**JCL Notes**

**sites**

**Tutorials**

<http://mainframeibm.blogspot.in/2009/04/job-card-parameters-jcl.html>

**Abends**

<http://www.ruifeio.com/2011/12/03/most-common-jcl-error-codes/>

http://mainframe-tips-and-tricks.blogspot.com.au/2011/12/jcl-how-to-fix-b37-abend.html

**Utilities**

http://www.mainframegurukul.com/srcsinc/drona/programming/languages/jcl/jcl.sort.html

**Example Programs**

-----------------------------------------------------------------------------------------------------------------------------

**Imp Points**

* **ABBREVATIONS**

**JES** - Job Entry Subsystem

**JCL –** Job Control Language

**JESMSGLG -** Job Entry Subsystem Message Log

**JESJCL -** Job Entry SubsystemJob Control Language

**JESYSMSG -** Job Entry Subsystem System Message

* A programmer codes the job CLASS in JCL code. Jobs with similar characteristics should have the same CLASS. Typically, every installation sets up categories or job classes, for instance **CLASS=A for small jobs, CLASS=G for long-running jobs, CLASS=T for tape jobs CLASS=K for quickly running the jobs.**
* **Continuation for jcl statements and job card must begin on orbefore column 16**
* **SYSIN parameters in case of file we give input from firstcolumn**
* **SYSIN parameters in case of utilities we start from 2nd column**
* **SYSIN parameters**

**1.continuation character is “–“ for IDCAMS**

**2. continuation character for sort is “,”**

* **Common JCL abends are S222, S806, SOCB, SOC7, SOC4, SOC1, SE37, SB37, SD37**
* **MSG LOGS AFTER EXECUTION OF JCL**

**JESMSGLG-**Contains all steps and return codes in tabular format

**JESJCL -**Contains Only JCL written for submission

**JESYSMSG -**Contains all steps and return codes expanded for each step and corresponding System messages

* **JOB cards**

**NAB**

//UGBPCRPS JOB AF-1055,'CREATE PS',

// MSGCLASS=X,MSGLEVEL=(1,1),

// CLASS=P,NOTIFY=&SYSUID

* **Using temp files injcl**

//SORT01 DD DSN=&&TEMP,DISP=(NEW,PASS,DELETE)

* **Using sort work files**

//SORTWK01 DD DISP=(NEW,DELETE),

// UNIT=SYSDA,SPACE=(CYL,(3335,3335))

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

----------------------------------END--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Questions and their Explanation**

**Q.Cobol compile job**

**ANS.**

//UGBCOMPC JOB AF-1055,'SKIP 7RC',

// MSGCLASS=X,MSGLEVEL=(1,1),

// CLASS=P,NOTIFY=&SYSUID

//\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* CHANGE THE MEM NAME TO YOUR PRORAM NAME

//\* CHANGE THE SYSIN LIBRARY AS YOUR PROGRAM LIBRARY

//\* CHANGE THE SYSLIB LIBRARY AS YOUR COPY LIBRARY

//\* CHANGE THE SYSLMOD LIBRARY AS YOUR LOAD MODULE LIBRARY

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*

//JOBPROC JCLLIB ORDER=SDAF.P732083.NAMI.JCL.CNTL

//COBCL EXEC **COBCL**,MEM=READSE

//**COMPILE.SYSIN** DD DSN=SDAF.P732083.NAMI.COBOL.LIBRARY(&MEM),

// DISP=SHR

//**COMPILE.SYSLIB** DD DSN=SDAF.P732083.NAMI.COPY.LIBRARY,DISP=SHR

/**/LKED.SYSLMOD** DD DSN=SDAF.P732083.NAMI.LOADLIB(&MEM),DISP=SHR

//

**The Proc COBCL**

//COBCL PROC LNGPRFX='SYS1',SYSLBLK=0000,LIBPRFX='SYS1'

//\*

**//COMPILE EXEC PGM=IGYCRCTL,REGION=2048K,**

**// PARM='RES,LIB,OBJECT,APOST,TEST,DYNAM,LIST'**

**//STEPLIB DD DSNAME=&LNGPRFX..SIGYCOMP,DISP=SHR**

//SYSPRINT DD SYSOUT=\*

//SYSLIN DD DSNAME=&&LOADSET,UNIT=SYSALLDA,

// DISP=(MOD,PASS),SPACE=(TRK,(3,3)),

// DCB=(BLKSIZE=&SYSLBLK)

//SYSUT1 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))

//SYSUT2 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))

//SYSUT3 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))

//SYSUT4 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))

//SYSUT5 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))

//SYSUT6 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))

//SYSUT7 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))

//\*

**//LKED EXEC PGM=HEWL,**

**// COND=(8,LT,COMPILE),REGION=1024K,PARM='XREF'**

**//SYSLIB DD DSNAME=&LIBPRFX..SCEELKED,DISP=SHR**

//SYSPRINT DD SYSOUT=\*

//SYSLIN DD DSNAME=&&LOADSET,DISP=(OLD,DELETE)

// DD DDNAME=SYSIN

//SYSUT1 DD UNIT=SYSALLDA,SPACE=(TRK,(10,10))

**Q.Cobol Run job**

**ANS.**

//UGB0RUNC JOB AF-1055,'Cob Run’,

// MSGCLASS=X,MSGLEVEL=(1,1),

// CLASS=P,NOTIFY=&SYSUID

//\*

**//JOBLIB DD DSN=SDAF.P732083.NAMI.LOADLIB,**

**// DISP=SHR**

//\*

**//STEP001 EXEC PGM=COBSAMP**

//SYSOUT DD SYSOUT=\*

//SYSPRINT DD SYSOUT=\*

//SYSUDUMP DD SYSOUT=\*

//SYSABEND DD SYSOUT=\*

//SYSDBOUT DD SYSOUT=\*

//SYSIN DD \*

S0

/\*

//

**Q.JOB CARD PARAMETERS:**

**ANS-Accounting information:** It is appositional parameter used to identify the account number which will be charged for utilizing CPU time by the job on mainframe.

It can be of 1 to 143 characters long

Syntax: account number

                ‘Account number, additional info’

                ‘Account number’ in case of special characters

EG: AF-1055(FOR cams)

**Programmer name:** It is appositional parameter used to specify the name of programmer who submits the job.

**It can be of 1 to 20 characters long**

 Syntax:  name

               ‘NAMI’

**CLASS:**  It is a key word parameter used to assign a class to the job. it can be a to z or 0 to 9

It is installation dependent.

Syntax: class=P(CAMS), CLASS=2(VISIONPLUS)

**NOTIFY:** it is keyword parameter used to specify the user id to which the status of job has to be notified.

Syntax NOTIFY=USERID

             NOTIFY=&SYSUID (The login USERID)

**MSGCLASS:** It is a keyword parameter used to specify how the JCL and output messages should be handled. It can be A to Z and 0 to 9.it is installation dependent.

Syntax: MSGCLASS=A

               MSGCLASS=9

MSGCLASS=X(CAMS,VPLUS)

**MSGLEVEL:** It is a keyword parameter used to specify how much of the JCL and output messages should be printed.

Syntax: **MSGLEVEL = (statements, messages)**

 The value of statements can be 0, 1 and 2

0 = print only job card

1=print JCL and procedure statements

2=print only JCL messages

**The value of messages can be 0, 1**

0=print O/P messages only if job abends

1=print O/P messages regardless of job outcome

Eg: MSGLEVEL(1,1) FOR CAMS,VPLUS

**REGION:** It is a key word parameter used to specify the amount of memory that the job can utilize for execution.

Syntax: REGION=1024K or 1024M

If the job utilizes more the memory specified in job card the job will be terminated abnormally.

We can specify the region parameter in exec statements also. In that case region will be limited to that particular execution statement only

   Ex: //STEP1 EXEC PGM=PGM1, REGION=1024K

**If we specify the region parameter in both job card and execution statement the value in job card overrides the value in EXEC statements.**

   Ex: //MTPLH01A JOB NOTIFY=MTLH01, REGION=1024K

     //STEP1 EXEC PGM=ADD, REGION=24K

     //STEP2 EXEC PGM=PGM2, REGION=34K

Step1 and step2 will be allocated 1024KB of memory

**TIME:** It is a key word parameter used to limit the time that the job can utilize.

Syntax: **TIME = (M, SEC) M= 1 to 1439 and SEC= 1 to 59**

If TIME= (M,) hen SEC =0

If TIME = (, SEC) then M=0

If the job utilizes more time than specified in job card the job will be terminated abnormally.

We can specify the TIME parameter in exec statements also. In that case time will be limited to that particular execution statement only.

Ex: //STEP1 EXEC PGM=PGM1, TIME= (1, 20)

**If we specify the TIME parameter in both job card and execution statement the value in execution statement overrides the value in job statement.**

Ex: //MTPLH01A JOB NOTIFY=MTLH01, TIME= (2, 40)

     //STEP1 EXEC PGM=ADD, TIME= (1, 20)

     //STEP2 EXEC PGM=PGM2, TIME= (1, 20)

Step1 will be allocated                 1 minute 20 seconds

Step1 will be allocated                 1 minute 20 seconds

**TYPRUN:** It is a keyword parameter used **to control the type of execution**.

Syntax TYPRUN=HOLD

It checks for syntax errors and the job will be held in the input queue .it will be executed when the user release the job.

TYPRUN=SCAN

It only checks for syntax errors.

TYPRUN=COPY

It copies the source content to the output device specified in message class parameter.

**RESTART:** it is a keyword parameter used to start the job from a step specified in the parameter other than the first step.

Syntax **RESTART=STEP3**

**Q.EXEC STATEMENT PARAMETERS:**

Ans.**We can write maximum of 255 steps.**

**PGM:**it is a key word parameter used to specify the program name that is to be executed.

Syntax PGM=ADD

**PARM:**it is a key word parameter used to pass the information to program that is being executed in the step.

Syntax PARM= ’PRINT’

             PARM=’PRINT, PASS’

**COND:** it is a key word parameter used to control the execution of subsequent steps.

**Condition parameter on job card**:

Syntax: COND= (COMPARISION CODE, CONDITON)

When executing each and every step the comparison code will be compared against the return code of prior step

Comparison code: this is a number from 0 t0 4095

Condition: GE, GT, LE, LT, NE, EQ

The first step will be executed normally. The remaining steps will be executed based on the condition.

Ex COND= (4, EQ)

4 = return code of previous step

If the condition is true the execution of second step will be skipped else it will be executed

Condition parameter on job card:

Syntax: COND= (COMPARISION CODE, CONDITON, STEPNAME)

STEPNAME = name of the step whose condition code is to be compared.

The comparison code will be compared against the return code of step which is mentioned

Ex:  COND= (4, EQ, STEP4)

4 = return code of STEP4

If the condition code is true the step on which the condition is coded will be skipped else it will be executed.

If condition is coded   both the statements job card condition overrides the condition code of step.

**DD STATEMENT PARAMETERS:**

We can specify maximum of 3273 DD statements

DISP: it is a key word parameter that specifies the current status of the dataset and what steps should be taken if the job terminates abnormally or normally

Syntax: DISP= (STATUS, NORMAL DISP, ABNORMAL DISP)

STATUS

**NEW, OLD, SHR, MOD**

NEW : The dataset is new

OLD: the dataset is old and the dataset will be used by the job exclusively

SHR: the dataset is old and the dataset will be used by the other jobs also at the same time

MOD: if he dataset is existing the data will be appended else it will be treated as new

NORMAL DISP

CATLG, UNCATLG, KEEP, PASS, DELETE

CATLG: After normal termination the dataset will be cataloged

UNCATLG: After normal termination the dataset will be uncataloged

DELETE: after normal termination the dataset will be deleted

KEEP: after normal termination the dataset will be kept

PASS: after normal termination the dataset will be passed to subsequent step

ABNORMAL DISP:

CATLG, UNCATLG, KEEP, DELETE

CATLG: After abnormal termination the dataset will be cataloged

UNCATLG: After abnormal termination the dataset will be uncataloged

DELETE: after abnormal termination the dataset will be deleted

KEEP: after abnormal termination the dataset will be kept

**SPACE:**it is keyword parameter used to allocate the memory for the dataset.

Syntax: SPACE= (UNIT, (P, S, D), RLSE)

UNIT=TRKS, CYL, number of blocks

P=primary allocation

S=secondary allocation

D=number of directory blocks in case of PDS otherwise it is 0

RLSE: If there is any unused space it will be released.

Eg: SPACE=(TRK,(5,5),RLSE)

**DSN OR DSNAME:**it is a keyword parameter used to specify a dataset name that the program is using

It can be PS or PDS.

Ex: DSN=SDAF.NANI.IN.SE

Ex: DSN=MTPLH01.DINU.PDS (M1)

**DCB:**it is a key word parameter used to specify the characteristics of dataset that is being used.

Syntax: DCB= (LRECL=N, BLKSIZE=M, RECFM=F/V/FB/VB/V)

N can be 1 to 32760 bytes

M can be 18 to 32760 bytes

**UNIT:**it is keyword parameter used to specify the unit name that the dataset is resided on.

Syntax: UNIT=SYSDA or SYSSQ, TAPE

**VOL=SER=**it is key word parameter used to specify the volume serial number in which the dataset is resided or to be created.

Syntax: VOL=SER=USER32

Q.**SPECIAL DD STATEMENTS**

**Ans.**

**JOBLIB:**it is special statement used to specify a PDS library where the load modules that is being executed by all the steps of the job is located

It should always be after job and before execution statement

Syntax:  //JOBLIB DD DSN=MTPLH01.GTP64.PDS, DISP=SHR

**STEPLIB:**it is special statement used to specify a PDS library where the load modules that is being executed by the particular step is located

It should always be after step statement

Syntax:  //STEPLIB DD DSN=MTPLH01.GTP64.PDS, DISP=SHR

If both the statements are specified then the STEPLIB overrides the JOBLIB.

**SYSPRINT:**it is a special DD statement used to print the system messages to the output device

Syntax: //SYSPRINT DD SYSOUT=\* star refers to the value in message class parameter

              //SYSPRINT DD DSN=A.B.C, DISP=SHR

**SYSOUT:**it is a special DD statement used to print the output messages to the output device

Syntax: //SYSOUT DD SYSOUT=\* star refers to the value in message class parameter

              //SYSOUT DD DSN=A.B.C, DISP=SHR

**SYSIN:**it is a special DD statement used to print supply the INPUT to the program that is being executed

Syntax: //SYSIN DD \*

               1

               2

Q.**How a job is scheduled for execution**

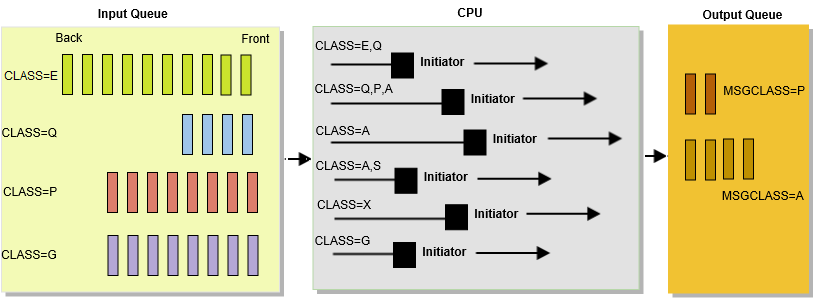
**ANS-**A programmer codes the job CLASS in JCL code. Jobs with similar characteristics should have the same CLASS. Typically, every installation sets up categories or job classes, for instance CLASS=A for small jobs, CLASS=G for long-running jobs, CLASS=T for tape jobs.

A special type of a program called an **initiator** picks up jobs for execution from JES spool. The initiator is like a *nanny* that selects a job from the JES spool, executes the job in its address space and returns to the JES spool for another job. Every initiator has a *class list*. In a hypothetical system, say there are three initiators - INIT1 A, INIT2 B,C,D, INIT3 B,C. An initiator selects jobs from only those classes **which are on its class list**. Every initiator can run only one program at a time. In the example, only one CLASS=A can run at a time. Two CLASS-B jobs can run at a time. The installation can control the number of initiators(JES managed).

The initiator goes through three phrases - allocation, execution and unallocation. First, it invokes*allocation routines* that analyze the SWA control blocks to see what resources(volumes, datasets) the job step needs and they're allocated. Next, the initiator creates a user region where the user-program can execute and loads the program into the region and transfers control to it. When the program completes, the initiator invokes the *unallocation routines* to release any resources.

As the program executes, it can produce output data that needs to be held in JES spool and printed later. This is called SYSOUT dataset. Three other SYSOUT datasets are produced - JES output messages, a JCL listing and a system message log that lists any messages issued by zOS as the job executed. These SYSOUT datasets are brought to the **output queue** in the JES spool.

Like jobs, SYSOUT datasets are each assigned an *output class*. Output classes usually indicate the printer or printers to which the output is sent. Sometimes, an output class MSGCLASS=H may however specify that output is not to be printed. Instead, it is held in the output queue, until it is purged.



**Q. What does one understand by the phrase – Formatting of Output Records? Why do we need to format them?**

When we generate an Output using **DFSORT** sometimes, the records are not in a presentable format. For example, in the output records, there might not be a space between two fields. Or you might want to insert a $ sign for the currency field. You might want to re-order the fields, or display only selected fields/columns in the Output. This type of formatting of records can be done using DFSORT.

You can format the records to be sorted in a particular way, prior to the sorting. This is also possible. So, for better presentation/easier understanding of the report, we re-format the records produced in Output.

For re-formatting of records, there are mainly 3 tools at your disposal -   
**BUILD** - Reformat each record by specifying all of its items one by one. Here, you can add,delete or re-arrange fields.

**OVERLAY** - It allows you to change (edit) the values in specific columns.

**IFTHEN** - Reformats different records in different ways.

The above re-formatting tools can be used with INREC, OUTREC and OUTFIL.

**INREC** - Used to reformat the records before applying sort.   
**OUTREC** - Used to reformat the records after applying sort.

**Q. How to generate a Report, which displays only specific(desired) columns in the Output?**

Suppose you are given the following Input Dataset.

|  |
| --- |
| ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--  AG10001  RAMESH CHANDRA GUHA INDIA     MAHARASHT MUMBAI    01000         AG10002  QUASAR SHABBIR CHUN INDIA     MAHARASHT PUNE      02000         AG10003  VEERAMANI DARYAGANJ INDIA     MAHARASHT NAGPUR    03000         AG10004  DELPHAN FERANDAZ    INDIA     TAMIL NAD CHENNAI   04000         AG10005  PADMAVATHI SUNDARAM INDIA     TAMIL NAD SALEM     05000         AG10006  NEELESHWARI IYER    INDIA     TAMIL NAD ERNAKULAM 06000         AG10007  AJAY NANUBHAI PARSA INDIA     WEST BENG KOLKATA   07000         AG10008  MEHUL POPATLAL SUTA INDIA     WEST BENG KHARAGPUR 08000         AG10009  KRITHIKA RAMANUJAM  INDIA     WEST BENG SUNDARBAN 09000         AG10010  ANKUR CHOUDHARY     INDIA     PUNJAB    AMRITSAR  10000         AG10011  HIREN NITIN SHETH   INDIA     PUNJAB    JALANDHAR 11000         AG10012  DARSHAN SHAH        INDIA     HARYANA   CHANDIGAR 12000 |

In the output report, you are asked to display only the name of the Insurance Agent and the Sales Figure. Since, we want reformat records after applying SORT, we will OUTREC FIELDS Control statement.

We want to have two columns/fields in the ouput -   
NAME   Start-Position 10    Length 20   
SALES  Start-Position 60    Length 05   
  
Thus, we can write the Batch JOB/JCL for the above requirement as follows :

|  |
| --- |
| ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--  //CAA0176J JOB A123,QUASAR,NOTIFY=&SYSUID                                //STEP01 EXEC  PGM=SORT,REGION=1024K                                     //SORTIN DD    DSN=CAA0176.DEMO.SORTIN,DISP=SHR                          //SYSPRINT DD  SYSOUT=\*                                                  //SYSOUT DD    SYSOUT=\*                                                  //SORTOUT DD   DSN=CAA0176.DEMO.SORTOUT,DISP=(OLD,KEEP,KEEP)             //SYSIN  DD    \*                                                           SORT FIELDS=COPY -                                                       OUTREC FIELDS=(10,20,60,5)                                             // |

Note that, we have written SORT FIELDS=COPY, which simply copies the Input Records to the Output, and displays NAME and SALES Columns. Upon submitting the above Batch JOB/JCL, we get the following Output :

|  |
| --- |
| ----+----1----+----2----+  RAMESH CHANDRA GUHA 01000  QUASAR SHABBIR CHUN 02000  VEERAMANI DARYAGANJ 03000  DELPHAN FERANDAZ    04000  PADMAVATHI SUNDARAM 05000  NEELESHWARI IYER    06000  AJAY NANUBHAI PARSA 07000  MEHUL POPATLAL SUTA 08000  KRITHIKA RAMANUJAM  09000  ANKUR CHOUDHARY     10000  HIREN NITIN SHETH   11000  DARSHAN SHAH        12000 |

A lot more formatting is possible on the above report. Read on to know more...

**Q. What if I wanted the SALES field/column to be displayed before(first) and then the NAME Field/Column? How do you re-order columns in the Output Report?**

If we would like to display the SALES Column first, and then the agent-names, we can simply change the order in OUTREC FIELDS statement.

**SORT FIELDS=COPY -                                                      
  OUTREC FIELDS=(60,5,10,20)**   
  
Upon submitting a Batch JOB/JCL with the above control statements, we get the following output :

|  |
| --- |
| ----+----1----+----2----+  01000RAMESH CHANDRA GUHA  02000QUASAR SHABBIR CHUN  03000VEERAMANI DARYAGANJ  04000DELPHAN FERANDAZ     05000PADMAVATHI SUNDARAM  06000NEELESHWARI IYER     07000AJAY NANUBHAI PARSA  08000MEHUL POPATLAL SUTA  09000KRITHIKA RAMANUJAM   10000ANKUR CHOUDHARY      11000HIREN NITIN SHETH    12000DARSHAN SHAH |

**Q.Explain DD STATEMENT PARAMETERS**

**A. We** can specify maximum of 3273 DD statements

**DISP:** it is a key word parameter that specifies the current status of the dataset and what steps should be taken if the job terminates abnormally or normally

Syntax: DISP= (STATUS, NORMAL DISP, ABNORMAL DISP)

**STATUS**

NEW, OLD, SHR, MOD

NEW : The dataset is new

OLD: the dataset is old and the dataset will be used by the job exclusively

SHR: the dataset is old and the dataset will be used by the other jobs also at the same time

MOD: if he dataset is existing the data will be appended else it will be treated as new

**NORMAL DISP**

CATLG, UNCATLG, KEEP, PASS, DELETE

CATLG: After normal termination the dataset will be cataloged

UNCATLG: After normal termination the dataset will be uncataloged

DELETE: after normal termination the dataset will be deleted

KEEP: after normal termination the dataset will be kept

PASS: after normal termination the dataset will be passed to subsequent step

**ABNORMAL DISP:**

CATLG, UNCATLG, KEEP, DELETE

CATLG: After abnormal termination the dataset will be cataloged

UNCATLG: After abnormal termination the dataset will be uncataloged

DELETE: after abnormal termination the dataset will be deleted

KEEP: after abnormal termination the dataset will be kept

**SPACE:** it is keyword parameter used to allocate the memory for the dataset.

Syntax: SPACE= (UNIT, (P, S, D), RLSE)

UNIT=TRKS, CYL, number of blocks

P=primary allocation

S=secondary allocation

D=number of directory blocks in case of PDS otherwise it is 0

RLSE: If there is any unused space it will be released.

**DSN OR DSNAME:** it is a keyword parameter used to specify a dataset name that the program is using

It can be PS or PDS.

Ex: DSN=MTPLH01.DINU.PS

Ex: DSN=MTPLH01.DINU.PDS (M1)

**DCB:** it is a key word parameter used to specify the characteristics of dataset that is being used.

Syntax: DCB= (LRECL=N, BLKSIZE=M, RECFM=F/V/FB/VB/V)

N can be 1 to 32760 bytes

M can be 18 to 32760 bytes

**UNIT:** it is keyword parameter used to specify the unit name that the dataset is resided on.

Syntax: **UNIT=SYSDA or SYSSQ, TAPE**

**VOL=SER=** it is key word parameter used to specify the volume serial number in which the dataset is resided or to be created.

Syntax: VOL=SER=USER32

**SPECIAL DD STATEMENTS:**

**JOBLIB:** it is special statement used to specify a PDS library where the load modules that is being executed by all the steps of the job is located

It should always be after job and before execution statement

Syntax: //JOBLIB DD DSN=MTPLH01.GTP64.PDS, DISP=SHR

**STEPLIB:** it is special statement used to specify a PDS library where the load modules that is being executed by the particular step is located

It should always be after step statement

Syntax: //STEPLIB DD DSN=MTPLH01.GTP64.PDS, DISP=SHR

**If both the statements are specified then the STEPLIB overrides the JOBLIB.**

**SYSPRINT:** it is a special DD statement used to print the system messages to the output device

Syntax: //SYSPRINT DD SYSOUT=\* star refers to the value in message class parameter

//SYSPRINT DD DSN=A.B.C, DISP=SHR

**SYSOUT:** it is a special DD statement used to print the output messages to the output device

Syntax: //SYSOUT DD SYSOUT=\* star refers to the value in message class parameter

//SYSOUT DD DSN=A.B.C, DISP=SHR

**SYSIN:** it is a special DD statement used to print supply the output to the program that is being executed

Syntax: //SYSIN DD \*

1

2

**Utilities**

**SORT**

**Standard Sort Step**

//STEP10 EXEC PGM=SORT,REGION=1024K,PARM=parameters

//SYSOUT DD SYSOUT=\* **Output messages from SORT**

//SORTIN DD DSN=...,DISP=SHR **Input if SORT request(\*mandatory)**

//SORTOUT DD DSN=... **Output for SORT request (\*mandatory)**

//SORTOFxx DD DSN=... **OUTFILE output data sets**

//SORTXSUM DD DSN=... **Output eliminated by the SUM stm**

//SORTWKnn DD UNIT=SYSDA, **Work files if SORT request**

//SYSIN DD \* **Control statement input data set**

sort control statements

/\*

**Basic Sort**

//UGBSORT1 JOB AF-1055,'SORT ',

// MSGCLASS=X,MSGLEVEL=(1,1),

// CLASS=P,NOTIFY=&SYSUID

//\*

//SORT001 EXEC PGM=SORT

//SYSOUT DD SYSOUT=\*

//SYSPRINT DD SYSOUT=\*

//\*BASIC SORT COPY SORTED FILE IN SAME INPUT

//SORTIN DD DSN=SDAF.NANI.P732083.INPUT.SE,

// DISP=SHR

//SORTOUT DD SYSOUT=\*

//SYSIN DD \*

SORT FIELDS=(1,3,ZD,D,4,10,CH,A)

/\*

**Sort Card**

//SYSIN DD \*

SORT FIELDS=(1,3,ZD,D,4,10,CH,A)

/\*

**Explanation**

Above syntax of SORT sorted the recrods, depends

on keys we have provided

(we have provided two keys in FIELDS parameter)

FIRST KEY

1,3,ZD,D - first key started at col 1 , its length is 3

SECOND KEY

4,10,CH,A - second key started at col 4, its length is 10

In the above example,

CH- means character we may use BI for binary ZD for decimal

A - Ascending order

D- Descending Order

**SORT OMIT**

if you wish to exclude certain records which match a specific criteria.(Using OMIT)

**Sort Card**

**SORT** FIELDS=COPY

OMIT COND=((4,10,CH,EQ,**C'SANDEEP'**),AND,(74,1,CH,EQ,C' '))

FIRST Selection

4,10,CH,- Selection started col 4 , its length is 10

**Note:Please note that characters are preceded by datatype i.e C’SANDEEP’ for character**

**SANDEEP**

if you wish to exclude certain records which match a specific criteria.(Using OMIT)

**SORT** FIELDS=COPY

OMIT COND=((34,2,CH,EQ,C'AB'),AND,(74,1,CH,EQ,C' '))

The above SORT card will exclude records which have 'AB' at 34th position and have spaces at 74th position.

- See more at: <http://mainframewizard.com/content/sort-jcl-omit#sthash.z397x5Op.dpuf>

**Using OUTFIL**

.

.

.

//SORTOF01 DD DSN=dataset1,

// DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,

// SPACE=(CYL,(1,4),RLSE),

// DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)

//SORTOF02 DD DSN=dataset2,

// DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,

// SPACE=(CYL,(1,4),RLSE),

// DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)

//SORTOF03 DD DSN=dataset3,

// DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,

// SPACE=(CYL,(1,4),RLSE),

// DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)

.

.

.

//SYSIN DD \*

SORT FIELDS=COPY

OUTFIL FILES=01,INCLUDE=(1,6,CH,EQ,C'MOHANK')

OUTFIL FILES=02,INCLUDE=(1,6,CH,EQ,C'SURESH')

OUTFIL FILES=03,INCLUDE=(1,6,CH,EQ,C'KRISHN')

/\*

1. SORT FIELDS=COPY - indicate , it for copy of records, not for sort

2. OUTFIL FILES=01,INCLUDE=(1,6,CH,EQ,C'MOHANK')

OUTFIL FILES=02,INCLUDE=(1,6,CH,EQ,C'SURESH')

OUTFIL FILES=03,INCLUDE=(1,6,CH,EQ,C'KRISHN')

- SYNCSORT will take data from 1st positioon to 6th position of input

file and it will compare that data with MOHANK or SURESH or KRISHN

- If data equals to MOHANK then that recorrd will copies to dataset defined

in SORTOF01 step. ( because we defined FILES=01 in second condition )

- If data equals to SURESH then that recorrd will pass to dataset defined

in SORTOF02 step. ( because we defined FILES=02 in second condition )

- If data equals to KRISHN then that recorrd will copied to dataset difned

in SORTOF03 step. ( because we defined FILES=03 in third condition )

**We can split the files using OUTFIL option. Check below example.**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//SPLITFLS EXEC PGM=SORT

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//SYSPRINT DD SYSOUT=\*

//SYSOUT DD SYSOUT=\*

//SYSUDUMP DD SYSOUT=\*

//SORTIN DD DSN=FILE1......,DISP=SHR

//SORTOF01 DD DSN=OUTPUTFILE1.......,

// DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,

// SPACE=(CYL,(1,1),RLSE),

// RECFM=FB,LRECL=20

//SORTOF02 DD DSN=OUTPUTFILE2..............,

// DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,

// SPACE=(CYL,(1,1),RLSE),

// RECFM=FB,LRECL=20

//SORTOF03 DD DSN=OUTPUTFILE3..............,

// DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,

// SPACE=(CYL,(1,1),RLSE),

// RECFM=FB,LRECL=20

//SYSIN DD \*

SORT FIELDS=COPY

OUTFIL FILES=01,ENDREC=200

OUTFIL FILES=02,STARTREC=201,ENDREC=400

OUTFIL FILES=03,STARTREC=401,ENDREC=700

/\*

The SORT program is used to sort data into a certain sequence or to merge from 2 to 100 previously sorted input data sets into 1 output data set.

//STEP10 EXEC PGM=SORT

//SYSOUT DD SYSOUT=\*

//SYSPRINT DD SYSOUT=\*

//SYSUDUMP DD SYSOUT=\*

//SORTIN DD DISP=SHR,DSN=<<>>

//SORTOUT DD DISP=(MOD,CATLG,DELETE),SPACE=(CYL,(5,5),RLSE),

// UNIT=SYSDA,DCB=\*.SORTIN,

// DSN=<<>>

//SYSIN DD \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\* INCLUDE SORT STEPS HERE

<!--[if !supportLists]-->1) <!--[endif]-->TO SORT ON POSITIONS say for eg. 1 to 7

SORT FIELDS=(1,7,CH,A)

Where

Sort fields = (position ,length ,format ,sequence) or

Sort fields = (position ,length , sequence..),format = CH/BI/PD/ZD.d

PD=packed Decimal(COMP-3), ZD=zone decimal.

NOTE :-

Instead of using JCL to perform SORT operation , there's one simple alternative,

For eg:- Open a Flat file in edit mode. On the command line type (say) SORT 1,7 and press ENTER, the file will be sorted on positions 1 to 7 bytes.

2) TO COPY ALL THE RECORDS FROM INPUT FILE TO OUTPUT FILE

SORT FIELDS=COPY

3) TO COPY THOSE RECORDS WHICH SATISFY A PARTICULAR CONDITION.

INCLUDE COND=(38,10,CH,EQ,C'57071509',OR,36,10,CH,EQ,C' 57105779')

4) TO OMIT THOSE RECORDS WHICH SATISFY A PARTICULAR CONDITION.

OMIT COND=(19,1,CH,EQ,C'S',OR,19,1,CH,EQ,C'S')

5) TO SKIP CERTAIN NO OF RECORDS

SORT FIELDS=COPY,SKIPREC=1000

6) TO STOP AFTER COPYING CERTAIN NO OF RECORDS

SORT FIELDS=COPY,STOPAFT=5000

7) SKIPREC AND STOPREC CAN BE USED IN COMBINATION

SORT FIELDS=COPY,SKIPREC=1000,STOPAFT=5000

<!--[if !supportLists]-->8) <!--[endif]-->TO REMOVE DUPLICATES FROM THE FILE USING SORT

SORT FIELDS=(1,7,A),FORMAT=CH

SUM FIELDS=NONE

## MERGE

The MERGE control statement defines the application as a MERGE application.

MERGE FIELDS=...

{,FILES=n}

{,EQUALS | NOEQUALS}

{,CKPT | CHKPT}

{,CENTWIN={0 | s | f}}

where

**FIELDS**=(pos1,len1,opt1,pos2,len2,opt2,...),**FORMAT**=type

**Keyword explanations:**

The **FIELDS**= keyword is used to identify the fields to use as merge keys. Each field is described using 4 values:

'pos', its position in the record, relative to 1;

'len', the field's length;

'type', the type of data stored in the field; and

'opt', the sort order for the field which can be

A for ascending, D for descending, or

E as modified by an E61 exit.

Up to 128 fields can be sorted or merged using one

MERGE control statement.

Please note in sorting if we are sorting two files say

sortin and sortout both should have same record length. Also note that during writing of omit cond /include cond the position of input files is same if it’s a FB file other wise for VB files position must be added a 4 offset andwritten in code

There are many types of variable position/length fields such as delimited fields, comma separated values (CSV), tab separated values, blank separated values, keyword separated fields, and so on. For example, you might have four records with comma separated values as follows:

Wayne,M,-53,-1732,Gotham

Summers,F,+7258,-273,Sunnydale

Kent,M,+213,-158,Metropolis

Prince,F,-164,+1289,Gateway

Note that each record has five variable fields separated by commas. The fields do not start and end in the same position in every record and have different lengths in different records, **so you could not just specify the starting position and length (p,m) for any of these fields in a BUILD (or FIELDS or OUTREC) or OVERLAY operand of the INREC, OUTREC or OUTFIL statement. But you can use the PARSE operand of an INREC, OUTREC or OUTFIL statement to define rules that tell DFSORT** how to extract the relevant data from each variable input field into a fixed parsed field, and then use the fixed parsed fields in a BUILD or OVERLAY operand as you would use fixed input fields.

You define a parsed field for converting a variable field to a fixed parsed field using a %nn name where nn can be 00 to 99. You can define and use up to 100 parsed fields per run. Each %nn parsed field must be defined only once. A %nn parsed field must be defined in a PARSE operand **before** it is used in a BUILD or OVERLAY operand.

Suppose you wanted to reformat the CSV records to produce these output records:

Wayne -178.5 Gotham

Summers 698.5 Sunnydale

Kent 5.5 Metropolis

Prince 112.5 Gateway

You can use the following OUTREC statement to parse and reformat the variable fields:

OUTREC PARSE=(%01=(ENDBEFR=C',',FIXLEN=8),

%=(ENDBEFR=C','),

%03=(ENDBEFR=C',',FIXLEN=5),

%04=(ENDBEFR=C',',FIXLEN=5),

%05=(FIXLEN=10)),

BUILD=(%01,14:%03,SFF,ADD,%04,SFF,EDIT=(SIIT.T),SIGNS=(,-),

25:%05)

The **PARSE** operand defines how each variable field is to be extracted to a fixed parsed field as follows:

* The **%01** parsed field is used to extract the first variable field into an 8-byte fixed parsed field. ENDBEFR=C',' tells DFSORT to stop extracting data at the byte before the next comma (the comma after the first variable field). FIXLEN=8 tells DFSORT that the %01 parsed field is 8 bytes long. Thus, for the first record, DFSORT extracts Wayne into the 8-byte %01 parsed field. Since Wayne is only 5 characters, but the %01 parsed field is 8 bytes long, DFSORT pads the %01 parsed field on the right with 3 blanks. ENDBEFR=C',' also tells DFSORT to skip over the comma after the first variable field before it parses the second variable field.
* The **%** parsed field is used to skip the second variable field without extracting anything for it. Since we don't want this field in the output record, we can use % to ignore it. Thus, for the first record, we ignore M. ENDBEFR=C',' tells DFSORT to skip over the comma after the second variable field before it parses the third variable field.
* The **%03** parsed field is used to extract the third variable field into a 5-byte fixed parsed field. ENDBEFR=C',' tells DFSORT to stop extracting data before the next comma (the comma after the third variable field). FIXLEN=5 tells DFSORT that the %03 parsed field is 5 bytes long. Thus, for the first record, DFSORT extracts -53 into the 5-byte %03 parsed field. Since -53 is only 3 characters, but the %03 parsed field is 5 bytes long, DFSORT pads the %03 parsed field on the right with 2 blanks. ENDBEFR=C',' also tells DFSORT to skip over the comma after the third variable field before it parses the fourth variable field.
* The **%04** parsed field is used to extract the fourth variable field into a 5-byte fixed parsed field. ENDBEFR=C',' tells DFSORT to stop extracting data before the next comma (the comma after the fourth variable field). FIXLEN= 5 tells DFSORT that the %04 parsed field is 5 bytes long. Thus, for the first record, DFSORT extracts -1732 into the 5-byte %04 parsed field. Since -1732 is 5 characters, it fills up the 5-byte %05 parsed field and padding is not needed. ENDBEFR=C',' also tells DFSORT to skip over the comma after the fourth variable field before it parses the fifth variable field.
* The **%05** parsed field is used to extract the fifth variable field into a 10-byte fixed parsed field. FIXLEN=10 tells DFSORT that the %05 parsed field is 10 bytes long. Thus, for the first record, DFSORT extracts Gotham and 4 blanks into the 10-byte %01 parsed field.

The **BUILD** operand uses the previously extracted fixed parsed fields to build the output record as follows:

* **%01** copies the 8-byte fixed-length data extracted from the first variable field to positions 1-8 of the output record. For the first record, positions 1-8 contain 'Wayne   '.
* **14:%03,SFF,ADD,%04,SFF,EDIT=(SIIT.T),SIGNS=(,-)** adds the 5-byte fixed-length data extracted from the third variable field to the 5-byte fixed-length data extracted from the fourth variable field and places the 6-byte edited result in positions 14-19 of the output record. For the first record, positions 14-19 contain -178.5 (-53 + -1732 = -1785 edited to -178.5). Note that since the %03 and %04 parsed fields may be padded on the right with blanks, we must use the SFF format to handle the sign and digits correctly.
* **25:%05** copies the 10-byte fixed-length data extracted from the fifth variable field to positions 25-34 of the output record. For the first record, positions 25-34 contain 'Gotham    '.

**TIP- To exclude row/rec from infiles**

**JCL sort** can allow you to easily omit or remove rows from your output data. You simply just call the sort job, pass it the files, and tell it the parms on what type of sorting you want and also what data you would like to omit from the output. The omit parameter is optional, but if you include the omit then the data will be removed from your output if it matches your criteria. For example, if you want your JCL to skip some input data, you use the omit parm just as it shows below.   
//SYSIN DD \*  
SORT FIELDS=COPY  
OMIT COND=((10,2,CH,EQ,C'IA'),AND,(67,2,CH,NE,C'IA')) (This means you will bypass rows where position 10, for 2 bytes, equals IA, and also position 67 for 2 bytes does not equal IA). By using OMIT, you are telling your jcl code that if these 2 conditions are true, then the data is bypassed so your output data will "not" include these rows. There is no exclude parm, omit is what you use. This can handle almost any condition for bypassing specifc rows from your output result file.

**ICETOOL**

**Pgm to count no of records in a file**

//UGBICECN JOB AF-1055,'ICETOOL CNT',

// MSGCLASS=X,MSGLEVEL=(1,1),

// CLASS=P,NOTIFY=&SYSUID

//\*

//ICE001 EXEC PGM=ICETOOL

//DFSMSG DD SYSOUT=\*

//TOOLMSG DD SYSOUT=\*

//SYSOUT DD SYSOUT=\*

//SYSPRINT DD SYSOUT=\*

//\*INPUT FILE COUNT

//\*INPUTFL DD DSN=SDAF.NANI.IN.SE,

//IN1 DD DSN=SDAF.NANI.IN.SE,

// DISP=SHR

//TOOLIN DD \*

COUNT FROM(IN1)

/\*

**ICEGENER**

ICEGENER is an integral component of IBM's DFSORTprogram product and can be used as a replacement or substitutefor the IEBGENER program. Most installations use theprogram in a selective way. The program is often entered as aname change in the PGM parameter on the JCL EXEC card inplace of IEBGENER.The ICEGENER program is far more efficient thanIEBGENER and can improve an installation's batch throughputbecause a large number of processes invoke IEBGENER.

MVS users know **IEBGENER is used to copy files**. However, it can also be very slow compared to SYNCSORT. The good news is that SYNCSORT has ICEGENER. The JCL for this is similar to IEBGENER except that it is more efficient.

I tried comparing these two programs by looking at the CPU time for reformatting a large file. The ICEGENER step used on the average, 0.12 CPU seconds while the IEBGENER used on the average, 0.15 CPU seconds. This may not be a huge but if you consider a larger file, then the impact may be noticeable. **The drawback is that if there are SYSIN statements specified to ICEGENER, it calls IEBGENER instead**.

The JCL to execute ICEGENER is similar to IEBGENER:

**//GENER    EXEC PGM=ICEGENER  
//SYSUT1   DD  DSN=YOUR.DATA.SET.NAME,DISP=SHR  
//SYSUT2   DD  DSN=YOUR.NEWDATA.SET.NAME,DISP=(,CATLG),  
//            DCB=\*.SYSUT1  
//SYSPRINT DD SYSOUT=\*  
//SYSIN    DD DUMMY**

**ABENDS**

**S0CB** - Attempting to divide by 0 and not using ON SIZE ERROR

**S002**  - Very large record length/ wrong record length

**Sx22** - Job has been cancelled. The value of x will vary depending on the way the job was

cancelled. S222 means job was cancelled by a user or operator without a dump.

If a TSO session times out you will probably get an S522 abend code.

**S222** - The job was cancelled (by subsystem or operator) because it violated

some restriction

**S522** - JOB or TSO session exceeded maximum job wait time OR

operator did not mount the require tape within allowed time limit

**S806**  - Load module not found

**S837** - Space problem, Alloted space is not enough for data set

**S913** - You are trying to access a dataset which you are not authorized to use.

**SOC7** - 1. Moving non-numeric value to numeric field

2. Not initilizing the numeric variables before first use

**SOC4** - 1. Index exceeds the size of table

2. Trying to use File Section variables without opening the file

**S0C1** - Operation Exception. Check for subscript errors, **missing DD card**, file not opened.

**SE37** - Insufficient disk space.

This occurs when there are no more volumes available.

**SB37 -**The system allocated all the primary and secondary space it could, but it still could not write the output.

**SD37 -**The abend code SD37 is raised when you don't specify a secondary allocation.

**U1026**- COBOL sort failed.

**U1056**- Program didn't close a file before ending