is not opened and SYSIN is assumed to specify the input file. NEW must be the first or only option in the PARM field.

SPF

Specifies that SPF statistics are to be generated for the members of the partitioned dataset.

S(namemask)

Specifies that only the members matching namemask will be written to the partitioned dataset. In namemask, the characters '*', '%', and '?' are treated the same and match any character.

UPDTE(xx)

Specifies that all occurrences of the two characters in parentheses, if found in columns 1-2 of the data records within the members, will be changed to './' before being written to the partitioned dataset. This is used in conjunction with OFFLOAD, which changes all occurrences of './' in columns 1-2 of the data records within the members to another constant so that IEBUPDTE and this program will not treat the data records as control statements. IEBUPDTE does not change the data back, but PDSLOAD does.

CTL(xx)

Specifies that all occurrences of the two characters in parentheses will be used to search for the ADD and ALIAS control statements.

NAME={CHK | ASIS | IBM}

Specifies validation of member names. The values that may be coded are:

- CHK - Allows all printable characters, except comma, using CODEPAGE 037.
- ASIS - Bypass all checks. The default is ASIS.
- IBM - Enforce strict IBM JCL standards. 'JCL' is a synonym for IBM.

CK3

When UPDTE(xx) is specified, verifies that column three is blank before changing the specified characters to './'.

If SPF is specified, it must be the first parameter, unless NEW is specified, then it must be the second parameter..

If UPDTE(><) is specified, it must be the last parameter.

Control Statements

There are no control statements. If specific members of a partitioned dataset are to be selected or excluded, their names are included in SYSIN. The names specified are separated by commas. As many statements as

are required may be included in a single execution.

Examples

Load partitioned dataset from OFFLOADed dataset

```
//PDSLOAD  JOB (SYS),'PDSLOAD-1',CLASS=A,MSGCLASS=X
//PDSLOAD EXEC PGM=PDSLOAD,PARM='NEW,SPF'
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DISP=SHR,DSN=HMVS01.VSAMIO.OFFLOAD,
//SYSUT2   DD  DISP=(,CATLG,DELETE),DSN=HMVS02.VSAMIO.SOURCE
//             DCB=(DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=23440),
//             UNIT=SYSDA,VOL=SER=MVS381,
//             SPACE=(CYL,(2,1,15),RLSE)
//
```

Load partitioned dataset using mask to select members

```
//PDSLOAD  JOB (SYS),'PDSLOAD-2',CLASS=A,MSGCLASS=X
//PDSLOAD EXEC PGM=PDSLOAD,PARM='NEW,SPF,S(%%DS%%%%)'
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DISP=SHR,DSN=HMVS01.VSAMIO.OFFLOAD
//SYSUT2   DD  DISP=(,CATLG,DELETE),DSN=HMVS02.VSAMIO.COBOL,
//             DCB=(DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=23440),
//             UNIT=SYSDA,VOL=SER=MVS381,
//             SPACE=(CYL,(2,1,15))
//
```

Add members to partitioned dataset from OFFLOADed dataset

```
//PDSLOAD  JOB (SYS),'PDSLOAD-3',CLASS=A,MSGCLASS=X
//PDSLOAD EXEC PGM=PDSLOAD,PARM='SPF,S(VSAMIO%%)'
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DISP=SHR,DSN=HMVS01.VSAMIO.OFFLOAD
//SYSUT2   DD  DISP=MOD,DSN=HMVS02.VSAMIO.COBOL
//
```

PDSMATCH **[written by Bill Godfrey, updated by Carl Hafner, located in SYSC.LINKLIB]**

Used for comparing two partitioned datasets. PDSMATCH compares the directories of the two datasets and prints a report showing which members match, which members do not match, or members that are not present in both directories.

The most recent version of PDSMATCH is located in File #357 of the CBT tape. This version of the program was modified by Carl Hafner; Steli, Inc. PDSMATCH requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSLIB1 - defines the first partitioned dataset.
- SYSLIB2 - defines the second partitioned dataset.

Program Parameter (PARM= on EXEC statement)

The operation of PDSMATCH is controlled by options which are entered as keywords in the parameter field. The keywords and their function are:

EQO

Specifies that only members that match are to be listed.

NEO

Specifies that only members that do not match are to be listed.

USER

Specifies that members match only if they have the same name and all the statistics match. Statistics of up to 62 bytes of data in the directory entry are compared. This includes data placed in the directory entry linkage editor, SPF edit, etc. Note: If two members have the same name and neither has any userdata, they are treated as unequal.

DATA

Specifies that members match only if all their data records are the same. Note: If two members have the same name and both are empty, they are treated as unequal.

If no PARM is supplied, only the directories are compared to report members present in both, or only one, of the datasets.

Control Statements

There are no control statements. If specific members of a partitioned dataset are to be selected or excluded, their names are included in SYSIN. The names specified are separated by commas. As many statements as are required may be included in a single execution.

Examples

Compare two partitioned datasets, names only

```
//PDSMATCH JOB (1),'PDSMATCH-1',CLASS=A,MSGCLASS=X
//PDSMATCH EXEC PGM=PDSMATCH,REGION=4M,
//            PARM=''
//SYSPRINT  DD SYSOUT=*
//SYSLIB1   DD DISP=SHR,DSN=HMVS02.VSAMIO.COBOL
//SYSLIB2   DD DISP=SHR,DSN=HMVS01.VSAMIO.SOURCE
//
```

Compare two partitioned datasets, names and userdata

```
//PDSMATCH JOB (1),'PDSMATCH-2',CLASS=A,MSGCLASS=X
//PDSMATCH EXEC PGM=PDSMATCH,REGION=4M,
//            PARM='NEO,USER'
```

```
//SYSPRINT  DD SYSOUT=*
//SYSLIB1   DD DISP=SHR,DSN=SYS2.LINKLIB
//SYSLIB2   DD DISP=SHR,DSN=SYSC.LINKLIB
//
```

Compare two partitioned datasets, names and actual data records

```
//PDSMATCH JOB (1),'PDSMATCH-1',CLASS=A,MSGCLASS=X
//PDSMATCH EXEC PGM=PDSMATCH,REGION=4M,
//            PARM='NEO,DATA'
//SYSPRINT  DD SYSOUT=*
//SYSLIB1   DD DISP=SHR,DSN=JAY01.OLD.PTFS
//SYSLIB2   DD DISP=SHR,DSN=JAY01.NEW.PTFS
//
```

PDSPRINT **[written by Jim Marshall, located in SYSC.LINKLIB]**

Used for printing or punching members from a partitioned dataset.

PDSPRINT is located in File #316 of the CBT tape. PDSPRINT requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines the optional sequential input dataset that contains PDSPRINT control statements.
- SYSUT1 - defines the input partitioned dataset.
- SYSUT2 - defines the sequential output dataset.

Program Parameter (PARM= on EXEC statement)

A program parameter may be supplied to PRINT or PUNCH an entire dataset, in which case it is not necessary to supply control statements and the SYSIN DD may be omitted or DUMMY. If SYSIN is provided, the PARM is ignored. The keywords and their function are:

PRINT

Specifies that all members of the dataset are to be printed.

PUNCH

Specifies that all members of the dataset are to be punched. The data records for each member are preceded by an IEBUPDTE ADD control statement.

Control Statements

```
PRINT  MEMBER=(name[,name]...) | SCAN=(namemask)
```

The PRINT control statement specifies that members are to be selected and printed. The mutually exclusive parameters are:

MEMBER=(name[,name]...)

Specifies that the named members are to be selected. The list of names is enclosed in parentheses and separated by commas. The list may be continued to as many cards as required by stopping with a comma before column 72 and continuing the list on a subsequent card in any column before 72.

SCAN(namemask)

Specifies that the directory names will be compared with the mask supplied and any matching member will be selected.

```
PUNCH  MEMBER=(name[,name]...) | SCAN=(namemask)
```

The PUNCH control statement specifies that members are to be selected and punched. The data records for each member are preceded by an IEBUPDTE ADD control statement. The mutually exclusive parameters are:

MEMBER=(name[,name]...)

Specifies that the named members are to be selected. The list of names is enclosed in parentheses and separated by commas. The list may be continued to as many cards as required by stopping with a comma before column 72 and continuing the list on a subsequent card in any column before 72.

SCAN(namemask)

Specifies that the directory names will be compared with the mask supplied and any matching member will be selected.

Examples

Punch all members with IEBUPDTE ADD cards

```
//PDSPRINT JOB (001),'PDSPRINT-1',CLASS=A,MSGCLASS=X
//PDSPRINT EXEC PGM=PDSPRINT,REGION=256K,PARM='PUNCH'
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DSN=SYSC.PROCLIB,DISP=SHR
//SYSUT2   DD  SYSOUT=B
//
```

Print selected members

```
//PDSPRINT JOB (001),'PDSPRINT-2',CLASS=A,MSGCLASS=X
//PDSPRINT EXEC PGM=PDSPRINT,REGION=256K,PARM=''
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DSN=SYSC.PROCLIB,DISP=SHR
//SYSUT2   DD  SYSOUT=*
//SYSIN    DD  *
  PRINT MEMBER=(ASMXC,ASMXCL)
//
```

Punch long list of selected members

```
//PDSPRINT JOB (001),'PDSPRINT-3',CLASS=A,MSGCLASS=X
//PDSPRINT EXEC PGM=PDSPRINT,REGION=256K,PARM=''
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DISP=SHR,DSN=SYSC.VSAMIO.SOURCE
```

```
//SYSUT2   DD  SYSOUT=*
//SYSIN    DD  *
  PUNCH MEMBER=(ESDSADDT,ESDSLOAD,ESDSUPDT,ESDSREAD,
                KSDSMULT,KSDSLVAR,KSDSRAND,KSDSREAD,KSDSSSEQ,KSDSUPDT)
//
```

Print members selected with name pattern

```
//PDSPRINT JOB (001),'PDSPRINT-4',CLASS=A,MSGCLASS=X
//PDSPRINT EXEC PGM=PDSPRINT,REGION=256K,PARM=''
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DISP=SHR,DSN=SYSC.PROCLIB
//SYSUT2   DD  SYSOUT=*
//SYSIN    DD  *
  PRINT SCAN=(PL1)
//
```

PDSPROGM **[written by Bill Godfrey, located in SYSC.LINKLIB]**

Used for maintenance on partitioned datasets. PDSPROGM can be used to:

- Delete a member.
- Rename a member.
- Create an alias for a member.

PDSPROGM is located in File #316 of the CBT tape. PDSPROGM requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines the sequential input dataset that contains PDSPROGM control statements.
- SYSLIB - defines the partitioned dataset.
- anyname - may be used to define the partitioned dataset when the FILE control statement is used.

Program Parameter (PARM= on EXEC statement)

A program parameter may be supplied to modify overall operation of PDSPROGM. The keywords and their function are:

NL

Specifies that control statements will not be logged to SYSPRINT.

NM

Specifies that normal messages will not be logged to SYSPRINT, only errors.

NF

Specifies that encountering an error will not cause flushing of SYSIN.

NR

Specifies that there will be no enforcement of characters used in a new member

name.

NC

Specifies that no capitalization will be done on new member names.

Control Statements

```
DELETE   membername
```

The DELETE control statement specifies that the named member is to be deleted. DELETE may be abbreviated DEL or D, and has the synonyms SCRATCH and SCR.

```
RENAME   membername

         newname
```

The RENAME control statement specifies that the named member is to be renamed. RENAME may be abbreviated REN or R.

```
ALIAS   membername

        newname
```

The ALIAS control statement specifies that an alias is to be created for the named member. ALIAS may be abbreviated ALI or A.

```
OPTION   [{LOG | NOLOG}]

         [{MSG | NOMSG}]

         [FLUSH | NOFLUSH}]
```

The OPTION control statement specifies options that modify PDSPROGM operations. The parameters are:

LOG | NOLOG

- LOG - Specifies that control statements will be logged to SYSPRINT. The default is LOG.
- NOLOG - Specifies that control statements will not be logged to SYSPRINT.

MSG | NOMSG

- MSG - Specifies that normal messages will not be logged to SYSPRINT, only errors. The default is MSG.
- NOMSG - Specifies that normal messages will not be logged to SYSPRINT, only errors.

FLUSH | NOFLUSH

- FLUSH - Specifies that encountering an error will cause flushing of SYSIN. The default is FLUSH.
- NOFLUSH - Specifies that encountering an error will not cause flushing of SYSIN.

`FILE  ddname`

The FILE control statement specifies that any subsequent operations are to be performed against the dataset referenced by ddname. This allows multiple partitioned datasets to be processed in a single execution. FILE may be abbreviated FIL or F.

Examples

Partitioned dataset maintenance

```
//PDSPROGM JOB (001),'PDSPROGM-1',CLASS=S,MSGCLASS=X
//PDSPROGM EXEC PGM=PDSPROGM
//SYSPRINT DD  SYSOUT=*
//SYSLIB   DD  DISP=SHR,DSN=SYSP.VSAMIO.SOURCE
//SYSLIB2  DD  DISP=SHR,DSN=SYSP.VSAMIOP.SOURCE
//SYSIN    DD  *
  OPTION NOFLUSH
  ALIAS KSDSLVAR KSDSVARL
  DELETE $INDEX
  RENAME ESDSUPDT ESDSUPDA
  FILE SYSLIB2
  RENAME ESDSUPDT ESDSUPDA          <- this statement will be applied to the dataset specifie
//
```

PDSSCAN **[written by Rick Fochtman, located in SYSC.LINKLIB]**

Used to scan all members of one or more partitioned datasets, searching for specified strings. Note: Does not update partitioned datasets, only reports datasets/members where strings are found. PDSSCAN is located in File #684 of the CBT tape. PDSSCAN requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines the sequential input dataset that contains strings to scan for.
- @anyname - defines the partitioned dataset(s) to be scanned.

The number of DD statements may be from one to the MVS limit for a step. Any name is allowed, as long as the name begins with '@'.

I would also suggest taking a look at PDSUPDTE.

Program Parameter (PARM= on EXEC statement)

If any parameter is supplied, PDSSCAN will print a brief description of the program use instructions.

Control Statements

The control statements are simply the list of strings to be searched for in each partitioned dataset. Each statement read from SYSIN may contain a single string. A string begins at the first non-blank character and is terminated by the first blank; i.e., no embedded blanks are allowed.

Examples

Scan two partitioned datasets searching for strings

```
//PDSSCAN  JOB 1,'SCAN PDS',CLASS=A,MSGCLASS=X
//PDSSCAN  EXEC PGM=PDSSCAN,REGION=1024K
//SYSPRINT DD  SYSOUT=*
//@DD1     DD  DISP=SHR,DSN=SYSP.SETUP.LOCAL.CNTL
//@DD2     DD  DISP=SHR,DSN=SYSC.CBT.SOURCE
//SYSIN    DD  *
  SYS2.LINKLIB
  SYSC.LINKLIB
//
```

PDSUPDTE [located in SYSC.LINKLIB]

Used to scan all members of one or more partitioned datasets, searching for specified strings. If a string is found and operating mode is UPDATE, each string that is found is changed to the supplied second string. PDSUPDTE was created by disassembling IPOUPDTE, an IBM installation aid. PDSUPDTE is located in File #065 of the CBT Overflow tape. PDSUPDTE requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines the sequential input dataset that contains strings to scan for.
- @anyname - defines the partitioned dataset(s) to be scanned.

The number of DD statements may be from one to the MVS limit for a step. Any name is allowed, as long as the name begins with '@'.

I would also suggest taking a look at PDSSCAN. There is a detailed user's guide for PDSUPDTE available for download from this site at: ../downloads/pdf/pdsupdte.pdf.

Program Parameter (PARM= on EXEC statement)

A keyword supplied in the PARM determines the operating mode of PDSUPDTE:

CHECK

Specifies that the report will be produced showing all successful searches for strings and what the updated data record would be, but no updates are written to the partitioned dataset(s).

UPDATE

Specifies that all changes will be written to the partitioned dataset(s).

Control Statements

The control statements supply the string to search for and the string to replace. The format is:

```
search<replace<[limiter<]
```

The delimiter for each string value is the less than symbol '<'.

The optional third string, shown as limiter above, allows the specification of a second search value, which must be present in the unmodified data record where the search string is found before the replacement will be made.

Comments are allowed following the search/replace strings; comments must be separated from strings by at least one blank.

Examples

Scan two partitioned datasets searching for strings, check only

```
//PDSUPDTE JOB 1,'PDSUPDTE-1',CLASS=A,MSGCLASS=X
//PDSUPDTE EXEC PGM=PDSUPDTE,REGION=1024K,PARM=CHECK
//SYSPRINT DD  SYSOUT=*
//@DD1     DD  DISP=SHR,DSN=SYSP.SETUP.LOCAL.CNTL
//@DD2     DD  DISP=SHR,DSN=SYSC.CBT.SOURCE
//SYSIN    DD  *
SYS2.LINKLIB<SYSC.LINKLIB<   Change target library
//
```

Scan a partitioned dataset, two matchings strings required, update if found

```
//PDSUPDTE JOB 1,'PDSUPDTE-2',CLASS=A,MSGCLASS=X
//PDSUPDTE EXEC PGM=PDSUPDTE,REGION=1024K,PARM=UPDATE
//SYSPRINT DD  SYSOUT=*
//@DD1 DD DISP=SHR,DSN=HMVS01.VSAMIO.SOURCE
//SYSIN    DD  *
PUB001<MVS380<GO<            Change volser on GO steps
//
```

PDSUR **[written by Gene Czarcinski, located in SYSC.LINKLIB]**

Used to copy partitioned datasets from DASD to tape (unload) and from tape to DASD (reload); the tape datasets are in IEHMOVE format and PDSUR will restore tapes created with IEHMOVE. PDSUPDTE was written by Gene Czarcinski and is located in File #949 of the CBT tape. PDSUPDTE requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines the sequential input dataset that contains PDSUR Control statements.
- anyname - defines the partitioned dataset.
- anyname - defines the tape dataset.

Control Statements

```
UNLOAD FROMDD=ddname

       ,TODD=ddnamel

       [,SELECT]

       [,EXCLUDE]
```

The UNLOAD control statement specifies that an unload (backup creation) operation is to be performed. UNLOAD may be abbreviated U. The parameters are:

FROMDD=ddname

Specifies the ddname of the partitioned dataset. FROMDD may be abbreviated F.

TODD=ddname

Specifies the ddname of the tape dataset. TODD may be abbreviated T.

SELECT

Specifies that only the members named on the following MEMBER control statements are to be processed. SELECT may be abbreviated S.

EXCLUDE

Specifies that all members in the partitioned dataset, except for the members named on the following MEMBER control statements, are to be processed. EXCLUDE may be abbreviated E.

```
RELOAD FROMDD=ddname

       ,TODD=ddname

       [,SELECT]
       [,EXCLUDE]

       [,REPLACE]

       [,LEAVE]
```

The RELOAD control statement specifies that a reload (restore from backup) operation is to be performed. RELOAD may be abbreviated R. The parameters are:

FROMDD

Specifies the ddname of the tape dataset. FROMDD may be abbreviated F.

TODD

Specifies the ddname of the partitioned dataset. TODD may be abbreviated T.

SELECT

Specifies that only the members named on the following MEMBER control statements are to be restored. SELECT may be abbreviated S.

EXCLUDE

Specifies that all members in the partitioned dataset, except for the members named on the following MEMBER control statements, are to be restored. EXCLUDE may be abbreviated E.

REPLACE

Specifies that if a member being restored already exists in the partitioned dataset, it is to be replaced with the member from the backup.

LEAVE

Specifies that at the end of the specified operation the tape file is closed and left mounted, even if DISP=(OLD,KEEP) is specified.

```
LIST  FROMDD=ddname

      [,LEAVE]
```

The LIST control statement specifies that the contents of an unloaded dataset are to be read and listed. The parameters are:

FROMDD

Specifies the ddname of the tape dataset. FROMDD may be abbreviated F.

LEAVE

Specifies that at the end of the specified operation the tape file is closed and left mounted, even if DISP=(OLD,KEEP) is specified.

`MEMBER` membername[,membername...]

The MEMBER control statement specifies member names for the SELECT or EXCLUDE option specified on the previous UNLOAD or RELOAD control statement. More than one member control statement may follow an UNLOAD or RELOAD statement, so that many names can be specified. The operand field must contain one or more member names separated by commas.

Examples

Unload two partitioned datasets

```
//PDSUR    JOB (1),'PDSUR-1',CLASS=A,MSGCLASS=X
//PDSUR    PROC OUT=A,BLK=3509,REG=4M
//PDSUR    EXEC PGM=PDSUR,REGION=&REG
//SYSPRINT  DD SYSOUT=&OUT,DCB=BLKSIZE=&BLK
//SYSIN     DD DCB=(BLKSIZE=800,BUFNO=1)
//         PEND
//         EXEC PDSUR
//SYSIN     DD *
UNLOAD FROMDD=DISK,TODD=TAPE
U F=D2,T=T2
//DISK      DD DISP=SHR,DSN=SYS2.PROCLIB
//D2        DD DISP=SHR,DSN=SYS2.LINKLIB
//TAPE      DD DISP=(,KEEP),DSN=A,LABEL=1,UNIT=TAPE,
//            VOL=SER=22007A
//T2        DD DISP=(,KEEP),DSN=B,LABEL=2,VOL=REF=*.TAPE
//
```

Reload a partitioned dataset

```
//PDSUR    JOB (1),'PDSUR-2',CLASS=A,MSGCLASS=X
//PDSUR    PROC OUT=A,BLK=3509,REG=50K
//PDSUR    EXEC PGM=PDSUR,REGION=&REG
//SYSPRINT  DD SYSOUT=&OUT,DCB=BLKSIZE=&BLK
//SYSIN     DD DCB=(BLKSIZE=800,BUFNO=1)
//         PEND
//         EXEC PDSUR
//SYSIN     DD *
RELOAD FROMDD=DD1,T=DD2,R
//DD1       DD UNIT=TAPE,DISP=OLD,DSN=HMVS01.A,VOL=SER=22007B
//DD2       DD DISP=(NEW,KEEP),DSN=HMVS01.CNTL.R,
//            UNIT=SYSDA,VOL=SER=MVS380,
//            DCB=(HMVS01.CNTL),
//            SPACE=(CYL,(1,1,5))
//
```

Unload and reload a partitioned dataset in one step

```
//PDSUR    JOB (1),'PDSUR-3',CLASS=A,MSGCLASS=X
//PDSUR    PROC OUT=A,BLK=3509,REG=50K
//PDSUR    EXEC PGM=PDSUR,REGION=&REG
//STEPLIB   DD DISP=SHR,DSN= ... (POINT AT LIBRARY FOR PDSUR PGM)
//SYSPRINT  DD SYSOUT=&OUT,DCB=BLKSIZE=&BLK
//SYSIN     DD DCB=(BLKSIZE=800,BUFNO=1)
//         PEND
//         EXEC PDSUR
//SYSIN     DD *
UNLOAD FROMDD=DD1,TODD=DD2
RELOAD FROMDD=DD3,TODD=DD1
//DD1       DD DISP=OLD,DSN=M2.USRID.LINKLIB
//DD2       DD DISP=(,KEEP),DSN=M2.USRID.BAKUP,LABEL=3,
//            UNIT=TAPE,VOL=SER=22007C
//DD3       DD DISP=OLD,DSN=M2.USRID.OLDTAP,LABEL=10,
//            UNIT=TAPE,VOL=SER=21250C
//
```

List three unloaded partitioned datasets

```
//PDSUR    JOB (1),'PDSUR-4',CLASS=A,MSGCLASS=X
//PDSUR    PROC OUT=A,BLK=3509,REG=50K
//PDSUR    EXEC PGM=PDSUR,REGION=&REG
```

```
//SYSPRINT  DD SYSOUT=&OUT,DCB=BLKSIZE=&BLK
//SYSIN     DD DCB=(BLKSIZE=800,BUFNO=1)
//         PEND
//         EXEC PDSUR
//SYSIN     DD *
LIST FROMDD=DD1,LEAVE
LIST F=DD2,LEAVE
L FROMDD=DD3,L
//DD1       DD DISP=OLD,DSN=A,LABEL=1,UNIT=TAPE,
//             VOL=SER=22007D
//DD2       DD DISP=OLD,DSN=E,LABEL=5,VOL=REF=*.DD1
//DD3       DD DISP=OLD,DSN=I,LABEL=9,VOL=REF=*.DD1
//
```

Unload with selects and excludes

```
//PDSUR    JOB (1),'PDSUR-5',CLASS=A,MSGCLASS=X
//PDSUR    PROC OUT=A,BLK=3509,REG=50K
//PDSUR    EXEC PGM=PDSUR,REGION=&REG
//SYSPRINT  DD SYSOUT=&OUT,DCB=BLKSIZE=&BLK
//SYSIN     DD DCB=(BLKSIZE=800,BUFNO=1)
//         PEND
//         EXEC PDSUR
//SYSIN     DD *
UNLOAD FROMDD=DD1,TODD=DD2,S
MEMBER ASMGASM,BASIC360,PASCAL
UNLOAD FROMDD=DD1,TODD=DD3,E
MEMBER ASMGASM,BASIC360,PASCAL
//DD1       DD DISP=SHR,DSN=SYSC.LINKLIB
//DD2       DD DISP=(,KEEP),DSN=USRID.X1,LABEL=1,UNIT=TAPE,
//             VOL=(PRIVATE,RETAIN,SER=22007E)
//DD3       DD DISP=(,KEEP),DSN=USRID.X2,LABEL=2,VOL=REF=*.DD2
//
```

Reload with selects and excludes

```
//PDSUR    JOB (1),'PDSUR-6',CLASS=A,MSGCLASS=X
//PDSUR    PROC OUT=A,BLK=3509,REG=50K
//PDSUR    EXEC PGM=PDSUR,REGION=&REG
//SYSPRINT  DD SYSOUT=&OUT,DCB=BLKSIZE=&BLK
//SYSIN     DD DCB=(BLKSIZE=800,BUFNO=1)
//         PEND
//         EXEC PDSUR
//SYSIN     DD *
RELOAD FROMDD=DD1,TODD=DD2,LEAVE,S
MEMBER PROG1,PROG2
RELOAD FROMDD=DD1,TODD=DD3,EXCLUDE
MEMBER PROG3,PROG4
MEMBER PROG5,PROGA
//DD1       DD DISP=OLD,DSN=M2.USRID.T,LABEL=5,UNIT=2400-4,
//             VOL=SER=BKPTAP
//DD2       DD DISP=OLD,DSN=M2.USRID.LIB1
//DD3       DD DISP=OLD,DSN=M2.USRID.LIB2
//
```

RECV370 **[written by Jim Morrison, located in SYSC.LINKLIB]**

Used to process XMIT files produced by either the TSO/E TRANSMIT command processor, the XMIT370 batch program, or a similar facility. RECV370 was written by Jim Morrison and is located in File #571 of the CBT tape. RECV370 requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset (IEBCOPY messages).
- RECVLOG - defines a sequential output message dataset (RECV370 messages).
- SYSIN - unused, but required by IEBCOPY.
- SYSUT1 - work dataset.
- SYSUT2 - defines the output dataset.
- XMITIN - defines sequential input dataset, BLKSIZE must be 80 (XMIT file).

Control Statements

RECV370 uses no control statements.

Examples

Unpack XMIT to create PDS

```
//RECV370  JOB (001),'RECV370',CLASS=A,MSGCLASS=X
//RECV370  PROC
//RECV370  EXEC PGM=RECV370,REGION=1024K
//RECVLOG   DD SYSOUT=*                    RECV370 OUTPUT MESSAGES
//SYSPRINT  DD SYSOUT=*                    IEBCOPY OUTPUT MESSAGES
//SYSIN     DD DUMMY                       IEBCOPY REQUIRES
//SYSUT1    DD UNIT=SYSDA,                 WORK DATASET
//             SPACE=(CYL,(10,10)),
//             DCB=BLKSIZE=5600
//XMITIN    DD DDNAME=XMITIN               INPUT DATASET
//SYSUT2    DD DDNAME=XMITOUT              OUTPUT DATASET
//         PEND
//RECV1    EXEC RECV370
//XMITIN   DD  UNIT=01C,DCB=BLKSIZE=80          <- INPUT
//SYSUT2   DD  DSN=HMVS01.FILE571.PDS,          <- OUTPUT
//             VOL=SER=PUB001,UNIT=SYSDA,
//             SPACE=(TRK,(200,,40),RLSE),
//             DISP=(,CATLG)
//
```

RESETDS **[author unknown, located in SYSC.LINKLIB]**

Used to reset to empty status partitioned or sequential datasets in preparation for a reload. The dataset will appear as though it had been scratched and reallocated, however, the overhead of scratch/allocate and uncatalog/catalog is avoided; additionally, the dataset will reside in the original location on the volume. The author of RESETDS is unknown; the source was NASPA 1986. RESETDS requires the following DD statements:

- RESET??? - defines a sequential or partitioned dataset.

  Any number of DD statements may be provided, as long as the name begins with 'RESET'. RESETDS will process all DD names in a single execution.

Control Statements

RESETDS uses no control statements.

Examples

Reset datasets to empty

```
//RESETDS  JOB (1),'RESET PDS/PS EMPTY',CLASS=A,MSGCLASS=X
//RESET    EXEC PGM=RESETDS
//STEPLIB   DD  DSN=SYSC.LINKLIB,DISP=SHR
//RESET1    DD  DISP=OLD,DSN=PDS.DATASET1
//RESETSEQ  DD  DISP=OLD,DSN=SEQ.DATASET
//RESET02   DD  DISP=OLD,DSN=PDS.DATASET2
//RESET003  DD  DISP=OLD,DSN=PDS.DATASET3
//
```

REVLMOD **[written by Greg Price, located in SYSC.LINKLIB]**

Used to reload load modules that have been offloaded with REVIEW's OFFLOAD command. REVLMOD was written by Greg Price and is located in File #134 of the CBT tape. REVLMOD requires the following DD statements:

- SYSIN - defines a sequential dataset containing the offloaded load module.
- SYSUT2 - defines a load library.

Control Statements

REVLMOD uses no control statements.

Examples

Reload load module from sequential dataset

```
////REVLMOD  JOB (001),REVLMOD,CLASS=S,MSGCLASS=X
//REVLMOD  EXEC PGM=REVLMOD,REGION=1M
//SYSUT2    DD DISP=MOD,DSN=SYS2.LINKLIB
//SYSIN     DD DISP=SHR,DSN=HMVS01.ISAMLOAD.OFFLOAD
//
```

SORT **[supplied by IBM, located in SYSC.LINKLIB]**

Used for sorting data records from one or more sequential datasets.

This IBM program originated with OS Release 21.8, so the datasets it utilizes must reside on tape or 2314 DASD. The limitation to 2314 DASD is the simple fact of when this program originated, even 3330 DASD were scarce, and 3350/3375/3380/3390 did not exist. But now that I have stated the limitation, I will say that I have found that there are times when SORT will read SORTIN from, and write SORTOUT to, a dataset on a DASD type later than 2314, as can be seen in some of my example jobstreams below. There were efforts discussed on the Hercules' forums, years ago, to rewrite the SORT program, using the source code available, and upgrade the program to use all of the types of DASD we have available now for MVS 3.8 running under Hercules. There was even a brute force modification made to the program that allowed it to use the later types, but nothing was done to update the algorithms that balance and optimize the sort, so I don't think that is a solution that appeals to many. As it is, the program will work just fine, as long as you

keep some 2314 DASD defined for it to use as sort work datasets, which hold the various sequence sets that are merged in the final phase to produce the output dataset.

SORT requires the following DD statements:

- SYSOUT - defines a sequential output message dataset (output from MSG=AP or MSG=CP).
- SORTLIB - defines the partitioned input dataset containing the Sort/Merge external modules.
- SORTIN - defines the data records to be sorted.

  When using the SORT control statement, SORTIN is the DD name. When using the MERGE control statement, SORTIN01 through SORTIN16 are the DD names to use.

- SORTWKnn - defines external work datasets; used only when using the SORT control statement. The first work dataset is SORTWK01; additional datasets may be provided, up to SORTWK32.
- SORTOUT - defines the sequential output dataset.
- SORTCKPT - defines a dataset for checkpoint records. If you are not using the checkpoint facility this statement is not required.
- SYSIN - defines a sequential input dataset that contains SORT/MERGE control statements.

Control Statements

```
SORTFIELDS={(p1,m1,f1,s1...p64,m64,f64,s64) | (p1,m1,s1...p64,m64,s64),FORMAT=xx}

    [,SIZE=y]

    [,SKIPREC=z]

    [,CKPT]
```

The SORT control statement specifies a SORT operation. The parameters are:

FIELDS=(p1,m1,f1,s1...p64,m64,f64,s64) **or**
FIELDS=(p1,m1,s1...p64,m64,s64),FORMAT=xx

p1 (...p64) - specifies the beginning (high-order location) of a control field relative to the beginning of the record which contains the control field. (For variable-length records, the logical record includes the four-byte record length indicator, so record offsets must include those four bytes in the offset position used.) The first (high-order) byte in a record is byte 1, the second is byte 2, etc. All control fields, except binary, must begin on a byte boundary.

m1 (...m64) - specifies the length of the control field. All control fields except binary must be a whole number of bytes long.

f1 (...f64) - specifies the format of the data in the control field. The valid two-character abbreviations are:

- CH - character
- ZD - zoned decimal
- PD - packed decimal
- FI - fixed-point
- BI - binary
- FL - floating point

If all the control fields contain the same type of data, you can omit the f parameters and use the optional FORMAT=xx operand.

SIZE=y

Specifies the number of records in the input dataset. The value y can be either the actual dataset size or an estimate of the size. If you give an actual dataset size, do not include any records inserted in the input dataset by one of your routines. If the number of records in the input dataset, as counted by the sort/merge program, does not agree with the value of the SIZE parameter, the sort terminates. The value specified in the SIZE parameter is placed in the IN field of message IER047A or IER054I. If you give an estimated dataset size, precede the value by E (for example, E5000).

SKIPREC=z

Specifies the number of records to be skipped when reading SORTIN. If you want the sort to skip a certain number of records before starting to process the input dataset, substitute the number of records you want skipped for z.

CKPT

Specifies that the sort/merge program is to activate the checkpoint facility of the operating system. The program takes checkpoints at various points in the sort or merge operation. You can have the program restart from the last checkpoint taken.

```
MERGEFIELDS={(p1,m1,f1,s1...p64,m64,f64,s64) | (p1,m1,s1...p64,m64,s64),FORMAT=xx}

     [,SIZE=y]
```

The MERGE control statement specifies a MERGE operation. The parameters are:

FIELDS=(p1,m1,f1,s1...p64,m64,f64,s64) **or**
FIELDS=(p1,m1,s1...p64,m64,s64),FORMAT=xx

p1 (...p64) - specifies the beginning (high-order location) of a control field relative to the beginning of the record which contains the control field. (For variable-length records, the logical record includes the four-byte record length indicator, so record offsets must include those four bytes in the offset position used.) The first (high-order) byte in a record is byte 1, the second is byte 2, etc. All control fields, except binary, must begin on a byte boundary.

m1 (...m64) - specifies the length of the control field. All control fields except binary must be a whole number of bytes long.

f1 (...f64) - specifies the format of the data in the control field. The valid two-character abbreviations are:

- CH - character
- ZD - zoned decimal
- PD - packed decimal
- FI - fixed-point
- BI - binary

- FL - floating point

If all the control fields contain the same type of data, you can omit the f parameters and use the optional FORMAT=xx operand.

SIZE=y

Specifies the number of records in the input dataset. The value y can be either the actual dataset size or an estimate of the size. If you give an actual dataset size, do not include any records inserted in the input dataset by one of your routines. If the number of records in the input dataset, as counted by the sort/merge program, does not agree with the value of the SIZE parameter, the sort terminates. The value specified in the SIZE parameter is placed in the IN field of message IER047A or IER054I. If you give an estimated dataset size, precede the value by E (for example, E5000).

```
RECORD{TYPE=F,LENGTH=l1,l2,l3 | TYPE=V,LENGTH=l1,l2,l3,[l4,l5]}
```

The RECORD control statement is required only when your routines change record lengths during a sort/merge program run. The control statement statement describes the format and lengths of the records being sorted or merged. The parameters are:

TYPE={F | V}

- F - indicates fixed-length records.
- V - indicates variable-length records.

LENGTH=l1,l2,l3[,l4,l5]

If your input records are fixed-length, use l1, l2, and l3 as follows:

l1 is the length of each record in the input dataset. If you use the RECORD control statement, you must include this value. The value should be the same as the value you specified in the LRECL subparameter of the DCB parameter on the SORTIN DD statement. If the values are not the same, sort/merge uses the value specified on the DD statement.

l2 is the length of each record handled by the sort phase. If you do not specify a value for l2 , the program assumes that it is equal to l1 as it is specified on the record card. If you are going to change record lengths in the sort phase, you must include a value for l2; you do not need l2 for a merging application.

l3 is the length of each record in the output dataset. If you do not specify a value for l3, the program assumes that l3=l2 for a sorting application and that l3=l1 for a merging application. If your routines change record lengths during the final merge phase of the program, you must specify a value for l3. This value should be the same as the value you specified for the LRECL subparameter of the DCB parameter on the SORTOUT DD statement. If the values are different, the sort/merge program uses the value given on the DD statement.

If your input records are variable-length, use l1,ll2, l3, l4, and l5 as follows:

l1 is the maximum length of the records in the input dataset. If you use the

RECORD statement, you must specify a value for l1. The value should be the same as the value you specified in the LRECL subparameter of the DCB parameter on the SORTIN DD statement. If the values are not the same, the program uses the LRECL value.

l2 is the maximum length of the records handled by the sort phase. If you do not specify a value for l2, the program assumes it is equal to l1 as it is specified on the record card. If you change record lengths in the sort phase, you must provide a value for l2. You do not need l2 for a merging application.

l3 is the maximum length of each record in the output dataset. If you do not specify a value for l3, the program assumes l3=l2 for a sort and l3=l1 for a merge. If you include a routine that changes record lengths in the final merge phase, you must specify a value for l3. The value should be the same as the value you provided for the LRECL subparameter of the DCB parameter on the SORTOUT DD statement. If it is not, the program uses the LRECL value.

l4 is the minimum length of records in the input dataset. If you do not specify a value for l4, the program assumes it is equal to the minimum record size necessary to contain the control fields defined on the SORT or MERGE control statement, or the minimum record length allowed by the operating system, whichever is greater. You need not specify this value for a merge.

l5 is the record length that occurs most frequently in the input dataset (modal length). You should use this value to help define a dataset biased toward a particular length. If you do not specify a value for l5, the program assumes it is equal to the average of the maximum and minimum record lengths in the input dataset. If, for example, your dataset contains mostly small records and just a few long records, the program would assume a high modal length and would allocate a larger record storage area than necessary. Conversely, if your dataset contains just a few short records and many long records, the program would assume a low modal length and might not allocate a large enough record storage area to sort your data.

MODS

The MODS control statement is required only if you want the sort/merge program to transfer control to your routine (s) at various points during sort/merge execution. The statement associates your routines with specific exits in the sort/merge program and provides the program with basic descriptions of your routines. For details about exits in the sort/merge program and how to use them, refer to Section 3: Program Modification in GC28-6543-8 OS Sort/Merge Program.

END

The END control statement marks the end of all sort/merge control statements and continuation statements for a particular sort/merge run.

Program Parameter (PARM= on EXEC statement)

The PARM parameter on the EXEC statement is used to supply optional parameters to the SORT program.

The parameters are:

[{BALN | OSCL | POLY | CRCX}]

Specifies the sequence distribution technique to be used. **For the majority of cases, this should be left for the program to determine the optimum setting.** You should be extremely cautious when forcing the sort/merge program to use a specific technique. The program tries to select the most efficient technique for a given application. If it is forced to use another, performance may not be as efficient.

BALN - balanced tape technique. If six or less work datasets are provided, this is the technique used.

OSCL - Oscillating tape technique.

POLY - polyphase tape technique.

CRCX - crisscross direct access technique. If more than six work datasets are provided, this is the technique used.

[,CORE=xxxxxx]

An optional main storage value which will override the storage allocation set up when the sort/merge program was installed.

[,MSG={NO | CC | CP | AC | AP}]

Message output option:

- NO - no messages are printed.
- CC - critical messages only are printed. They appear on the console.
- CP - critical messages only are printed. They appear on the printer.
- AC - all messages are printed. They appear on the console.
- AP - all messages are printed. They appear on the printer.

[,DIAG]

Only for use in problem determination. It should never be used for normal sort or merge jobs, as it degrades sort/merge performance. It causes the program to print diagnostic messages and control cards. If the program terminates in the sort or merge phases with a critical message, use of this parameter also produces an OCl abnormal termination dump for diagnostic use.

Examples

Sort 1 tape dataset, creating a disk dataset

```
//SORT     JOB 1,'SORT-1',CLASS=A,MSGCLASS=X
//SORT     EXEC PGM=SORT,REGION=4M,PARM='MSG=AP'
//SYSOUT    DD SYSOUT=*
//SORTLIB   DD DSN=SYSC.SORTLIB,DISP=SHR
//SORTIN    DD DSN=STATIS.DMTD,UNIT=TAPE,VOL=SER=018352,LABEL=(1,SL),
//             DCB=(RECFM=FB,LRECL=135,BLKSIZE=1350)
//SORTOUT   DD DISP=(,CATLG,DELETE),DSN=HMVS01.STATIS.DATA,
```

```
//             DCB=(RECFM=FB,LRECL=135,BLKSIZE=23355),
//             SPACE=(CYL,(3),RLSE),UNIT=SYSDA,VOL=SER=MVS385
//SYSIN     DD *
 SORT FIELDS=(1,8,ZD,A)
 RECORD TYPE=F,LENGTH=(135)
 END
/*
//SORTWK01  DD UNIT=SORTDA,SPACE=(CYL,(50,50))
//SORTWK02  DD UNIT=SORTDA,SPACE=(CYL,(50,50))
//
```

Sort 2 tape datasets (implicit merge), creating a disk dataset

```
//SORT     JOB 1,'SORT-2',CLASS=A,MSGCLASS=X
//SORT     EXEC PGM=SORT,REGION=4M,PARM='MSG=AP'
//SYSOUT    DD SYSOUT=*
//SORTLIB   DD DSN=SYSC.SORTLIB,DISP=SHR
//SORTIN    DD DSN=STATIS.DMTD.PH1,UNIT=TAPE,VOL=SER=025972, in1
//             LABEL=(1,SL),
//             DCB=(RECFM=FB,LRECL=135,BLKSIZE=1350)
//          DD DSN=STATIS.DMTD.PH2,UNIT=TAPE,VOL=SER=010877, in2
//             LABEL=(1,SL),
//             DCB=(RECFM=FB,LRECL=135,BLKSIZE=1350)
//SORTOUT   DD DISP=(,CATLG,DELETE),DSN=HMVS02.STATIS.DATA,
//             DCB=(RECFM=FB,LRECL=135,BLKSIZE=23355),
//             SPACE=(CYL,(6),RLSE),UNIT=SYSDA,VOL=SER=MVS385
//SYSIN     DD *
 SORT FIELDS=(1,8,ZD,A)
 RECORD TYPE=F,LENGTH=(135)
 END
/*
//SORTWK01  DD UNIT=SORTDA,SPACE=(CYL,(50,50))
//SORTWK02  DD UNIT=SORTDA,SPACE=(CYL,(50,50))
//
```

Merge 2 disk datasets, creating a disk dataset

```
//SORT     JOB 1,'SORT-3',CLASS=A,MSGCLASS=X
//SORT     EXEC PGM=SORT,REGION=4M,PARM='MSG=AP'
//SYSOUT    DD SYSOUT=*
//SORTLIB   DD DSN=SYSC.SORTLIB,DISP=SHR
//SORTIN01  DD DSN=HMVS02.STATIS.DATA.BL1,DISP=SHR
//SORTIN02  DD DSN=HMVS02.STATIS.DATA.BL2,DISP=SHR
//SORTOUT   DD DISP=(,CATLG,DELETE),DSN=HMVS02.STATIS.DATA,
//             DCB=(RECFM=FB,LRECL=135,BLKSIZE=23355),
//             SPACE=(CYL,(12),RLSE),UNIT=SYSDA,VOL=SER=MVS385
//SYSIN     DD *
 MERGE FIELDS=(50,15,CH,A)
 RECORD TYPE=F,LENGTH=(135)
 END
/*
//
```

Sort disk dataset on split sequence field

```
//SORT     JOB 1,'SORT-4',CLASS=A,MSGCLASS=X
//SORT     EXEC PGM=SORT,REGION=4M,PARM='MSG=AP'
//SYSOUT    DD SYSOUT=*
//SORTLIB   DD DSN=SYSC.SORTLIB,DISP=SHR
//SORTIN    DD DSN=HMVS02.STATIS.DATA,DISP=SHR
//SORTOUT   DD DISP=(,CATLG,DELETE),DSN=HMVS02.STATIS.RVI01,
//             DCB=(RECFM=FB,LRECL=135,BLKSIZE=23355),
//             SPACE=(CYL,(12),RLSE),UNIT=SYSDA,VOL=SER=MVS385
//SYSIN     DD *
 SORT FIELDS=(80,5,CH,D,1,8,ZD,A)
 RECORD TYPE=F,LENGTH=(135)
```

```
 END
/*
//SORTWK01  DD UNIT=SORTDA,SPACE=(CYL,(50,50))
//SORTWK02  DD UNIT=SORTDA,SPACE=(CYL,(50,50))
//SORTWK03  DD UNIT=SORTDA,SPACE=(CYL,(50,50))
//
```

Sort disk dataset on split sequence field, all CH data

```
//SORT     JOB 1,'SORT-5',CLASS=A,MSGCLASS=X
//SORT     EXEC PGM=SORT,REGION=4M,PARM='MSG=AP'
//SYSOUT    DD SYSOUT=*
//SORTLIB   DD DSN=SYSC.SORTLIB,DISP=SHR
//SORTIN    DD DISP=SHR,DSN=HMVS01.DF125A5
//SORTOUT   DD DISP=(,CATLG,DELETE),DSN=HMVS01.DF125A5.SORTED,
//             DCB=(RECFM=FB,LRECL=125,BLKSIZE=23375),
//             SPACE=(CYL,(15),RLSE),UNIT=SYSDA,VOL=SER=MVS380
//SYSIN     DD *
SORT FIELDS=(80,5,A,2,5,D),FORMAT=CH         <- both fields are CH, so able to use alternate f
RECORD TYPE=F,LENGTH=(125)
END
/*
//SORTWK01  DD UNIT=SORTDA,SPACE=(CYL,(50,50))
//SORTWK02  DD UNIT=SORTDA,SPACE=(CYL,(50,50))
//SORTWK03  DD UNIT=SORTDA,SPACE=(CYL,(50,50))
//
```

Source: GC28-6543-8 OS Sort/Merge Program

SUPERLST **[written by R. F. Morse, located in SYSC.LINKLIB]**

Used to provide VTOC listing, with optional Partitioned dataset directory information and volume allocation map. SUPERLST was originally written by R. F. Morse and is now maintained by Greg Price and is located in File #134 of the CBT tape. SUPERLST requires the following DD statements:

- SYSPRINT - defines a sequential output dataset.
- VOL????? - defines the DASD volume to be listed.

Any number of DD statements may be provided, as long as the name begins with 'VOL'. SUPERLST will process all DD names in a single execution.

Program Parameter (PARM= on EXEC statement)

The PARM parameter on the EXEC statement is used to supply optional parameters to the SUPERLST program. The parameters are:

MAP

Specifies that after the dataset name and attribute list (in collating sequence order) a volume space (allocation) map is to be produced.

PDS

Specifies that the directories of all Partitioned datasets found on the volume are to

be opened and read. Statistics such as member and directory block counts are reported. If DUMP was also specified then the contents of the directories are also dumped.

DUMP

Specifies that the report is to also include a hex dump of each VTOC entry.

Control Statements

SUPERLST uses no control statements.

Examples

List VTOCs and PDS directories on two volumes

```
//SUPERLST JOB (0001),SUPERLST,CLASS=S,MSGCLASS=X
//SUPERLST EXEC PGM=SUPERLST,REGION=2048K,PARM=''  PDS/MAP/DUMP
//SYSPRINT DD  SYSOUT=*
//VOLUME1  DD  DISP=SHR,UNIT=SYSALLDA,VOL=SER=PUB001
//VOLUME2  DD  DISP=SHR,UNIT=SYSALLDA,VOL=SER=PUB000
//
```

SYSREPRO **[written by Bill Godfrey, located in SYSC.LINKLIB]**

Used for copying data records from a sequential dataset or member of a partitioned dataset to another sequential dataset or member. It is faster than IEBGENER and prints a summary of the copy operation including the record counts. If the output DD does not have DCB attributes supplied, the attributes will be copied from the input dataset. If the BLKSIZE is omitted, the system determined BLKSIZE will be used. The program was written by Bill Godfrey and the source may be found in File #316 of the CBT Tape. SYSREPRO requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSUT1 - defines the input sequential dataset or partitioned dataset member.
- SYSUT2 - defines the output sequential dataset or partitioned dataset member.

Program Parameter (PARM= on EXEC statement)

The optional PARM is used to modify the operation of SYSREPRO. The values and their function are:

n1,n2,n3,n4

If n1 is specified, that number of records will be copied from the input dataset to the output dataset.

If n2 is specified, that number of records will be skipped before copying begins

If n3 and n4 are specified, only n4 out of every n3 records are to be copied.

Control Statements

SYSREPRO uses no control statements.

Examples

Print a card dataset

```
//SYSREPRO JOB (001),'SYSREPRO-1',CLASS=A,MSGCLASS=X
//SYSREPRO EXEC PGM=SYSREPRO,REGION=512K
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=01C,DCB=(RECFM=FB,LRECL=80,BLKSIZE=80)
//SYSUT2   DD  SYSOUT=*
//
```

Copy a sequential dataset, disk to disk

```
//SYSREPRO JOB (001),'SYSREPRO-2',CLASS=A,MSGCLASS=X
//SYSREPRO EXEC PGM=SYSREPRO,REGION=512K
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DISP=SHR,DSN=HMVS01.SORTDATA.TRE
//SYSUT2   DD  DISP=(,CATLG),DSN=HMVS01.SORTDATA.QUATRO,
//             UNIT=SYSDA,VOL=SER=PUB000,
//             DCB=(JAY01.SORTDATA.TRE),
//             SPACE=(CYL,(12),RLSE)
//
```

Copy a tape dataset to disk

```
//SYSREPRO JOB (001),'SYSREPRO-3',CLASS=A,MSGCLASS=X
//SYSREPRO EXEC PGM=SYSREPRO,REGION=512K
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DISP=OLD,DSN=TEST.D0135.ONE,
//             UNIT=TAPE,VOL=SER=22008C,LABEL=(1,SL)
//SYSUT2   DD  DISP=(,CATLG),DSN=JAY01.SORTTEST.DATA,
//             UNIT=SYSDA,VOL=SER=PUB000,
//             SPACE=(CYL,(12),RLSE),
//             DCB=(JAY01.POLICY.ONE)
//
```

Add member to existing partitioned dataset from cards

```
//SYSREPRO JOB (001),'SYSREPRO-4',CLASS=A,MSGCLASS=X
//SYSREPRO EXEC PGM=SYSREPRO,REGION=512K
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=01C,DCB=BLKSIZE=80
//SYSUT2   DD  DISP=MOD,DSN=JAY01.VSAMIO.SOURCE(NEWPROG)
//
```

TAPEMAP **[originally from UCLA, last modified by Sam Golob, located in SYSC.LINKLIB]**

Used to read a tape volume and report everything it contains, including detailed information about file contents when the creation program is recognized. The version currently installed is 2.5 and the source may be found on File #299 of the CBT Tape. TAPEMAP requires the following DD statements:

- SYSPRINT - defines a sequential output dataset (list of files found on tape).
- SYSPRNT2 - defines a sequential output dataset (detailed information about individual files).
- SYSUT1 - defines the tape volume to be mapped.

Control Statements

TAPEMAP uses no control statements.

Examples

Analyze tape and report on files and their contents

```
//TAPEMAP  JOB (001),'TAPEMAP',CLASS=A,MSGCLASS=X
//TAPEMAP  EXEC PGM=TAPEMAP,REGION=512K,PARM='TEST'
//SYSPRINT  DD SYSOUT=*
//SYSPRNT2  DD SYSOUT=*
//SYSUT1    DD DSN=T000001,
//             VOL=SER=000001,
//             DISP=(OLD,KEEP),
//             LABEL=(1,BLP),
//             UNIT=(TAPE,,DEFER)
//
```

TAPESCAN **[written by Will Daland, located in SYSC.LINKLIB]**

Used to read a tape volume and report an overview of the datasets on a tape, copy files and recover data past the first end of volume indicator. Information includes record and byte count, length estimate, display fo the first 100 bytes of the first records of each dataset, and the physical tape file number. Original author was Will Daland, but modified by others; the source may be found on File #102 of the CBT Tape. **I would suggest taking a look at COPYMODS.**

TAPESCAN requires the following DD statements:

- SYSPRINT - defines a sequential output dataset (list of files found on tape).
- INPUT - defines the tape volume to be read.
- OUTPUT - optionally defines a tape volume to receive copy of input tape volume.

Program Parameter (PARM= on EXEC statement)

The PARM is used to supply all parameters to TAPESCAN. The parameter values are:

COPY

Create a copy of the input datasets, labels, and tape marks on OUTPUT DD. NOCOUNT may not be specified when COPY is specified. The default is COPY off.

EOVMOD

Places the INPUT data to be copied after the last dataset on OUTPUT. This parameter implies COPY. NOCOUNT may not be specified when EOVMOD is specified. The default is EOVMOD off.

ERRLIMn

The maximum number of I/O errors allowed before processing is terminated is specified by n. The default is ERRLIM50.

LISTn

The number of blocks from each dataset which have records printed is specified by n. The default is 4.

MAXEOVn

Processing will continue until n end of volume indicators have been found. Processing will stop when either MAXTMn or MAXEOVn is exceeded. If SKIPEOVn is used, then MAXEOVn must be one greater than SKIPEOVn. The default value is 1.

MAXTMn

Stop processing after n tape marks have been encountered. This will not continue beyond MAXEOVn. The default is to process until MAXEOVn is reached.

NOCOUNT

To save I/O time, the counting feature may be turned off with NOCOUNT. The number of physical blocks, the maximum, average, and minimum block sizes, the length and the total number of bytes processed will not be reported. This parameter may not be used with COPY or EOVMOD. The default is NOCOUNT off.

NOHEX

The hexadecimal equivalent of the blocks printed from each record will not be produced. The default is NOHEX off.

NOLIST

The content of data records will not be printed. The default is NOLIST off.

NOSUMMARY

The summary of datasets for Standard Label tape will not be printed. The default is NOSUMMARY off.

NOVOLSER

During a COPY operation between two Standard Label tapes, the volume header label will not be copied. The default is NOVOLSER off.

SKIPEOVn

Processing will start after n number of end of volume indicators have been encountered. Ensure that MAXEOVn is at least one greater than SKIPEOVn. The default is 0.

SKIPTMn

Start processing after n tape marks have been encountered. Processing will not start past the MAXEOVn. When both SKIPTMn and SKIPEOVn are specified the maximum amount of skipping is done. The default is 0.

VTOC

For a Standard Label tape, produce only the VTOC report.

Control Statements

TAPESCAN uses no control statements.

Examples

Print dataset VTOC for SL tape

```
//TAPESCAN JOB 1,'TAPESCAN-1',CLASS=A,MSGCLASS=X
//TAPESCAN EXEC PGM=TAPESCAN,REGION=1024K,
//   PARM='VTOC'
//SYSPRINT  DD SYSOUT=*
//INPUT     DD DSN=DUMP.JAY01,
//             UNIT=(TAPE,,DEFER),DISP=OLD,VOL=SER=22003A,
//             DCB=(EROPT=ACC),LABEL=(,SL)
//
```

Copy 4th and 5th datasets to second tape

```
//TAPESCAN JOB 1,'TAPESCAN-2',CLASS=A,MSGCLASS=X
//*
//* Copy the 4th and 5th files of a SL tape to the 2nd and 3rd
//* files to a tape which already contains 1 file. Do not copy
//* the volume header record or produce the hex output. List
//* the first 20 records of the input datasets. Stop processing
//* after 5 I/O errors.
//*
//TAPESCAN EXEC PGM=TAPESCAN,REGION=1024K,
//   PARM='COPY,NOVOLSER,EOVMOD,SKIPTM10,MAXTM16,LIST20,NOHEX'
//SYSPRINT  DD SYSOUT=*
//INPUT     DD UNIT=2400-7,DISP=OLD,
//             DCB=(TRTCH=ET,DEN=1,ERROPT=ACC),VOL=SER=TRACK7
//OUTPUT    DD UNIT=2400-4,LABEL=2,VOL=SER=OTPTTP
//
```

UNUPDTE **[originally from SPLA PL1 MODs tape, located in SYSC.LINKLIB]**

Used for sequentializing a partitioned dataset. All members of a partitioned dataset are used to create a sequential dataset; each member is preceded by an ADD control statement. UNUPDTE will process Fixed or Variable length

records, with any LRECL allowed by MVS. The resulting file may then be input to UPDTE, IEBUPDTE, or PDSLOAD to construct a new partitioned dataset containing the offloaded members. The most recent version of UNUPDTE is located in File #093 of the CBT tape. The program was updated by Art Tansky. UNUPDTE requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSUT1 - defines the input partitioned dataset.
- SYSUT2 - defines the output sequential dataset.

I would also suggest taking a look at OFFLOAD.

Control Statements

UNUPDTE uses no control statements.

Examples

Offload members of a partitioned dataset to sequential dataset

```
//UNUPDTE  JOB (01),'UNUPDTE-1',CLASS=S,MSGCLASS=X
//UNUPDTE  EXEC PGM=UNUPDTE,REGION=1024K
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD DISP=SHR,DSN=SYSC.VSAMIO.SOURCE
//SYSUT2    DD DISP=NEW,DSN=SYSC.VSAMIO.SOURCE,
//             UNIT=TAPE,VOL=SER=SCRTCH,
//             DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=3120,DEN=3)
//
```

UPDTE **[originally from SPLA PL1 MODs tape, located in SYSC.LINKLIB]**

Used for recreating a partitioned dataset from a sequential dataset that contains ADD control statements interspersed with data records. UPDTE will process Fixed or Variable length records, with any LRECL allowed by MVS. The most recent version of UPDTE is located in File #093 of the CBT tape. The program was last updated by R. E. Styma. UPDTE requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines the input sequential dataset.
- SYSUT2 - defines the output partitioned dataset.

I would also suggest taking a look at PDSLOAD.

Control Statements

UPDTE uses no control statements.

Examples

Load partitioned dataset from sequential dataset

```
//UPDTE    JOB (SYS),'UPDTE-1',CLASS=S,MSGCLASS=X
//UPDTE    EXEC PGM=UPDTE,REGION=1024K
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD DISP=OLD,DSN=SYSC.VSAMIO.SOURCE,
//             UNIT=TAPE,VOL=SER=22009A,LABEL=(,SL)
//SYSUT2   DD  DISP=(,KEEP,DELETE),DSN=HMVS02.VSAMIO.SOURCE,
//             UNIT=SYSDA,VOL=SER=TST380,
//             SPACE=(TRK,(180,,30),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=23440)
//
```

VTOCLIST **[written by P. E. Havercan, located in SYSC.LINKLIB]**

Used to provide formatted VTOC listing. The most recent version of VTOCLIST is located in File #343 of the CBT Overflow tape. The program was last updated by John Kalinich. VTOCLIST requires the following DD statements:

- SYSPRINT - defines a sequential output dataset.
- SYSUT1 - defines the DASD volume to be listed.

I would also suggest taking a look at SUPERLST.

Program Parameter (PARM= on EXEC statement)

The PARM parameter on the EXEC statement may be used to supply an optional parameter to the VTOCLIST program. The parameter is:

PDS

Specifies that the creation date and last reference date will be replaced with directory blocks allocated and used. Use of the PDS parameter may cause waits for datasets reserved/in use by other programs.

Control Statements

VTOCLIST uses no control statements.

Examples

List VTOC for DASD volume

```
//VTOCLIST JOB (1),VTOCLIST,CLASS=A,MSGCLASS=X
//VTOCLIST EXEC PGM=VTOCLIST'
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD DISP=OLD,UNIT=SYSALLDA,VOL=SER=MVSRES
```

```
//
```

List VTOC for DASD volume with PDS directory information

```
//VTOCLIST JOB (1),VTOCLIST,CLASS=A,MSGCLASS=X
//VTOCLIST EXEC PGM=VTOCLIST,PARM='PDS'
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD DISP=OLD,UNIT=SYSALLDA,VOL=SER=PUB000
//
```

XMIT370 **[written by Jim Morrison, located in SYSC.LINKLIB]**

Used to read partitioned datasets to produce XMIT files equivalent to the TSO/E TRANSMIT command processor or a similar facility. The resulting XMIT dataset can be processed using the TSO/E RECEIVE command processor, Hercules dasdload, RECV370, or any of the other UnXmit programs as detailed at http://planetmvs.com on the UnXmit Information Exchange page. XMIT370 was written by Jim Morrison and is located in File #571 of the CBT tape. RECV370 requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset (IEBCOPY messages).
- XMITLOG - defines a sequential output message dataset (XMIT370 messages).
- SYSIN - unused, but required by IEBCOPY.
- SYSUT1 - defines partitioned input dataset.
- SYSUT2 - work dataset.
- XMITOUT - defines sequential output dataset (XMIT).

Control Statements

XMIT370 uses no control statements.

Examples

Pack partitioned dataset into XMIT

```
//XMIT370  JOB (001),'XMIT370',CLASS=A,MSGCLASS=X
//XMIT370  PROC
//XMIT370  EXEC PGM=XMIT370,REGION=1024K
//XMITLOG   DD SYSOUT=*                    XMIT370 OUTPUT MESSAGES
//SYSPRINT  DD SYSOUT=*                    IEBCOPY OUTPUT MESSAGES
//SYSIN     DD DUMMY                       IEBCOPY REQUIRES
//SYSUT1    DD DDNAME=XMITIN               INPUT DATASET
//SYSUT2    DD UNIT=SYSDA,                 WORK DATASET
//             SPACE=(CYL,(10,10)),
//             DCB=BLKSIZE=5600
//XMITOUT   DD DDNAME=XMITOUT              OUTPUT DATASET
//         PEND
//XMIT1    EXEC XMIT370
//XMITIN   DD  DISP=SHR,DSN=JAY01.COBOL.XMIT     <- INPUT
//SYSUT2   DD  DSN=HMVS01.MVTCOBOL.SOURCE,       <- OUTPUT
//             VOL=SER=MVS381,UNIT=SYSDA,
//             SPACE=(CYL,(5,,15),RLSE),
//             DISP=(,CATLG)
```

```
//
```

ZAPDSCB [written by Dave Phillips, located in SYSC.LINKLIB]

Used to update DSCBs for specified datasets with values supplied on DD statements. The DSCB is updated without regard to the actual dataset format. ZAPDSCB was written by Dave Phillips and is located in File #163 of the CBT tape. ZAPDSCB requires the following DD statements:

- SNAP - defines a sequential output dataset to receive a snap dump if an I/O error occurs.
- ZAP????? - defines datasets whose DSCBs are to be modified.

Any number of DD statements may be provided, as long as the name begins with 'ZAP'.

The values which may be changed are:

- Expiration date - can be set to zero via RETPD=0
- ASM2 use count - can be set to zero via FCB=ZUSE
- Secondary space amount - set via SPACE=(TRK,(0,##))
- DSORG
- RECFM
- BLKSIZE
- LRECL
- KEYLEN
- RKP
- OPTCD
- Password protection - can be removed via FCB=NOPW or set via LABEL=(,,PASSWORD) and LABEL=(,,NOPWREAD)

I would also suggest taking a look at FIXDSCB.

Control Statements

ZAPDSCB uses no control statements.

Examples

Modify DSCB fields for dataset

```
//ZAPDSCB  JOB (001),'ZAPDSCB',CLASS=A,MSGCLASS=X
//ZAPDSCB  EXEC PGM=ZAPDSCB
//SNAP      DD SYSOUT=*     SNAP DUMP WHEN I/O ERROR
//SYSUDUMP  DD SYSOUT=*     ABEND DUMP OTHER ERRORS
//ZAP0000   DD UNIT=3350,VOL=SER=SRC000,DISP=SHR,
//             DSN=MVSSRC.EREPSY.F01,DCB=DSORG=PS
//
```

ZTDUMPTP **[written by Jim Marshall, located in SYSC.LINKLIB]**

Used to print data records from datasets on tape. ZTDUMPTP was written by Jim Marshall and is located in File #316 of the CBT tape. ZTDUMPTP requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSIN - defines the sequential input dataset that contains ZTDUMPTP Control statements.
- FILE - defines the tape volume.

Control Statements

FILES=n

The FILES control statement specifies the number of consecutive files to dump. The default value is 1.

REC=n

The REC control statement specifies the number of consecutive records to dump. The default value is ALL.

SKPFIL=n

The SKPFIL control statement specifies the number of consecutive files to skip. The default value is 0.

SKPREC=n

The SKPREC control statement specifies the number of consecutive records to skip. The default value is 0.

PRINT=YES

The PRINT control statement specifies character format of data record is to be printed alongside hex dump. The default is NO.

RUN=YES

The RUN control statement specifies the tape is to be rewound and unloaded before processing next control card. The default is NO. Allows for processing multiple tapes in a single execution.

REDUN=NO

The REDUN control statement specifies exit to end of job on redundant record. The default is NO. When set to YES, all redundant records are indicated.

MODE=nn

The MODE control statement specifies the tape drive mode set command code.

Examples

Dump and print first 20 records from first 5 files on tape

```
//ZTDUMPTP JOB (1),'ZTDUMPTP',CLASS=A,MSGCLASS=X
//ZTDUMPTP EXEC PGM=ZTDUMPTP
//FILE     DD UNIT=TAPE,DSN=ANYTAPE,VOL=SER=000000,LABEL=(1,BLP)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  FILES=5,REC=20,PRINT=YES
//
```

ZZRELINK **[written by J. Scullion, located in SYSC.LINKLIB]**

Used to re-link-edit one or more existing load modules. ZZRELINK was written by J. Scullion and has been updated by Gerhard Postpischil. The source may be found on CBT Tape File #860 (original in #316). ZZRELINK requires the following DD statements:

- SYSPRINT - defines a sequential output message dataset.
- SYSPUNCH - (optional, required to produce JCL for Link Editor job) defines a sequential output dataset.
- SYSLOUT - (optional, required when Link Editor invoked directly) defines a sequential output dataset.
- SYSUT1 - work dataset
- SYSUT2 - work dataset
- SYSUT3 - work dataset
- SYSIN - defines a sequential input dataset that contains ZZRELINK control statements.

Note: If SYSPUNCH is provided, the JCL is created and punched that will perform the re-link-edit of the modules specified for selection. If SYSPUNCH is **not** provided, the Link Editor is loaded dynamically by ZZRELINK and the modules specified for selection will be re-link-edited immediately.

Control Statements

```
LINKOUTDD=ddname

    ,INDD={ddname | (ddname1[,ddname2]...)}
```

The LINK control statement specifies the DD names to be used for the target library to receive the re-link-edited load modules and to specify the library, or libraries, containing source load modules. The parameters are:

OUTDD=ddname,

Specifies the DD name for the output dataset.

INDD={ddname | (ddname1[,ddname2]...)}

Specifies one or more DD names for the input dataset(s). If more than one ddname is specified, they must be enclosed in parentheses and separated by commas.

If a load module would duplicate one already in the target load library, the existing load module will not be replaced, unless the replace option is specified. To replace any duplicate members encountered, enclose the ddname for which the replace option is to be used in an additional pair of parentheses and follow the ddname with a comma followed by R:

INDD=(ddname1,(ddname2,R),ddname3)

will result in any members in ddname2 that have the same name as a load module already present in the target load library to replace the load module already existing in the target library.

```
SELECT     MEMBER={name | (name1[,name2]...) | ((name,newname)[,(name2,newname2)...)}
```

The SELECT control statement specifies individual members to be copied from the input dataset(s). If more than one member name is specified, they must be enclosed in parentheses and separated by commas. Multiple SELECT statements may be specified, in which case the subsequent statements are processed as a continuation of the first. In order to rename a member, the name must be enclosed in an additional set of parentheses and followed by a comma and the newname.

If no SELECT statement is provided, **all** members of the input load library will be selected for re-link-edit to the target library.

Examples

Create (on SYSPUNCH) JCL for job to relink a load module

```
//ZZRELINK JOB (001),'ZZRELINK-1',CLASS=A,MSGCLASS=X
//ZZRELINK EXEC PGM=ZZRELINK,REGION=4096K
//SYSPRINT  DD SYSOUT=*
//OUTLIB    DD DISP=MOD,DSN=JAY01.RELINK.LOADLIB
//INLIB     DD DISP=SHR,DSN=SYS2.LINKLIB
//SYSPUNCH  DD SYSOUT=*,DCB=(BLKSIZE=80,RECFM=F)
//SYSUT1    DD UNIT=VIO,SPACE=(80,(320,90))
//SYSUT2    DD UNIT=VIO,SPACE=(80,(320,90))
//SYSUT3    DD UNIT=VIO,SPACE=(80,(320,90))
//SYSLOUT   DD SYSOUT=*
//SYSIN     DD  *
  LINK INDD=INLIB,OUTDD=OUTLIB
  SELECT MEMBER=VSIOMODT
/*
//
```

Re-link-edit all members of a load library to a new library

```
//ZZRELINK JOB (001),'ZZRELINK-2',CLASS=A,MSGCLASS=X
//ZZRELINK EXEC PGM=ZZRELINK,REGION=4096K
//SYSPRINT  DD SYSOUT=*
//OUTLIB    DD DISP=(,CATLG),DSN=HMVS02.LOAD
```

```
//INLIB     DD DISP=SHR,DSN=HMVS01.LOAD
//SYSUT1    DD UNIT=VIO,SPACE=(80,(320,90))
//SYSUT2    DD UNIT=VIO,SPACE=(80,(320,90))
//SYSUT3    DD UNIT=VIO,SPACE=(80,(320,90))
//SYSLOUT   DD SYSOUT=*
//SYSIN     DD  *
  LINK INDD=INLIB,OUTDD=OUTLIB
/*
//
```

Re-link-edit a load module, renaming it into the input library

```
//ZZRELINK JOB (001),'ZZRELINK-3',CLASS=A,MSGCLASS=X
//ZZRELINK EXEC PGM=ZZRELINK,REGION=4096K
//SYSPRINT  DD SYSOUT=*
//OUTLIB    DD DISP=MOD,DSN=HMVS01          <- target and
//INLIB     DD DISP=SHR,DSN=HMVS01          <- source library the same
//SYSUT1    DD UNIT=VIO,SPACE=(80,(320,90))
//SYSUT2    DD UNIT=VIO,SPACE=(80,(320,90))
//SYSUT3    DD UNIT=VIO,SPACE=(80,(320,90))
//SYSLOUT   DD SYSOUT=*
//SYSIN     DD  *
  LINK INDD=INLIB,OUTDD=OUTLIB
  SELECT MEMBER=((VSIOMODX,VSIOMODT))
/*
//
```

Re-link-edit a group of modules to a new library, replacing duplicates

```
//ZZRELINK JOB (001),'ZZRELINK-4',CLASS=A,MSGCLASS=X
//ZZRELINK EXEC PGM=ZZRELINK,REGION=4096K
//SYSPRINT  DD SYSOUT=*
//OUTLIB    DD DISP=MOD,DSN=HMVS02
//INLIB     DD DISP=SHR,DSN=SYS2.LINKLIB
//SYSUT1    DD UNIT=VIO,SPACE=(80,(320,90))
//SYSUT2    DD UNIT=VIO,SPACE=(80,(320,90))
//SYSUT3    DD UNIT=VIO,SPACE=(80,(320,90))
//SYSLOUT   DD SYSOUT=*
//SYSIN     DD  *
  LINK INDD=(INLIB,R),OUTDD=OUTLIB
  SELECT MEMBER=(UPDMAIN,UPDTSUB1,UPDTSUB2)
  SELECT MEMBER=(UPDSUB3,UPDTSUB4,UPDTSUB5)
/*
//
```

Source: http://www.jaymoseley.com/hercules/os_utilities/mvs_utilities.html
Document created: February 7, 2022
