TATA CONSULTANCY SERVICES

TATA

OS/390 & z/OS - JCL

Experience certainty. IT Services
Business Solutions
Outsourcing

Course Objectives

- To understand the
  - Job Control Language
    - Introduction
    - Overview
  - JOB, EXEC and DD (Data Definition) Statements
    - Syntax of the various statements
    - Execution process

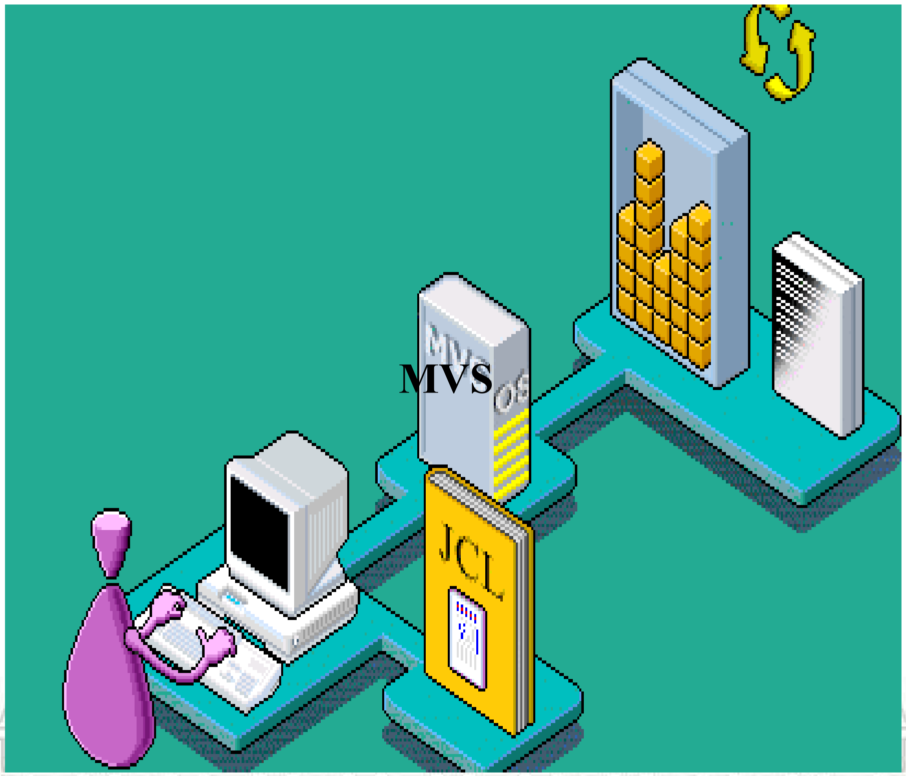
Sessions Outline

- Introduction to JCL
- JOB Statement
- EXEC Statement
- DD Statement

Introduction to JCL

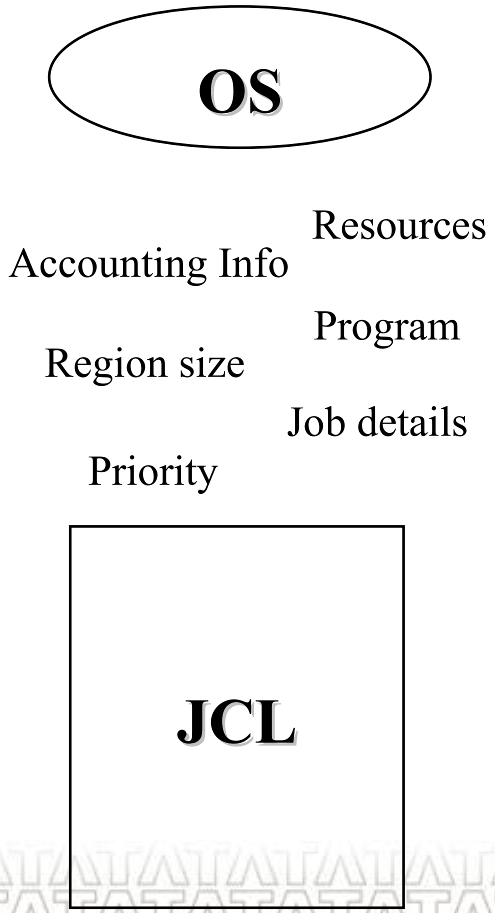
Job Control Language (JCL)

- Describes to the operating system the work that has to be done and the resources required to do the work



TATA CONSULTANCY SERVICES

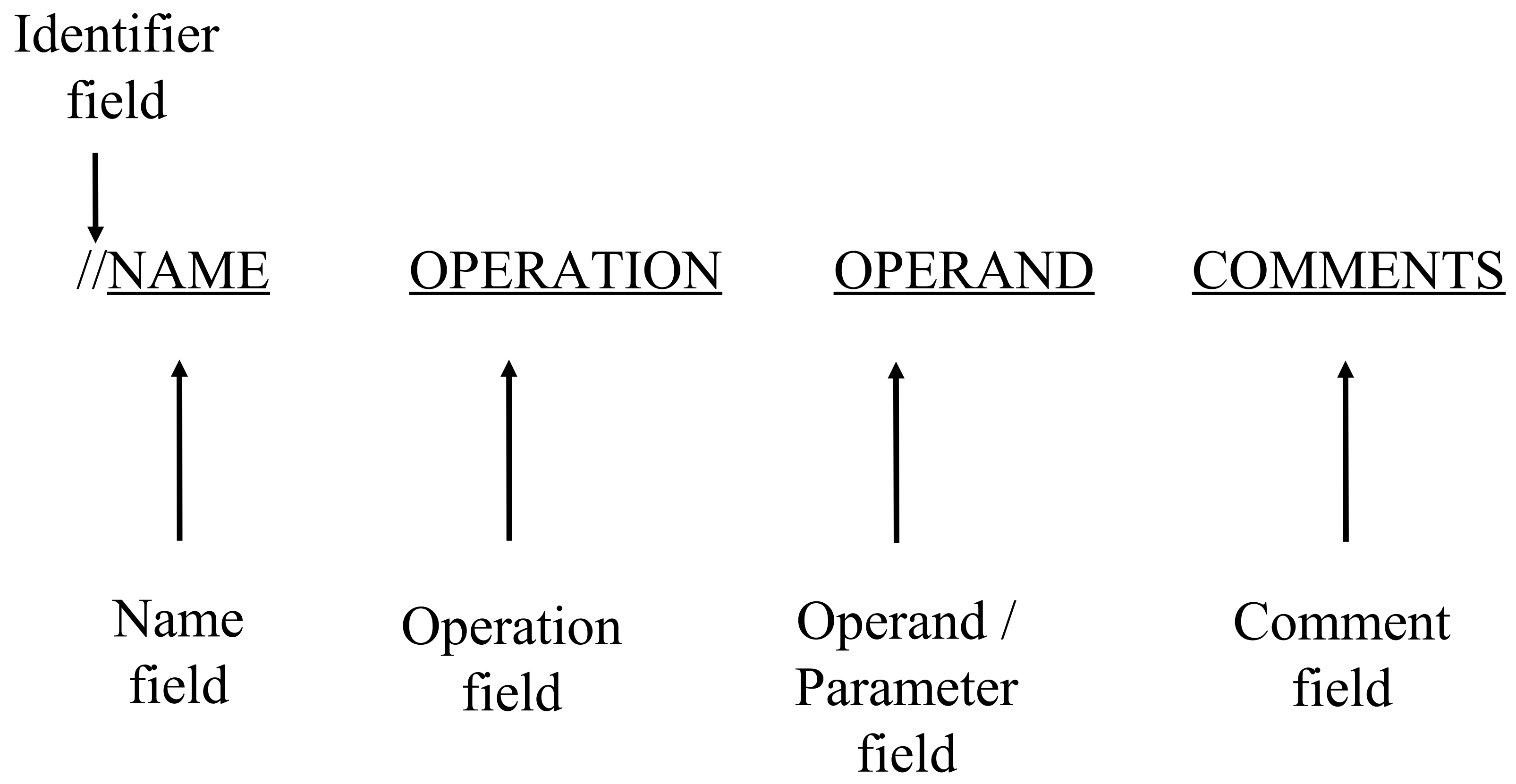
Need for JCL


OS
Resources
Accounting Info
Program
Region size
Job details
Priority
JCL

JCL Features

- Consists of a set of statements called as Job Control Statements
- Group of related JCL statements is known as Job
- JCL consists of one or more Jobs
- Job consists of Job steps to execute the instructions (tasks)

JCL Statement Syntax



Identifier Field

- Identifier field indicates to the system that a statement is a JCL rather than data
- Code the Identifier field beginning in column 1
- Consists of
  - Column 1 and 2 of all the JCL statements (//)
  - Column 1 2 and 3 of a JCL statement (//* for comment)
  - Column 1 and 2 to mark end of data (/*)

Name field

- Identifies a particular statement so that the other statements and the system can refer it
- Syntax
  - name should start in column 3
  - name is 1 through 8 alphanumeric
  - name should be followed by a blank
  - first character should be a alphabetic or national ($, #, @)

Operation Field

- Specifies the type of the statement
  - Consists of characters in the syntax box for the statement
  - Follows the name field
  - Operation must be followed and preceded by at least one blank
  - Must begin on or before column 16

Parameter or Operand Field

- Also called as Operand field
- Must end before column 62
- Contains parameters separated by commas
    - Positional and Keyword Parameters. An eg.

```
//RT452216 JOB 45992,CLASS=A
```

- All positional parameters must precede all keyword parameters
- Sub-parameters may be coded under Parameters to add further meaning to the statement An example.

```
//RT452216 JOB (45992,100,40),COND=(9,LT)
```

Comment Field

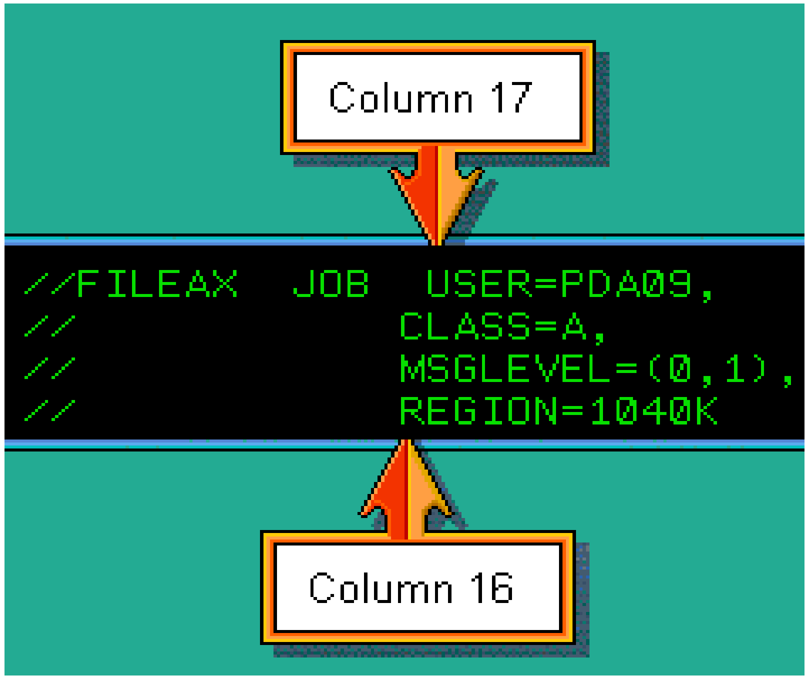
- Used to enter a comment in the output listing
- Used to document a job and its resource requirements
- Can be placed anywhere after the job statement
- Appears as //* in Column 1, 2 and 3
- Comment field can be coded till Column 80

JCL Statements - Basics

- Code the Operation, Operand and Comments fields in free form with at least one blank in-between
- Code the Name field immediately after the Identifier without any blank column
- All fields except for Operand must be separated by one blank
- Only Columns 1 to 71 are used for JCL Content
- Is case sensitive (Lower case not permitted)

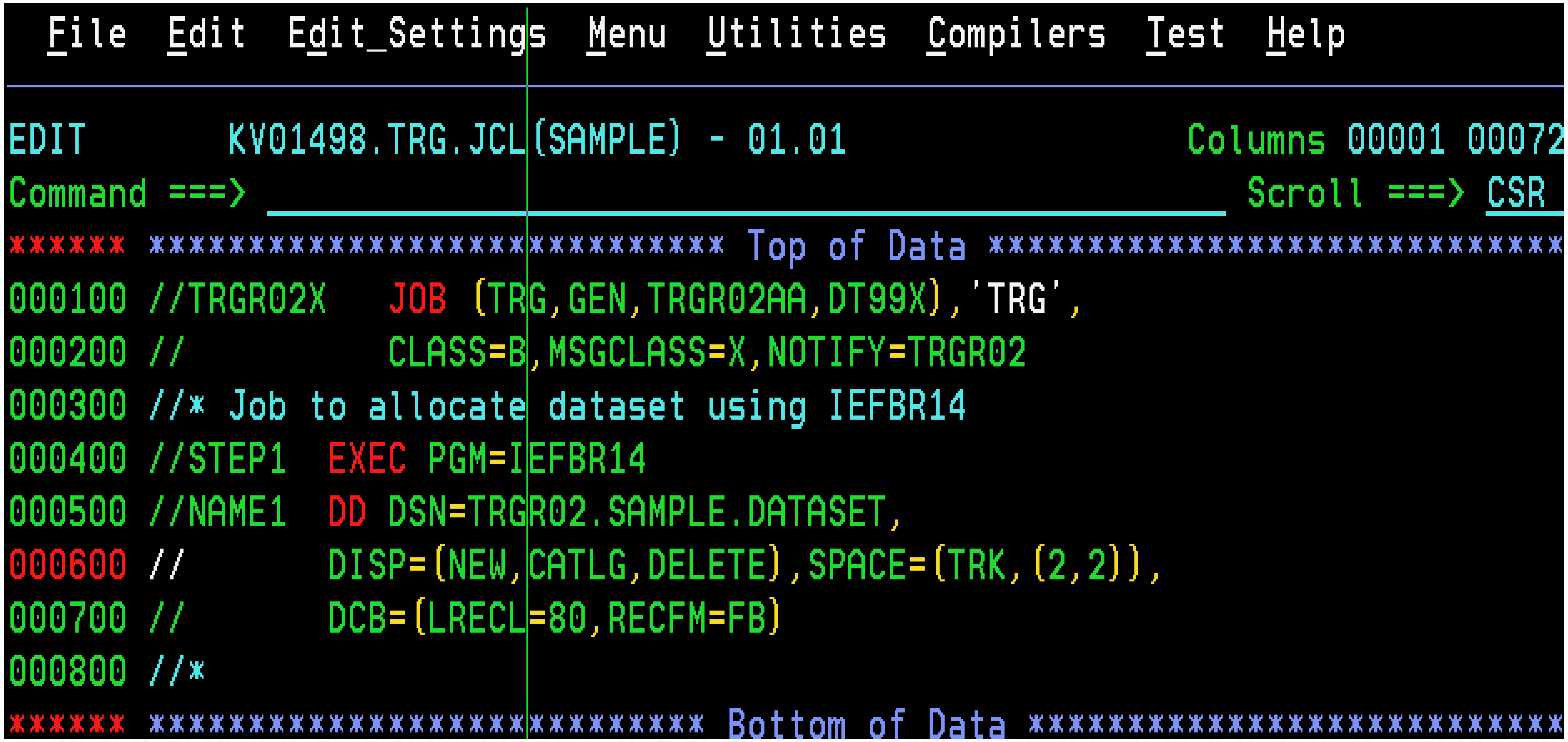
JCL Statements – Basics (Contd.)

- Statements can be continued in the next line with // in 1 & 2 columns
- The continued line must start between columns 4 and 16
- Marking the end of the JCL is optional



```
//FILEAX   JOB   USER=PDA09,
//              CLASS=A,
//              MSGLEVEL=(0,1),
//              REGION=1040K
```

JCL Example



```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help

EDIT       KV01498.TRG.JCL(SAMPLE) - 01.01                 Columns 00001 00072
Command ===>                                                  Scroll ===> CSR
****** ***************************** Top of Data ******************************
000100 //TRGR02X   JOB (TRG,GEN,TRGR02AA,DT99X),'TRG',
000200 //             CLASS=B,MSGCLASS=X,NOTIFY=TRGR02
000300 //* Job to allocate dataset using IEFBR14
000400 //STEP1  EXEC PGM=IEFBR14
000500 //NAME1  DD DSN=TRGR02.SAMPLE.DATASET,
000600 //          DISP=(NEW,CATLG,DELETE),SPACE=(TRK,(2,2)),
000700 //          DCB=(LRECL=80,RECFM=FB)
000800 //*
****** **************************** Bottom of Data ****************************
```

TATA CONSULTANCY SERVICES

JCL Statements - Essential

- Each Job is identified by a JOB statement that marks the beginning of a Job
  - Every job has one and only one job statement
- Each Exec step is identified by an EXEC statement
  - EXEC (Execute) Statements follow the JOB Statement and has the name of the Programs / PROC's to execute

JCL Statements – Essential (Contd.)

- Each Exec step may contain one or more control statements (DD statements) to describe resources required for the Job step
  - DD Statements describe each data set & request the allocation of I/O devices
- Code the comment (//*) Statement to document the JCL
- Code the delimiter (/*) Statement for marking the end-of-data for in-stream data

JCL Statements - Optional

- IF / THEN / ELSE / ENDIF Statement for selective execution of a Job Step
- SET Statement assigns symbolic parameter values
- INCLUDE Statement to copy JCL from a file into Job Stream
- PROC & PEND Statements mark the beginning and end of a Procedure (Cataloged / in-stream)
- JCLLIB Statement names the PROC & JCL Data set

JOB Statement

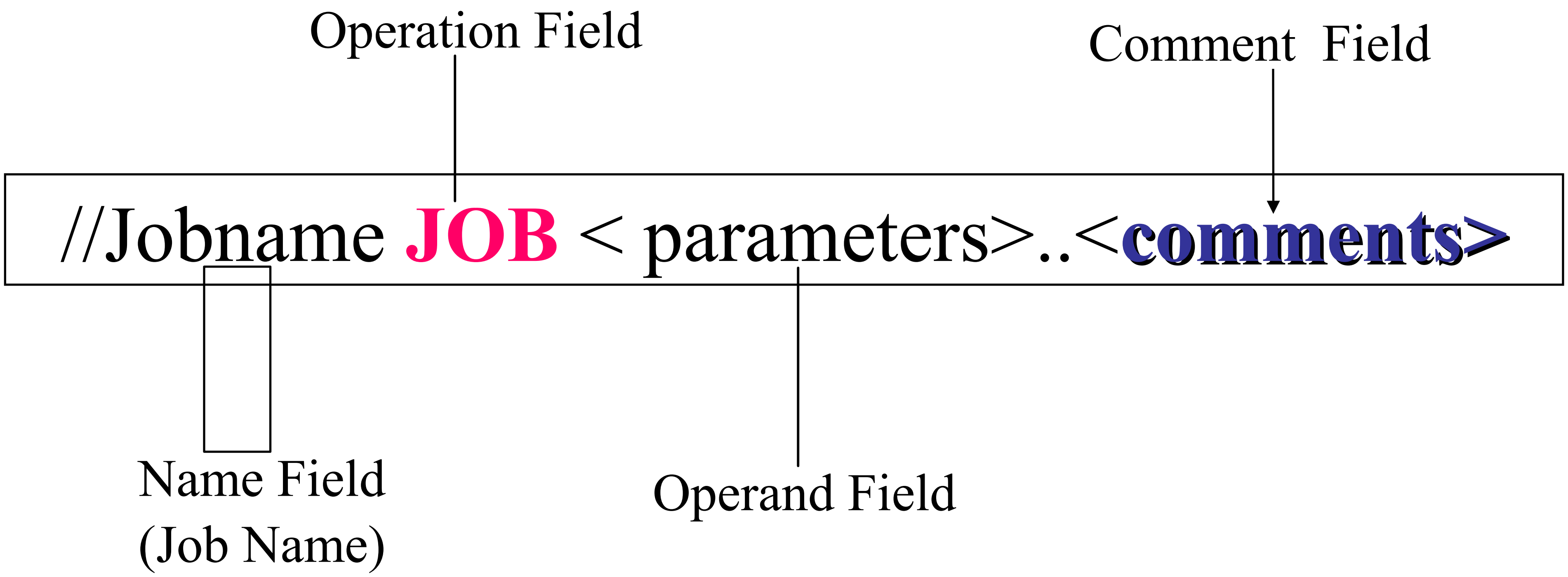
Session Coverage

- Purpose of JOB Statement
- Coding Syntax of JOB statements
- Positional and Keyword parameters in a Job Statement
- Examples

Purpose of JOB Statement

- Identifies a job to the OS using a job name
- Indicates which user is responsible for the job
- Tells the system how to process the job

JOB Statement - Syntax



```
//Jobname JOB < parameters>..<comments>
```

JOB Statement - Name field

//**TRGR02Y** JOB ....

Job name is **TRGR02Y**

Job name is **@TEST**

//**@TEST** JOB ....

JOB Statement - Operation field

```
//MYJOB   JOB ...

//JOB1    JOB ...
```

JOB in Operation Field specifies it as Job statement

Parameter Field

//JOBNAME JOB <Parameters>

The following list shows the parameters on Job statement

Positional Parameters

- ACCT-PARAMETER
- PROGRAMMER NAME

Keyword Parameters

CLASS	COND
PRTY	NOTIFY
MSGCLASS	RESTART
MSGLEVEL	ADDRSPC
REGION	TIME

Positional Parameters

- Job Statement can contain two positional parameters
- First parameter should contain Accounting Information and the second parameter should contain the Programmer's name
- These two parameters are optional, but installation may define them as mandatory.

Positional Parameters - Accounting Information

//JOB1 JOB <u>(TCS,TRG,3,DB2,123) </u>...........

Accounting information

//JOB2 JOB <u>(TCS,DHC,1,DTX99,TRGG01) </u>......

Accounting Information- At TCS

- Five Fields - all are mandatory
  - Field1 - Group or Project
  - Field2 - Sub-Group or module or Project
  - Field3 - User-id
  - Field4 - Location (here type D2 or D1)
  - Field5 - Type of Job (Give DT99X)
- Example for Training:

(TRG,TRG,*TRGG01*,D2,DT99X)

Positional Parameters - Programmer’s Name

//JOB1 JOB (TCS,TRG,3,DB2,123),<u>TRAINING,</u>......

Accounting parameter

2nd positional parameter- programmer's name

//JOB2 JOB (TCS,DH,1,DTX99,TRGR01),<u>’TCS TRG’</u>.

Programmer's name parameter in quotes

//JOB3 JOB ,<u>’Shane o’’Hara’</u>.....

No accounting parameter

Enclose programmer‘s name within apostrophe if it contains special character

Key Word Parameters

The Job Statement (or Job Card) can contain the following Keyword parameters

- CLASS
- TIME
- REGION
- MSGLEVEL
- COND
- TYPRUN
- PRTY
- ADDRSPC
- MSGCLASS
- NOTIFY
- RESTART

Keyword Parameters - CLASS

- Syntax:

  CLASS=<*job class*>

- Examples:

JOB1 IS ASSIGNED TO ***CLASS* A**

```
//JOB1 JOB ACCT1,CLASS=A..
```

JOB2 IS ASSIGNED TO ***CLASS* B**

```
//JOB2 JOB ACCT1,CLASS=B....
```

JOB3 IS ASSIGNED TO ***CLASS*** C

```
//JOB3 JOB ACCT1,CLASS=C....
```

JOB4 IS ASSIGNED TO ***CLASS* D**

```
//JOB4 JOB ACCT1,CLASS=D....
```

Class Definitions at TCS Installation

- Class A -- 2 sec
- Class B -- 7 sec
- Class C -- 19 sec
- Class D -- 46 sec
- Class E -- 5 mins
- Class H -- Held Class
- Class T -- 5 mins (for tapes only)
- Class R -- 5 mins. (cartridges only)

❖ Time specified are CPU time

Keyword Parameters - TIME

- Syntax:

  TIME=([*minutes*][,*seconds*])
- Examples:

  //JOB1 JOB ACCT1,CLASS=E,TIME=(<u>2,30</u>)....

  Allows this job to use 2 min 30 sec of CPU time

  //JOB2 JOB ACCT1,CLASS=E,TIME=(<u>,30</u>)...

  Allows this job to use 30 sec of CPU time

  //JOB3 JOB ACCT1,CLASS=E,TIME=<u>2</u>....

  Allows this job to use 2 mins of CPU time

Keyword Parameters – TIME (Contd.)

- Syntax:

  TIME=([1440]| [NOLIMIT] | [MAXIMUM])
- Examples:

  //JOB4 JOB ACCT1,CLASS=E,TIME=1440....

  These jobs can use the processor for unlimited amount of time

  //JOB5 JOB ACCT1,CLASS=E,TIME=NOLIMIT....

  This job can run for maximum amount of time that is 357912 minutes.

  //JOB6 JOB ACCT1,CLASS=E,TIME=MAXIMUM....

Keyword Parameters - PRTY

- To assign the selection priority to the JOB
- Syntax:

  PRTY=(number)

- Examples:

  //JOB4 JOB ACCT1,CLASS=E,PRTY=<u>12</u>...

Keyword Parameters - REGION

- Syntax:

  REGION={*valueK* | *valueM*}

- Examples:

  //JOB1 JOB ACT1,CLASS=A,ADDRSPC=REAL,REGION=<u>200K</u>.

  Specifies <u>200 kilo bytes</u> of ***central storage*** is required for this job(JOB1)

  //JOB2 JOB ACT1,'TRG',CLASS=A,REGION=<u>20M</u>.

  Specifies <u>20 mega bytes</u> of ***virtual storage*** is required for this job(JOB2)

Keyword Parameters - ADDRSPC

Syntax:

ADDRSPC={*REAL* | *VIRT*}

- Examples:

//JOB1 JOB ACT1,CLASS=A,ADDRSPC=REAL,REGION=20K

Non pageable central storage of 20k
is required for this job(JOB1)

//JOB1 JOB ACT1,CLASS=A,ADDRSPC=VIRT,REGION=20K

Pageable virtual storage of 20k
is required for this job(JOB1)

Keyword Parameters - MSGCLASS

- Syntax:

  MSGCLASS=<*class*>

- Examples:

```
//JOB1 JOB (TCS,TRG,3,,123),’TRG GROUP’,
//         CLASS=B,MSGCLASS=X...
//JOB2 JOB (TCS,TRG,3,,123),’TRG GROUP’,
//         CLASS=B,MSGCLASS=A...
```

Keyword Parameters - MSGLEVEL

- Details in the listing of Joblog can be controlled, using MSGLEVEL parameter
- Syntax:

  MSGLEVEL=([statements][,messages])
- Job log contains the following information :
  - JCL Statements
  - Procedure Statements for any procedure job step calls
  - Messages about Job control statements, allocation of devices and volumes, execution and termination of job steps and disposition of datasets

Keyword Parameters - MSGLEVEL

Statements - Controls the listing of JCL statements in Job log

0 -   The system prints the JOB statement, comments and all statements up to the first EXEC statement

1 – The system prints all the JCL statements, control statements, Procedure statements and values assigned to symbolic parameter’s

2 – The system prints all the JCL statements and control statements

Keyword Parameters - MSGLEVEL

Messages - To control the listing of JCL and JES

messages in Job log

0 – The system prints only the JCL messages

1 – The system prints, JCL, operator and SMS

messages

Statements sub-parameter - MSGLEVEL

MSGLEVEL=([*statements*][,messages])

```
//JOB1 JOB AC1,MSGCLASS=X,MSGLEVEL=(0,1)
```

Only job statement is printed for job(job1)

```
//JOB2 JOB AC1,MSGCLASS=X,MSGLEVEL=2..
```

NO parentheses ??

```
//JOB3 JOB AC1,MSGCLASS=X,MSGLEVEL=(1,1)
```

Both Statements and Messages are printed

Messages sub-parameter - MSGLEVEL

MSGLEVEL=([statements][,messages])

//JOB1 JOB AC1,MSGCLASS=X,MSGLEVEL=(0,0).. Only JCL messages are printed for job(job1)

//JOB2 JOB ACC1,MSGCLASS=X,MSGLEVEL=2.. No value for messages Sub-parameter. Takes default.

//JOB3 JOB AC1,MSGCLASS=X,MSGLEVEL=(,1) All messages are printed

Keyword Parameters - NOTIFY

- Syntax:

  NOTIFY=<*user-id*> or

  NOTIFY=<*nodeid.userid*> *or*

  NOTIFY=<*&SYSUID*>

- Examples:

```
//JOB1 JOB ACCT1,CLASS=A,NOTIFY=TRGG01..
```

User-id **TRGG01** is notified after completion of this job

```
//JOB2 JOB ACCT1,NOTIFY=TCSMN03.TRGG02..
```

User-id **TRGG02** at **TCSMN03** is Notified after completion of this job

Keyword Parameters - TYPRUN

- Syntax:

  TYPRUN={HOLD|SCAN}
- Examples:

  //JOB1 JOB ACCT1,MSGCLASS=X,TYPRUN=SCAN.. Checks job's JCL for *syntax errors*. Job is **not submitted** for execution

  //JOB2 JOB ACCT1,MSGCLASS=X,TYPRUN=HOLD.. This job waits for execution until operator releases it for execution

TATA CONSULTANCY SERVICES

Keyword Parameter - RESTART

- Syntax:

```
RESTART={*   }          Restart job at first job step itself
        {stepname}
        {stepname.procstepname}
```

- Stepname:
  - indicates the job step from which to restart the job
- Stepname.procstepname:
  - stepname refers to the EXEC statement of the jobstep that calls the procedure.
  - Procstepname refers to the EXEC statement of the procedure step.
  - If the stepname refers to the EXEC statement in a procedure then procedure name is also specified.

Keyword parameter - RESTART

Example:

```
//JOB1 JOB ACCT1,MSGCLASS=X,RESTART=COUNT..
```

Restart from step COUNT

```
//JOB1 JOB CCT1,MSGCLASS=X,RESTART=PROC1.COUNT..
```

Restart from step COUNT in procedure step PROC1

JOB Statement Examples

- Example 1 :

```
//TRGG02X  JOB (TRG,GEN,TRGG02,AA,DT99X),’TRG’,
//       CLASS=B,MSGCLASS=X,NOTIFY=TRGG02
```

- Example 2 :

```
//GEMT01XX  JOB (GEM,CIC,GEMT01,D2,DT99X),
//       ‘GEM-PROJ’,CLASS=C,MSGLEVEL=(1,1),
//        MSGCLASS=X,NOTIFY=GEMT01
```

Summary

- The purpose of the JOB statement
- Coding Syntax of JOB statement
- Various options for the Positional and keyword parameters
- Examples of JOB statement

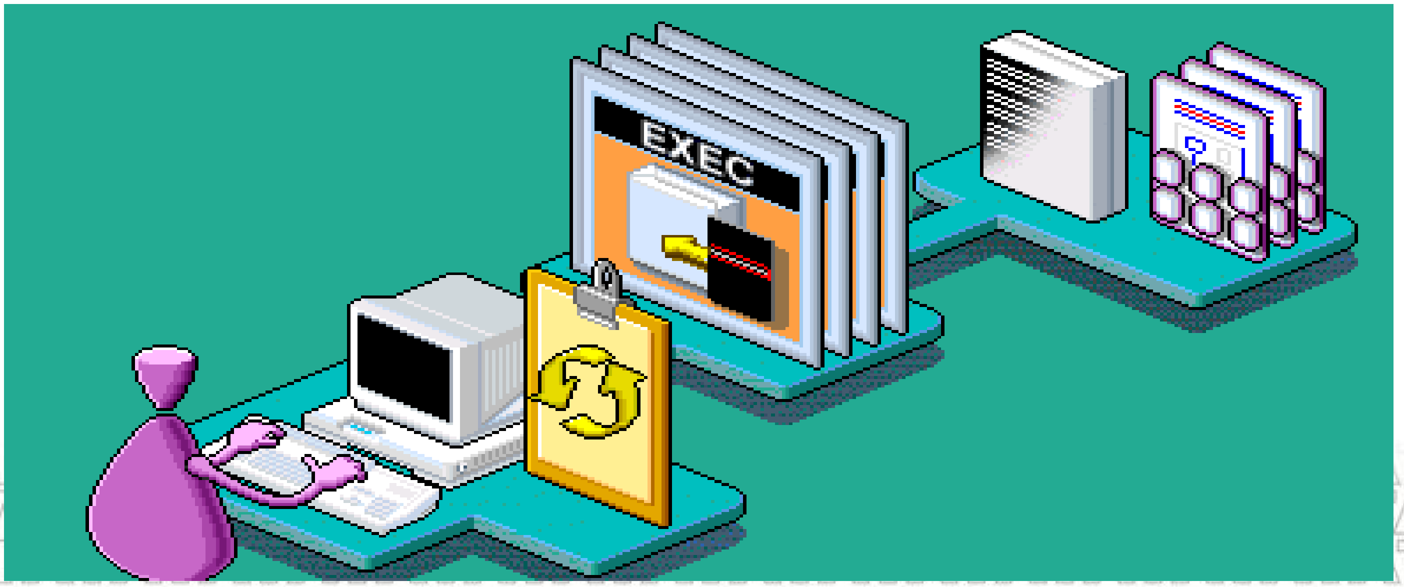
EXEC Statement

Session Coverage

- Purpose of Exec Statement
- Coding Syntax of Exec statements
- Positional and Keyword parameters in an Exec Statement
- Examples

Purpose of Exec Statement

- Used to specify which program or procedure an individual job step has to execute
- Tells system how to process the job step



Exec Statement: Example

OS

Processing parameters

Beginning of step

Parameters to program

Program or Proc name

JCL

```
      //JOB1 JOB AC1,
          //  CLASS=A
   //ST1 EXEC PGM=P1,
//   PARM='04/11/01'.
      //   REGION=800K
     // .....   DD ..
                   //
```

EXEC Statement in JCL

//JOB1 JOB	→	Beginning of JOB
//ST1 EXEC PGM=P1	→	Beginning of step ST1
//INPUT DD	→	Statements that follow the Exec step ST1
....		
....		
		Beginning of step ST2
//ST2 EXEC PGM=P2	→	Statements that follow the Exec stepST2
//INPUT DD	→	
....		
//	→	End of Job

EXEC STATEMENT - Syntax

Operation Field

Comment Field

//Stepname **EXEC** < parameters>.. **<comments>**

Name Field

Parameters Field

Operation field

```
//MYSTEP   EXEC ...

//STEP1      EXEC ...
```

EXEC in Operation Field specifies it as **Exec** statement

Parameter Field

//STEPNAME EXEC <Parameters>

The following list shows the important parameters on Exec statement

PROGRAM NAME
(PGM= or PROC=)

Positional Parameter

ACCT	PARM
ADDRSPC	REGION
TIME	COND

Keyword Parameters

Positional Parameters

- Syntax:

  PGM=<*program name*> OR

  PROC=<*procedure name*> OR

  <*procedure name*>
- Examples:

//STEP1 EXEC PGM=<u>TEST</u>

STEP1 executes program **<u>TEST</u>**

//STEP2 EXEC PROC=<u>COMPILE</u>

//STEP3 EXEC <u>COMPILE</u>

STEP2 & STEP3 invokes JCL procedure **<u>COMPILE</u>**

Key Word Parameters

The EXEC statement can contain the following Keyword parameters :

- ACCT
- PARM
- ADDRSPC
- REGION
- TIME
- COND

Key word Parameters

- ACCT – provides accounting information for the job step
- ADDRSPC – prevents the step from being paged
- REGION - specifies the region size to allocate to a job step
- TIME - imposes a CPU time limit on the job step

Keyword Parameter - PARM

- Syntax:

  PARM=(<*sub parameter*,[*sub parameter*]>)

- Examples:

  //STEP1 EXEC PGM=P1,PARM=TRGTCS

  //STEP2 EXEC PGM=P2,PARM=(10,’25/01/01’)

  25/01/01 is enclosed in apostrophes because it contains special characters ‘/’

  //STEP3 EXEC PGM=P3,PARM=‘25&&45’

  Passes string 25&45 to program P3

Program Execution using PARM

JCL for executing program CP1

```
//JOB1 JOB ACCT1,TRG-GRP,
// CLASS=B,NOTIFY=TRG???,
// MSGCLASS=X
//*This is to demonstrate
   passing
//* parameters to program
   thru JCL
//STEP1 EXEC PGM=CP1,
//   PARM='31/03/01'
//  ......   ......   .....
//
//*This JCL executes program
   CP1
```

Cobol Program-CP1

```
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING STORAGE SECTION.
LINKAGE SECTION.
01 WS-DATE
   05 WS-CTRL-LEN PIC S9(4) COMP
   SYNC.
     05 WS-CTRL-DATE  X(8).
PROCEDURE  DIVISION USING
WS-DATE.
```

EXEC Statement Examples

```
//TRGR02XX  JOB (TRG,GEN,TRGR02,AA,DT99X),
//  ‘TRG’,CLASS=B,NOTIFY=TRGR02,MSGCLASS=X

//ALLOC  EXEC PGM=IEFBR14                 ➔ This step allocates data sets
....    ........    ..........   ..........   ....
//COB  EXEC PGM=IGYCRCTL,COND=(4,LT),  ➔ This step compiles a program
//    PARM=‘NUM,SOURCE,LIB,RES’,
//    REGION=2048K
....    ........    ..........   ..........   ....
//STPZ  EXEC PGM=IEWL,PARM=‘REUS,LET’, ➔ This step does link editing
//  COND=((4,LT,COB),(4,LT,ALLOC))
....    ........    ..........   ..........   ....
//
```

DD statement in EXEC cont....

DD statement in EXEC statement is optional. DD statement is used for specifying any input to the program or to specify where the output is to be printed. If there is no input file and if there is no need of output, then we can ignore the DD statement in EXEC part.

System generation programs and the programs that are in system library **syslib** and **proclib** can be executed by using EXEC statement without a DD statement.

Eg:
```
//TRGRXX JOB 'MVS-MVS-U237011-Z9-DT08X',REGION=2096K
//IKJACCNT EXEC IKJACCNT
```

Once you login the mainframes region, the above job will start executing automatically, this needs no DD statement.

Summary

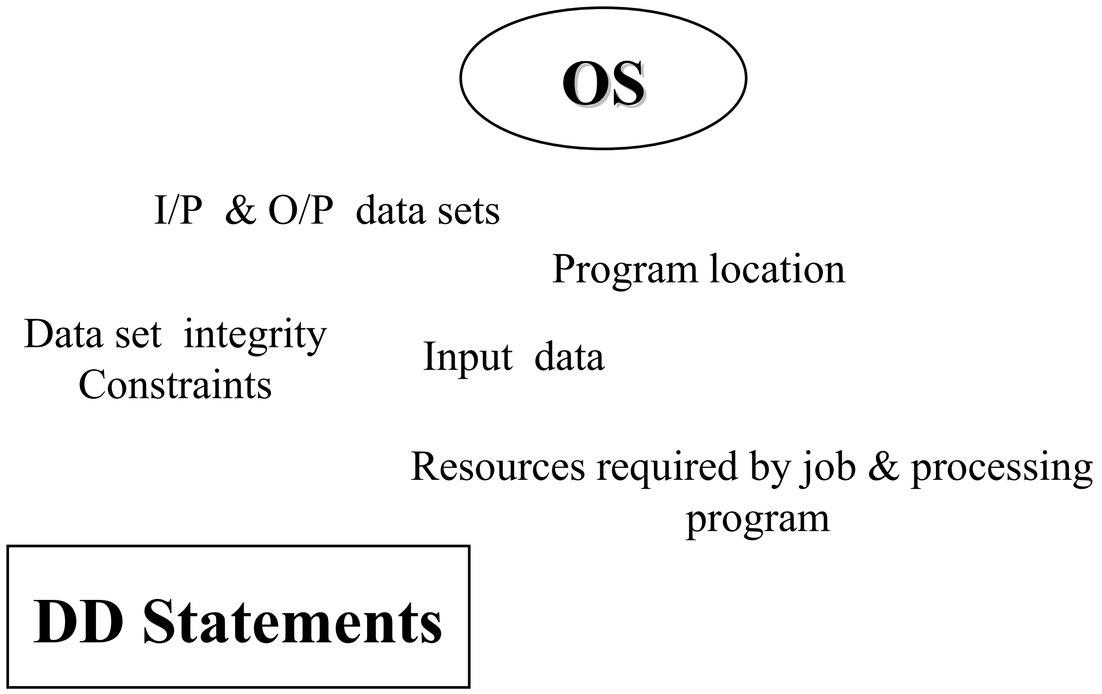
- The purpose of the Exec statement
- Coding Syntax of Exec statement
- Various options for the Positional and keyword parameters
- Examples of Exec statement

DD Statement

Session Coverage

- Purpose of DD Statement
- Coding Syntax of DD statements
  - Temporary data sets
  - DASD data sets
  - Instream and SYSOUT data sets
- Examples

Purpose of DD Statement


OS
I/P & O/P data sets
Program location
Data set integrity Constraints
Input data
Resources required by job & processing program
DD Statements

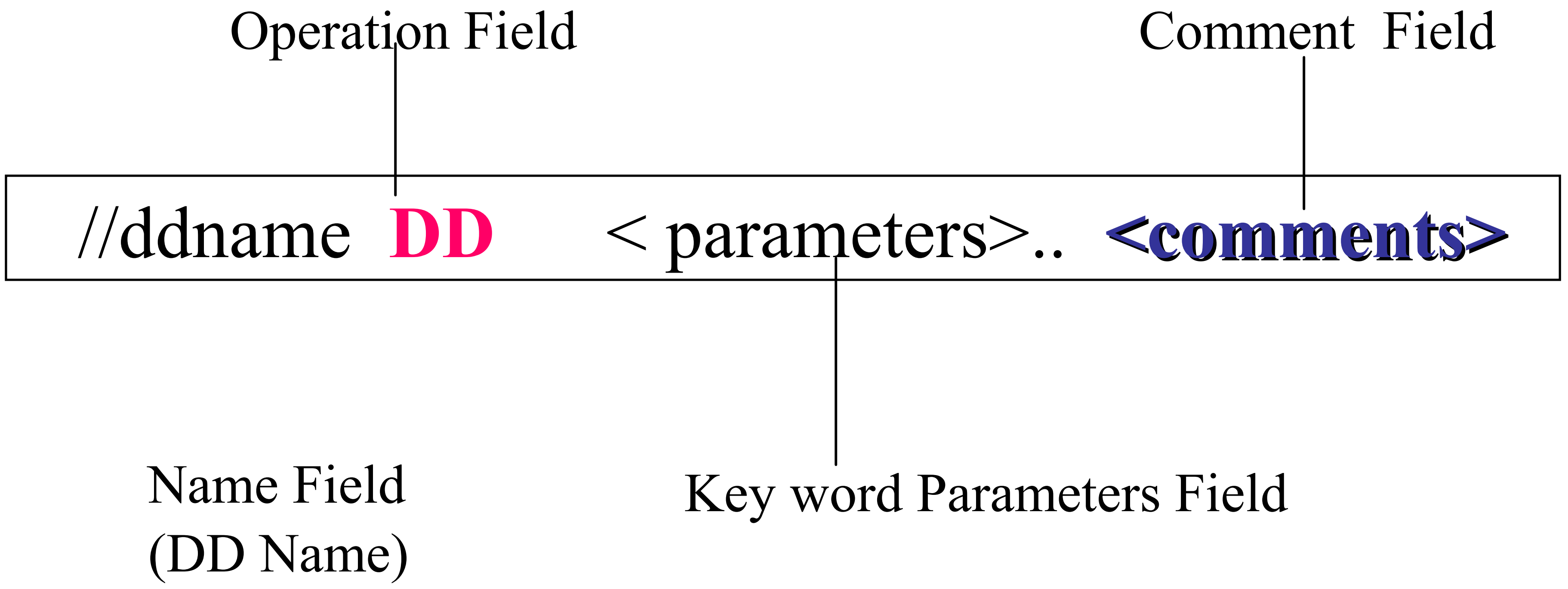
DD Statement in JCL

```
//JOB1 JOB ACT1,NOTIFY=TRGXXX
//STEP1 EXEC PGM=P1
//DD1    DD ...
//DD2    DD ...
....
//STEP2 EXEC PGM=P2
//INDD   DD ...
//DD2    DD ...
....
//
```

⇨ **DD statements** for step ***STEP1***

⇨ **DD statements** for step ***STEP2***

DD Statement - Syntax



```
//ddname  DD      < parameters>..  <comments>
```

Keyword Parameters for DASD datasets

```
//ddname  DD  DSNAME=dataset name
              ,DISP=file's status
         [    ,UNIT=device where the file exists   ]
         [    ,VOL=SER=serial number of the volume      ]
         [    ,SPACE=DASD space to be allocated     ]
         [    ,DCB=Options for file's data control block ]
```

DSNAME Parameter

- DSN Parameter specifies the *Physical file name* to be associated with the DD Name referred by processing program or system.

DD Name	Data set name in DSN
LOGICAL FILES NAME or ***DD NAME -*** Referred by processing *program* or *system*	***PHYSICAL DATA SET*** For the DD name referred by program or system

TATA CONSULTANCY SERVICES

DSNAME Parameter

- Syntax:

  DSN=<*data set name*> | <*DSNAME(member)*>| NULLFILE

- *Dataset name can be* :
  - Qualified data set name
  - Unqualified data set name
  - NULLFILE
  - PDS member name
  - Temporary data set name

DSNAME Parameter : Examples

- Qualified data set name

  //INFILE DD DSN=TRGXX.SRC.COBOL, ...

- Unqualified name as data set name

  //DD1 DD DSNAME=TRGXX, ...

- Data set name for a Dummy Data set

  //OUTFILE DD DSN=NULLFILE

- Member of a PDS as data set name(qualified data set name)

  //INFILE DD DSN=TRGXX.SRC(<u>MEM1</u>) .....

DD Statements in relation with programs

Cobol Program-**CP1**

```
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
SELECT  INPUT ASSIGN TO
 INDD.
SELECT  OUTPUT ASSIGN TO
 OUTDD.
DATA DIVISION.
WORKING STORAGESECTION.
PROCEDURE
   DIVISION.......   ......   ...
   ....
```

JCL for executing program CP1

```
//JOB1 JOB ACCT1,TRG-GRP,
// CLASS=B,NOTIFY=TRG???,
//STEP1 EXEC PGM=CP1
//INDD  DD ......
//OUTDD   DD .......
//
//*This is to demonstrate
   relation between program
   ddnames names to ddnames in
   JCL.
//*This  Run JCL for program CP1
```

DISPOSITION Parameter - DISP

- Syntax:

  DISP=<([*current status*][,*normal termination DISP*]
  [,*abnormal termination DISP*])>

CURRENT ***STATUS***	*NORMAL* ***TERMINATION***	*ABNORMAL* ***TERMINATION***
NEW	DELETE	DELETE
OLD	KEEP	KEEP
SHR	PASS	CATLG
MOD	CATLG	UNCATLG
	UNCATLG	

DISP - Current Status : Examples

- Sharing the already existing data set.

  ```
  //INFILE  DD  DSN=TRGXX.SRC(MEM1),DISP=SHR
  ```

- Exclusive control over already existing data set or writing to existing data set. Existing records will be deleted.

  ```
  //OUTFILE DD DSN=TRGXX.OUT,DISP=OLD
  ```

- Creating new data set.

  ```
  //NEWDD  DD  DSN=TRGXX.NEW,DISP=NEW........
  ```

- For updating the existing data set in exclusive access mode and is positioned at the end of the data, so the records may be added to the end.

  ```
  //DD1  DD DSN=TRGXX.TRN,DISP=MOD .....
  ```

DISP - Normal Termination : Examples

- Request to keep the data set on normal completion of step.

  //INFILE DD DSN=TRGXX.SRC,DISP=(SHR.KEEP)

- To delete the data set on normal completion of step.

  //OUTFILE DD DSN=TRGX.OUT,DISP=(OLD,DELETE)

- To catalog the data set on normal termination of step.

  //NEWDD DD DSN=TRGX.NEW,DISP=(NEW,CATLG)

- To pass data set for use by subsequent step on normal termination of step.

  //DD1 DD DSN=TRGXX.TRN,DISP=(MOD,PASS)

DISP - Abnormal Termination : Examples

- To Keep the data set on abnormal completion of step.

```
//FILE1  DD  DSN=TRGX.SRC,DISP=(SHR,PASS,KEEP)
```

- To Uncatalog the data set on abnormal completion of step.

```
//OUT1 DD DSN=TRGX.OUT,DISP=(OLD,,UNCATLG)
```

- To Catalog the data set on abnormal termination of step.

```
//NEWDD  DD  DSN=TRGX.NEW,DISP=(NEW,,CATLG)
```

- To Delete data set on abnormal termination of step.

```
//DD1  DD DSN=TRGXX.TRN,DISP=(,PASS,DELETE)
```

DISP Parameter - Defaults

DISP coded on DD statement	System Interpretation by default
NO DISP PARAMETER	DISP=(NEW,DELETE,DELETE)
DISP=SHR	DISP=(SHR,KEEP,KEEP)
DISP=OLD	DISP=(OLD,KEEP,KEEP)
DISP=(,CATLG)	DISP=(NEW,CATLG,CATLG)
DISP=(OLD,,DELETE)	DISP=(OLD,KEEP,DELETE)
DISP=MOD (Treated as new)	DISP=(MOD,DELETE,DELETE)
DISP=MOD (Treated as old)	DISP=(MOD,KEEP,KEEP)

Parameters to Define Location of Datasets

- UNIT
  - Specifies the physical device where an existing dataset can be found or where a new one can be found
  - Specify only installation defined group of devices
- VOLUME
  - Specifies the volume serial number of the particular tape or disk involved
  - Volume parameter is optional for new datasets
- For Cataloged datasets, Unit and Volume details need not be specified

UNIT Parameter

- Syntax:

  UNIT=({[*device-number*][*device-type*] [*group-name*]}

  {[,*unit-count*] [,P]} [,DEFER])

  or UNIT=AFF=*ddname*

- Examples:

  //DD1 DD DSN=TRG1.U1,DISP=(NEW,KEEP),

  // UNIT=3380 ⟶ *device-type*

  //DD2 DD DSN=TRG1.U,UNIT=SYSDA ⟶ *group name*

UNIT Parameter

//DD1 DD DSN=TRG1.U1,DISP=MOD,

// UNIT=(3390,2) $\longrightarrow$ 2 *devices are requested*

//DD2 DD DSN=TRG1.U,DISP=SHR,

// UNIT=(,,DEFER) $\longrightarrow$ *request to defer mounting*

//DD3 DD DSN=TRG1.AFF,DISP=OLD,

// UNIT=AFF=DD1 $\longrightarrow$ *requests same two devices as Step* DD1

VOLUME Parameter

- Syntax:

  VOLUME= ([PRIVATE] [,RETAIN]

  [,*vol-seq-number*] [,*vol-count*]

  [,{SER=(*ser-no*[,*ser-o*]..)} {REF=*dsname*} {REF=*.*ddname*}])

VOLUME Parameter : Examples

```
//IN  DD  DSN=TRG.V1,DISP=(,KEEP),SPACE=(TRK,10).
//  VOLUME=(PRIVATE,SER=WORK01)

//OUT1 DD DSN=TRG.V2,DISP=SHR,
//  VOLUME=(,RETAIN,2)

//OUT2 DD DSN=TRG.V3,DISP=NEW,SPACE=(1024,10),
//  VOLUME=(,RETAIN,,2,SER=(DEV01,DEV02))
```

VOLUME Parameter : Examples

```
//IN  DD  DSN=TRG.V1,DISP=(,KEEP),SPACE=(TRK,10),
//  VOLUME=(,RETAIN,REF=TRG.V1)

//OUT1 DD DSN=TRG.V2,DISP=SHR,
//  VOLUME=SER=DEV01

//OUT2 DD DSN=TRG.V3,DISP=NEW,SPACE=(1024,10),
//  VOLUME=REF=*.OUT1
```

SPACE Parameter

- Syntax:

  SPACE=

  ({TRK,}(*prqty*[,*secqty*][,*dir*])[,RLSE][,CONTIG][,ROUND])

  {CYL,}

  {*blklen*,}

Number of Units

Special Processing Requests

Measurement Unit

SPACE Parameter : Examples

- Requests space for a PS data set in tracks.

```
//DD1 DD DSN=TRG.X1,SPACE=(TRK,(10,5))
```

- Requests 1 cylinder for a new PS data set.

```
//DD2 DD DSN=TRG.X2,SPACE=(CYL,1)
```

- Requests space for a new PDS in data blocks of size 1KB.

```
//DD3 DD DSN=TRG.X3,SPACE=(1024,(9,5,2))
```

- Requests 10 tracks out of which four 256-byte records are for directory space, for a new PDS.

```
//DD4 DD DSN=TRG.X4,SPACE=(TRK,(10,,4))
```

SPACE Parameter : Examples

- Releases the DASD space which is not used by the dataset.

  //DD1 DD DSN=TRG.X1,SPACE=(TRK,(10,5),RLSE)

- Instructs the OS to make primary allocation with a single extent of contiguous cylinders

  //DD2 DD DSN=TRG.X2,SPACE=(CYL,1,,CONTIG)

- Instructs the OS to allocate in terms of whole cylinders even though you specify the amount of space in terms of blocks.

  //DD3 DD DSN=TRG.X3,SPACE=(1024,(9,5,2),,ROUND)

- Instructs the OS to allocate the largest available free extent on the volume to the file.

  //DD4 DD DSN=TRG.X4,SPACE=(TRK,10,RLSE,MXIG)

DCB Parameter

- DCB to define the Characteristics of individual data sets.
- Syntax:

```
DCB=(LRECL=nn[,BLKSIZE=yy][,RECFM=zz]
 [,DSORG=mm])
```

nn	- Logical Record Length
yy	- Block size
zz	- Record Format (F,FB,V,VB or U)
mm	- Data Set Organization (PS or PO)

DCB Parameter : Examples

- DCB for new PS data set with *fixed record length* of 80 with *blocking*(block contains 10 records)

```
//DD1  DD  DSN=TRG.DCB,DISP=(NEW,CATLG),
//   DCB=(LRECL=80,RECFM=FB,BLKSIZE=800),
//   SPACE=(TRK,(5,5))
```

- DCB for new PDS data set with *variable record length* of 80 and no blocking..

```
//DD1  DD  DSN=TRG.DCB,DISP=(NEW,CATLG),
//   DCB=(LRECL=80,RECFM=V,DSORG=PO),
//  SPACE=(TRK,(5,5,2))
```

DD Statement for SYSOUT data sets

- Syntax:
  //ddname DD SYSOUT=x

- Example:
  //SYSPRINT DD SYSOUT=A

- The SYSOUT parameter
  - indicates that the data set should be processed by JES2/JES3
  - Specifies the output class associated with the data set

Summary

- The purpose of the DD statement
- Coding Syntax of DD statements for
  - DASD data sets
  - Instream data sets
  - SYSOUT data sets
- Dataset Attributes

THANK YOU