JCL & UTILITIES CHEAT SHEET

Common Statements • Examples • Utility Commands

z/OS | JES2 / JES3 | IBM Mainframe Batch

TABLE OF CONTENTS

SECTION 01

JCL STATEMENT TYPES

Column layout, comment lines, and continuation rules

Fixed-Format Column Layout (default mode)

Cols 1–2	// — identifies a JCL statement (every statement starts here).
Cols 3–10	Name field — job name, step name, or DD name (max 8 chars, A–Z 0–9 @#$, start alpha).
Col 11	Single blank separator between name field and operation.
Cols 12–15	Operation — JOB, EXEC, DD, PROC, PEND, IF, THEN, ELSE, ENDIF, SET, INCLUDE, JCLLIB.
Col 16	Single blank separator between operation and parameters.
Cols 17–71	Parameters field — keyword and positional parameters, comma-separated.
Col 72	Continuation indicator — any non-blank character (usually a digit or X) to continue on next line.
Cols 73–80	Sequence field — optional card number, ignored by JES.

Statement Identification Rules

//*	JCL comment line — ignored by the system, appears in job log.
/*	In-stream delimiter — marks end of in-stream data (SYSIN DD *).
//	Normal JCL statement prefix.
//	Continuation line — cols 4–16 must be blank, parameters start col 16+.
blank	Any line not starting // is treated as in-stream data or ignored.

! NOTE: Continuation: end the line at col 71 (non-blank col 72), then resume on the next line with // in cols 1–2 and first parameter in col 16 or later.

Anatomy of a JCL Job

```
//-- JOB statement: must appear first ---
//BATCHJOB JOB (ACCT01),'MY BATCH',CLASS=A,MSGCLASS=X,
// NOTIFY=&SYSUID,REGION=0M
//-- Optional: search private procedure library ---
//*
// JCLLIB ORDER=(MY.PROCLIB)
//-- STEP 1: run a compiled program ---
```

```
//STEP1 EXEC PGM=MYPGM,PARM='MODE=PROD'
//STEPLIB DD DSN=MY.LOAD.LIBRARY,DISP=SHR
//INPUT DD DSN=MY.INPUT.FILE,DISP=SHR
//OUTPUT DD DSN=MY.OUTPUT.FILE,DISP=(NEW,CATLG,DELETE),
// SPACE=(TRK,(10,5),RLSE),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
```

SECTION 02

JOB STATEMENT

Identifies and controls the entire job

JOB Statement — All Key Parameters

(acct-info),'programmer'	Positional params: accounting info (site-specific) and programmer name (in quotes).
CLASS=A	Job class — routes job to appropriate initiator queue. Classes A–Z and 0–9.
MSGCLASS=X	Output class for JES job log (JCL listing + messages). X usually holds for viewing.
MSGLEVEL=(1,1)	Controls JCL listing (0=none,1=input,2=all) and message level (0=none,1=errors,2=all).
NOTIFY=&SYSUID;	Send TSO message when job ends. &SYSUID; substitutes submitting user's ID.
REGION=0M	Memory region size. 0M = installation maximum. Also per-step with EXEC REGION=.
TIME=(mm,ss)	CPU time limit. CANCEL abend if exceeded. TIME=NOLIMIT removes limit.
PRTY=8	JES input queue priority (0–15, higher = sooner). Not widely used with WLM.
RESTART=stepname	Restart job from a specific step (used after partial failure).
TYPRUN=SCAN	Scan JCL for syntax errors only — no execution.
TYPRUN=HOLD	Submit to JES hold queue — must release manually.
TYPRUN=COPY	Copy JCL to SYSOUT without executing.
COND=(rc,op)	Job-level COND — skip all remaining steps if condition is true.
USER=userid	Run job under a different user ID (requires authorization).
PASSWORD=pw	Password for USER= (or RACF passphrase).

Complete JOB Statement Example

```
//-- Full JOB statement with common parameters ---
//MYJOB01 JOB (DEPT01,PROJ99),
// 'JOHN DOE - ETL JOB',
// CLASS=B,
// MSGCLASS=X,
// MSGLEVEL=(1,1),
```

```
// NOTIFY=&SYSUID,
// REGION=512M,
// TIME=(5,0)
```

>> TIP: Use TYPRUN=SCAN before first production run — it costs no CPU and catches most JCL errors including parameter typos and missing statements.

SECTION 03

EXEC STATEMENT

Defines each step — PGM or PROC, parameters, and conditions

EXEC Statement Parameters

PGM=program-name	Run a load module from JOBLIB, STEPLIB, or system link list.
PROC=proc-name (or just proc-name)	Invoke a catalogued or in-stream PROC.
PARM='string'	Pass a parameter string to the program (max 100 chars; received via PARM field in COBOL/assembler).
COND=(rc,op)	Skip this step if condition is true (see Section 05).
COND=(rc,op,step)	Test return code of a specific previous step.
REGION=256M	Override job-level REGION for this step only.
TIME=(mm,ss)	CPU time limit for this step.
ACCT=(acct-data)	Step-level accounting information.
DYNAMNBR=n	Number of dynamic allocations the step may make.
ADDRSPC=REAL/VIRT	Force real or virtual storage (rarely used; default VIRT).

EXEC Examples

```
//-- Run a program with a PARM ---
//STEP1 EXEC PGM=COBPGM01,PARM='REPORT,PROD'
//-- Invoke a catalogued procedure ---
//STEP2 EXEC PROC=COBUCLG
//-- Run SORT utility ---
//STPSORT EXEC PGM=SORT
//-- Skip step if STEP1 RC > 0 ---
//STEP3 EXEC PGM=NEXTPGM,COND=(0,NE,STEP1)
//-- Invoke proc and override symbolic ---
//STEP4 EXEC MYJCLPRC,ENV=PROD,REGION.STEP1=1024M
```

>> TIP: When calling a PROC, override a step's parameter with parm.stepname='value' syntax — e.g., PARM.COMPILE='LIST'.

SECTION 04

DD STATEMENT

Data Definition — the most parameter-rich JCL statement

DD Statement — Core Parameters

DSN=dataset.name	Dataset name (alias: DSNAME=). Enclose in quotes if it contains special chars.
DISP=(status,norm,abnorm)	Dataset disposition: status=NEW|OLD|SHR|MOD, normal end=KEEP|CATLG|DELETE|PASS, abnormal=KEEP|CATLG|DELETE.
UNIT=SYSDA	Device type for new datasets. SYSDA=any direct access, TAPE=tape.
SPACE=(type,(pri,sec),RLSE)	Allocation: type=TRK|CYL|BLKSIZE, pri=primary qty, sec=secondary qty, RLSE=release unused at close.
DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)	Data Control Block: record format, logical record length, block size. 0=system-determined blocksize.
SYSOUT=*	Route to MSGCLASS spool class. SYSOUT=(A,) routes to class A. SYSOUT=(,PGM) specifies a writer.
*** or DATA**	In-stream data follows. /* ends data stream. DATA allows /* within the data.
DUMMY	Nullifies DD: reads return immediate EOF, writes are discarded.
NULLFILE	z/OS synonym for DUMMY.
VOL=SER=volser	Specify volume serial. VOL=REF=dsname borrows volume from another DD.
LABEL=(n,SL)	Tape label: n=file sequence number, SL=standard, NL=no label.
EXPDT=yyyy/ddd	Expiration date (Julian). Dataset cannot be deleted before this date.
RETPD=nnn	Retention period in days from creation date.
FREE=CLOSE	Release dataset allocation at close rather than step end.
DISP=SHR	Share dataset with other jobs (read-only or VSAM update sharing).

DISP Values Quick Reference

NEW	Dataset does not yet exist; create it.
OLD	Dataset exists; exclusive control (no sharing).
SHR	Dataset exists; shared access with other jobs.
MOD	Extend existing sequential dataset; position to end-of-data.

KEEP	Keep dataset after step/job; do not update catalog.
CATLG	Keep and add entry to system catalog (use for new datasets).
DELETE	Delete dataset and remove catalog entry.
PASS	Pass dataset to the next step in same job (temporary intra-job pass).
UNCATLG	Keep dataset but remove from catalog.

Concatenated DD Statements

```
//-- Concatenate multiple datasets to one DD name ---
//INPUT DD DSN=PROD.DATA.MONTH01,DISP=SHR
// DD DSN=PROD.DATA.MONTH02,DISP=SHR
// DD DSN=PROD.DATA.MONTH03,DISP=SHR
//-- In-stream data ---
//SYSIN DD *
SORT FIELDS=(1,10,CH,A)
/*
//-- Temporary dataset (system-named) ---
//TEMP1 DD DSN=&&TEMPFILE,DISP=(NEW,PASS),
// SPACE=(TRK,(5,2)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
```

*** RULE:** RECFM codes: F=Fixed, V=Variable, U=Undefined; B=Blocked, S=Spanned. FB (Fixed Blocked) is the most common for batch files.

SECTION 05

CONDITION CODES & COND

Return codes, conditional step execution, IF/THEN/ELSE

Standard Return Code Conventions

RC = 0	SUCCESSFUL — step completed without error.
RC = 4	WARNING — completed with minor issues; usually acceptable.
RC = 8	ERROR — step failed; review output before continuing.
RC = 12	SEVERE ERROR — significant failure; subsequent steps usually skipped.
RC = 16	TERMINAL ERROR — catastrophic failure; immediate job termination recommended.
RC = 4095	Maximum user return code.
Sxxx	System abend code (e.g. S0C7=data exception, S322=time exceeded, S806=load error).
Unnnn	User abend code set by program via ABEND macro or COBOL STOP RUN with non-zero RC.

COND Parameter — Skip Logic

COND=(4,LT)	Skip this step if ANY previous RC is Less Than 4 (i.e. run only if some RC >= 4).
COND=(0,NE)	Skip if any previous RC is Not Equal to 0 (run only if all steps RC=0).
COND=(8,LE,STEP1)	Skip if STEP1 RC is Less than or Equal to 8.
COND=(0,EQ,STEP2)	Skip if STEP2 RC equals 0.
COND=EVEN	Run this step even if a previous step abended.
COND=ONLY	Run this step ONLY if a previous step abended (cleanup/notification step).
COND=((4,LT),(8,GT,STEP1))	Multiple conditions — step skipped if ANY is true.

IF/THEN/ELSE/ENDIF Construct (preferred over COND)

```
//-- Modern conditional execution ---
//IF (STEP1.RC = 0) THEN
//STEP2 EXEC PGM=NEXTPGM
//INPUT DD DSN=STEP1.OUTPUT,DISP=SHR
//ELSE
//ERRNOTIF EXEC PGM=ERRPGM
//ENDIF
//-- Check for abend ---
```

```
//IF (STEP1.ABEND) THEN
//CLEANUP EXEC PGM=CLEANPGM
//ENDIF
//-- Multiple step RC check ---
//IF (STEP1.RC LE 4 AND STEP2.RC = 0) THEN
//STEP3 EXEC PGM=FINALPGM
//ENDIF
```

>> TIP: Prefer IF/THEN/ELSE/ENDIF over COND= — it is far more readable and supports AND/OR logic, parentheses, and ABEND checks that COND= cannot express.

! NOTE: COND= logic is inverted: COND=(4,LT) means SKIP if RC < 4 (which feels backwards). IF/THEN/ELSE avoids this confusion.

SECTION 06

PROCEDURES (PROCs)

Reusable JCL — in-stream and catalogued procedures

In-Stream PROC

```
//-- Define in-stream procedure ---
//MYPROC PROC ENV=TEST
//STEP1 EXEC PGM=COBPGM,PARM='&ENV'
//STEPLIB DD DSN=MY.&ENV..LOADLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
// PEND
//-- Invoke and override symbolic parameter ---
//S1 EXEC MYPROC,ENV=PROD
```

JCLLIB — Private Procedure Library Search

```
//-- Search MY.PROCLIB before system PROCLIB ---
// JCLLIB ORDER=(MY.PROCLIB,SYS1.PROCLIB)
```

Overriding PROC Parameters

PARM.stepname='new'	Override PARM for a specific step in the PROC.
DD.stepname.ddname override	Override a DD in a PROC step: //ddname DD DSN=... after EXEC procname.
REGION.stepname=512M	Override REGION for a PROC step.
TIME.stepname=(2,0)	Override TIME for a PROC step.
sym=value on EXEC	Pass symbolic parameter value: EXEC MYPROC,ENV=PROD.

>> TIP: Symbolic parameters (&&SYMB;) in PROCs allow one PROC to serve multiple environments (TEST/PROD) simply by changing a single EXEC parameter.

X WARN: A catalogued PROC cannot contain a JOB statement. It must start with //procname PROC and end with // PEND (or just // at end-of-member).

SECTION 07

SPECIAL DD NAMES

Reserved system DD names and their purposes

JOBLIB	Job-level load library search — searched before system link list for ALL steps. Defined after JOB, before first EXEC.
STEPLIB	Step-level load library search — overrides JOBLIB for that step only.
SYSPRINT	Most utilities write messages here; route to SYSOUT=* for spool.
SYSIN	Control statement input for utilities (IEBGENER, SORT, IDCAMS, etc.).
SYSOUT	IEBGENER output DD name (not the system spool keyword).
SYSUT1	Input dataset for IEBGENER, IEBPTPCH (utility convention).
SYSUT2	Output dataset for IEBGENER, IEBPTPCH (utility convention).
SYSUDUMP	Formatted storage dump on abend — most useful for program debugging.
SYSABEND	Full system dump on abend — very large; use only when SYSUDUMP insufficient.
SYSMDUMP	Machine-readable dump for IPCS analysis (component debugging).
CEEDUMP	Language Environment dump — for LE-conforming programs (COBOL, PL/I, C).
SYSCHK	Checkpoint dataset used with RESTART.
IEFRDER	SMF recording DD used by some IBM utilities.
JESYSMSG	JES system messages (informational, usually not coded explicitly).

! NOTE: Omitting SYSUDUMP/SYSABEND means no dump is produced on abend — always include at least SYSUDUMP DD SYSOUT=* in production steps.

>> TIP: STEPLIB takes precedence over JOBLIB which takes precedence over system link list (LPA / LNKLST). Use STEPLIB when different steps need different load libraries.

SECTION 08

IEBGENER

Copy and reformat sequential datasets — the workhorse utility

Required DD Statements

SYSUT1	Input dataset — the source file to be copied.
SYSUT2	Output dataset — the target file. Must match or be compatible with SYSUT1 DCB.
SYSIN	Control statements (use DUMMY if no reformatting needed — pure copy).
SYSPRINT	Messages and statistics output — route to SYSOUT=*.

Example 1: Simple File Copy

```
//COPY EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT1 DD DSN=PROD.INPUT.FILE,DISP=SHR
//SYSUT2 DD DSN=BACKUP.OUTPUT.FILE,
// DISP=(NEW,CATLG,DELETE),
// SPACE=(TRK,(20,5),RLSE),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
```

Example 2: Copy PDS Member

```
//CPYMEM EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT1 DD DSN=MY.PDS(MEMBER1),DISP=SHR
//SYSUT2 DD DSN=ANOTHER.PDS(MEMBER1),DISP=SHR
```

IEBGENER Control Statements (SYSIN)

GENERATE MAXFLDS=n	Header — required when using RECORD statements. MAXFLDS = total field operations.
RECORD FIELD=(len,in,conv,out)	Reformat: extract LEN bytes from IN position, convert, place at OUT position.
LABELS LABELS=(NO)	Suppress or handle labels on tape input.

>> TIP: For a straight copy with no reformatting, always use SYSIN DD DUMMY — it avoids the overhead of reading control statements and is slightly faster.

! NOTE: IEBGENER processes one record at a time; for very large datasets with complex transformations, SORT/ICETOOL or a custom program is more efficient.

SECTION 09

IEBCOPY

Copy, compress, and manage PDS and PDSE datasets

Required DD Statements

SYSPRINT	Messages and member list — route to SYSOUT=*.
SYSIN	COPY/SELECT/EXCLUDE control statements.
SYSUT3	Spill dataset for IEBCOPY work area (TRK,(10,10) usually sufficient).
SYSUT4	Second spill dataset (same size as SYSUT3).
inddname	User-named DD for input PDS — referenced by INDD= in control statements.
outddname	User-named DD for output PDS — referenced by OUTDD= in control statements.

Example 1: Compress a PDS In-Place

```
//-- Compress PDS (reclaim space from deleted members) ---
//COMPRESS EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSUT3 DD SPACE=(TRK,(10,10))
//SYSUT4 DD SPACE=(TRK,(10,10))
//MYPDS DD DSN=MY.PARTITIONED.DS,DISP=OLD
//SYSIN DD *
COPY OUTDD=MYPDS,INDD=MYPDS
/*
```

Example 2: Copy Selected Members

```
//SELCOPY EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSUT3 DD SPACE=(TRK,(5,5))
//SYSUT4 DD SPACE=(TRK,(5,5))
//INPDS DD DSN=SOURCE.PDS,DISP=SHR
//OUTPDS DD DSN=TARGET.PDS,DISP=SHR
//SYSIN DD *
COPY OUTDD=OUTPDS,INDD=INPDS
SELECT MEMBER=PROG001
SELECT MEMBER=PROG002
SELECT MEMBER=((PROG003,PROG003NEW))
/*
```

IEBCOPY Control Statements

COPY OUTDD=out,INDD=in	Copy all members from in to out. Multiple INDD: INDD=((in1,R),(in2)).
SELECT MEMBER=name	Copy only the named member.
SELECT MEMBER=((old,new))	Copy and rename member.
EXCLUDE MEMBER=name	Copy all members EXCEPT the named one.
COPY OUTDD=out,INDD=((in,R))	Replace — overwrite existing members in output PDS.

>> TIP: Always compress a PDS regularly — IBM says 10–15% directory fullness causes performance degradation. A PDS that shows MEMBERS=xx and FREE=00 needs compression.

SECTION 10

IEFBR14

The do-nothing program — allocate and delete via DISP

IEFBR14 contains exactly **one instruction**: Branch Register 14 (return to caller). It does nothing itself — all work is done by JES through the DD statement DISP parameter at step end.

Example 1: Allocate a New Dataset

```
//ALLOC EXEC PGM=IEFBR14
//NEWFILE DD DSN=MY.NEW.DATASET,
// DISP=(NEW,CATLG,DELETE),
// SPACE=(CYL,(5,2),RLSE),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=0),
// UNIT=SYSDA
```

Example 2: Delete a Catalogued Dataset

```
//DELETE EXEC PGM=IEFBR14
//DELFILE DD DSN=MY.OLD.DATASET,
// DISP=(OLD,DELETE,DELETE)
```

Example 3: Allocate Several Datasets in One Step

```
//SETUP EXEC PGM=IEFBR14
//FILE1 DD DSN=WORK.FILE.ONE,DISP=(NEW,CATLG,DELETE),
// SPACE=(TRK,(10,5)),DCB=(RECFM=FB,LRECL=100)
//FILE2 DD DSN=WORK.FILE.TWO,DISP=(NEW,CATLG,DELETE),
// SPACE=(TRK,(5,2)),DCB=(RECFM=VB,LRECL=256)
//FILE3 DD DSN=WORK.FILE.THREE,DISP=(MOD,DELETE)
```

>> TIP: Use IEFBR14 to pre-allocate output datasets before a job stream starts, then reference them with DISP=SHR in subsequent steps.

! NOTE: IEFBR14 always returns RC=0. It cannot fail programmatically — any errors are JES allocation failures (DISP processing).

SECTION 11

SORT / ICETOOL

IBM DFSORT — sort, merge, copy, select, summarise, transform

SORT JCL DD Names

SORTIN	Primary input dataset.
SORTOUT	Primary output dataset.
SYSIN	SORT control statements.
SYSPRINT	Messages and statistics.
SORTIN01–SORTINnn	Additional input files for MERGE operation.
SORTWKnn	Work datasets (SORTWK01–SORTWK09) — DFSORT usually manages these dynamically.

Core SORT Control Statements

```
SORT FIELDS=(start,len,format,order)
* format: CH=char, ZD=zoned dec, PD=packed, BI=binary, FI=fixed
* order: A=ascending, D=descending

//-- Sort by cols 1-10 ascending, then cols 11-15 descending ---
SORT FIELDS=(1,10,CH,A,11,5,ZD,D)

//-- Copy without sort (faster than SORT FIELDS=COPY) ---
SORT FIELDS=COPY

//-- Merge already-sorted files ---
MERGE FIELDS=(1,10,CH,A)

//-- Include only records where col 5 = 'PROD' ---
INCLUDE COND=(5,4,CH,EQ,C'PROD')

//-- Omit records where col 1 = 'D' ---
OMIT COND=(1,1,CH,EQ,C'D')

//-- Sum numeric field, remove duplicates on key ---
SUM FIELDS=(51,10,ZD)

//-- Reformat output: extract fields, add literals ---
OUTREC FIELDS=(1,10,C'|',21,5,C'|',X)

//-- Keep only unique keys (remove duplicates) ---
SORT FIELDS=(1,10,CH,A)
SUM FIELDS=NONE
```

ICETOOL Operators

COPY	Copy records — with optional INCLUDE/OMIT.
SORT	Sort and output to one or more datasets.
MERGE	Merge pre-sorted files.
SELECT	Select first/last/unique/duplicate occurrences.
COUNT	Count records and write count to a dataset.
STATS	Compute min, max, average, total for numeric fields.
OCCURS	Count occurrences of each unique key value.
DISPLAY	Print fields in formatted report.
RANGE	List records within a value range.
MODE	Find most-frequently-occurring value.

>> TIP: DFSORT's OPTION COPY is faster than IEBGENER for large files because DFSORT uses larger internal buffers and I/O optimisation.

SECTION 12

IDCAMS

Access Method Services — VSAM and catalog management

IDCAMS JCL Skeleton

```
//IDCAMS EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
commands go here
/*
```

DEFINE CLUSTER — Create VSAM KSDS

```
DEFINE CLUSTER -
(NAME(MY.VSAM.KSDS) -
CYLINDERS(5 1) -
RECSZ(100 200) -
KEYS(10 0) -
INDEXED -
SHAREOPTIONS(2 3)) -
DATA -
(NAME(MY.VSAM.KSDS.DATA)) -
INDEX -
(NAME(MY.VSAM.KSDS.INDEX))
```

Common IDCAMS Commands

DEFINE CLUSTER (...) DATA (...) INDEX (...)	Create a VSAM cluster. Use INDEXED for KSDS, NONINDEXED for ESDS, NUMBERED for RRDS.
DELETE cluster-name CLUSTER	Delete a VSAM cluster and all components. Add PURGE to delete before expiry date.
REPRO INFILE(in) OUTFILE(out)	Copy records between VSAM files or between VSAM and sequential. Also used to load initial data.
REPRO INFILE(in) OUTFILE(out) COUNT(100)	Copy only the first 100 records.
REPRO INFILE(in) OUTFILE(out) SKIP(50) COUNT(10)	Skip 50 records then copy 10.
LISTCAT ENTRIES(name) ALL	Display all catalog information for a dataset (statistics, components, attributes).
LISTCAT LEVEL(MY.DATA) ALL	List all datasets whose name starts with MY.DATA.

PRINT INFILE(ddname) COUNT(25)	Print records in CHARACTER, HEX, or DUMP format.
PRINT INFILE(ddname) CHARACTER SKIP(0) COUNT(10)	Print first 10 records in character format.
ALTER cluster NEWNAME(newname)	Rename a VSAM cluster in the catalog.
VERIFY FILE(ddname)	Reset VSAM end-of-file marker after crash; always run before opening a damaged VSAM.
EXPORT cluster OUTFILE(dd) TEMPORARY	Export VSAM cluster for backup. TEMPORARY = cluster remains usable.
IMPORT INFILE(dd) OUTDATASET(name)	Import a previously exported VSAM cluster.
DEFINE ALIAS (NAME(alias) RELATE(cluster))	Create a catalog alias so a short name points to a fully-qualified cluster name.
DELETE alias ALIAS	Remove an alias from the catalog.
IF LASTCC = 0 THEN …	IDCAMS conditional: test return code of last command.
SET MAXCC = 0	Reset maximum condition code (suppress non-zero RC from preceding command).

>> TIP: SET MAXCC=0 after a DELETE command suppresses RC=8 when the dataset doesn't exist — useful in housekeeping JCL that runs unconditionally.

! NOTE: SHAREOPTIONS(2 3): first value controls sharing within a system, second across systems. (2 3) is the most common production setting.

SECTION 13

IEBPTPCH

Print and punch sequential datasets and PDS members

IEBPTPCH DD Names

SYSUT1	Input dataset (sequential or PDS).
SYSUT2	Output — printer DD (SYSOUT=*) or card punch.
SYSIN	PRINT/PUNCH control statements.
SYSPRINT	Messages output.

Example: Print First 50 Records

```
//PRINT EXEC PGM=IEBPTPCH
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DSN=MY.INPUT.FILE,DISP=SHR
//SYSUT2 DD SYSOUT=*
//SYSIN DD *
PRINT MAXFLDS=1,STOPAFT=50
RECORD FIELD=(80,1,,1)
/*
```

IEBPTPCH Control Statements

PRINT MAXFLDS=n	Print operation header. MAXFLDS = number of FIELD parameters that follow.
PUNCH MAXFLDS=n	Punch operation header (output to card image file).
STOPAFT=n	Stop after printing n records.
SKIP=n	Skip first n records before printing.
TYPORG=PO	Input is a PDS (Partitioned Organization). Default is PS (sequential).
RECORD FIELD=(len,in,conv,out)	Select and reformat fields: length, input position, conversion, output position.
TITLE ITEM=('heading',col)	Print a title line at the specified column.

>> TIP: For a quick formatted hex dump of a file, use PRINT MAXFLDS=1 with RECORD FIELD=(80,1,,1) — this prints all 80 bytes starting at column 1.

SECTION 14

IKJEFT01 — TSO BATCH

Run TSO commands, CLISTs, and REXXs in batch mode

IKJEFT01 DD Names

SYSTSPRT	Output from TSO commands and REXX say/trace — route to SYSOUT=*.
SYSTSIN	Input — TSO commands or CLIST/REXX to execute.
SYSEXEC	REXX exec library (PDS containing REXX EXECs).
SYSPROC	CLIST library (PDS containing CLISTs).
SYSPRINT	Some TSO commands write here instead of SYSTSPRT.

Example 1: Run a REXX Exec

```
//TSOSTEP EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSEXEC DD DSN=MY.REXX.LIBRARY,DISP=SHR
//SYSTSIN DD *
EX 'MY.REXX.LIBRARY(MYREX)' 'PARM1 PARM2'
/*
```

Example 2: Run DB2 Commands via DSN

```
//DB2BATCH EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(DB2P)
RUN PROGRAM(COBDB2PG) PLAN(MYPLAN) -
LIB('MY.LOAD.LIBRARY')
END
/*
```

Example 3: Submit Another Job from Batch

```
//SUBMIT EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//MYJCL DD DSN=MY.JCL.LIBRARY(NEXTJOB),DISP=SHR
//SYSTSIN DD *
SUBMIT 'MY.JCL.LIBRARY(NEXTJOB)'
/*
```

>> TIP: Use IKJEFT01 to chain jobs: submit the next job in a sequence from within the current job, passing dynamic parameters via REXX.

! NOTE: IKJEFT01 RC=0 does NOT mean the REXX ran successfully — check SYSTSPRT output and trap REXX errors explicitly with 'SIGNAL ON ERROR'.

SECTION 15

JCL SYMBOLS & GDGs

Symbolic parameters, generation data groups, and handy patterns

JCL Symbolic Parameters

&SYMBOL;	Reference a symbolic parameter (defined in PROC or SET).
&&SYMBOL;	Define a symbolic (in PROC header) — resolves to &SYMBOL; at invocation.
SET SYMBOL=value	Set or override a symbolic at job level (JES2 SET statement).
&SYSUID;	System symbol — submitting user's TSO ID.
&SYSDATE;	System symbol — current date (MM/DD/YY).
&SYSTIME;	System symbol — current time (HH.MM).
&SYSJOBNAME;	System symbol — current job name.
&SYSJE2;	True if running under JES2 (for conditional JCL).

Generation Data Groups (GDGs)

DSN=MY.GDG(0)	Current (latest) generation — used for INPUT (read existing gen).
DSN=MY.GDG(+1)	Next new generation — used for OUTPUT (creates new gen).
DSN=MY.GDG(-1)	Previous generation (read-only).
DSN=MY.GDG(-2)	Two generations back.
DEFINE GDG	IDCAMS command to create the GDG base entry in the catalog.
LIMIT(n)	Maximum number of generations to retain (1–255).
SCRATCH	Automatically delete oldest generation when LIMIT is reached.
NOEMPTY	Do not delete all generations when new one is added past LIMIT (default).
EMPTY	Uncatalog all generations when LIMIT is reached (not just oldest).

IDCAMS: Define a GDG Base

```
//DEFGDG EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DEFINE GDG -
(NAME(MY.DAILY.REPORT) -
LIMIT(30) -
```

```
  NOEMPTY -
  SCRATCH)
  /*
```

Useful JCL Patterns

Restart from step	Add RESTART=STEP3 on JOB statement; use DISP=SHR for already-created datasets.
Dynamic step skip	IF (STEP1.RC > 4) THEN / EXEC PGM=NOTIFYPGM / ENDIF
Temporary datasets	DSN=&&TEMPNAME; — system-named temp; deleted at job end or PASS to next step.
Refer-back volume	VOL=REF=*.STEP1.DDNAME — use same volume as earlier DD.
Refer-back DCB	DCB=*.STEP1.DDNAME — inherit DCB attributes from another DD.
Multi-volume tape	VOLUME=(PRIVATE,RETAIN,,,3) — reserve 3 tape volumes.
Job-level notify on abend	Use COND=ONLY step with PGM=IEBGENER to write notification record.
JCLLIB for private PROCs	// JCLLIB ORDER=(MY.PROCLIB) — place immediately after JOB statement.

>> TIP: GDG relative generation references (+1, 0, -1) are resolved at job OPEN time. Within a single job, (+1) is always the brand-new generation being created.

*** RULE:** Never code DISP=(NEW,KEEP) for a GDG generation — always CATLG so the new generation is registered and the old ones can be aged off by GDG base LIMIT.

END OF JCL & UTILITIES CHEAT SHEET

JOB • EXEC • DD • COND • PROC • IEBGENER • IEBCOPY • SORT • IDCAMS