Job Control Language (JCL)

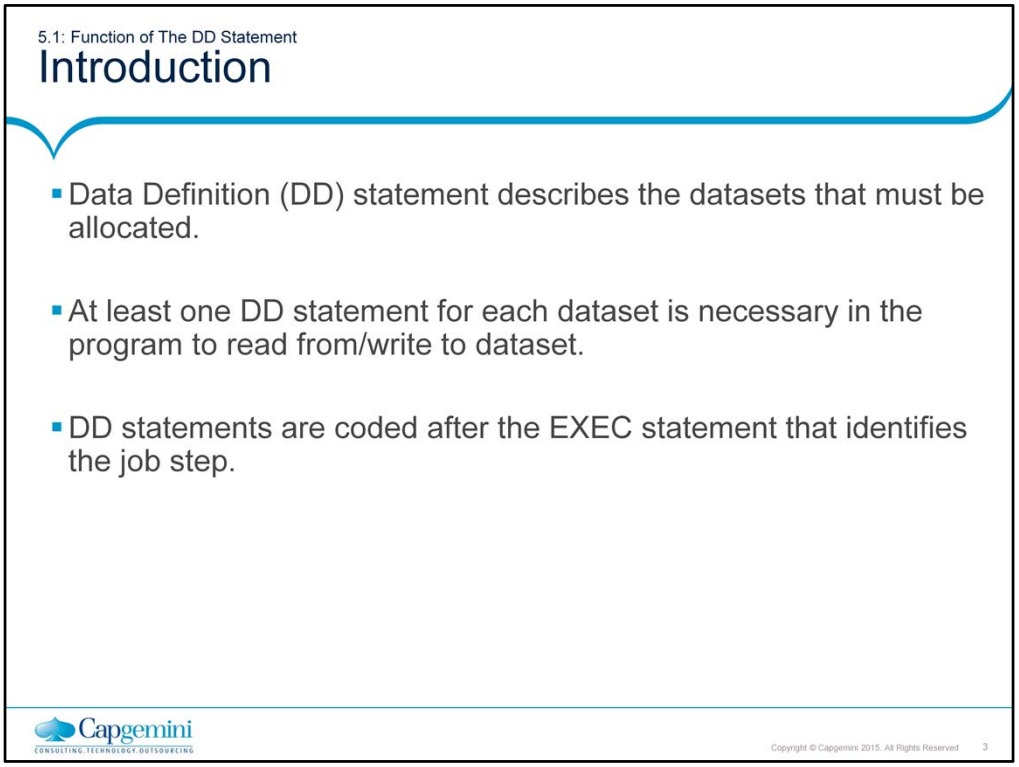
Lesson 5: The DD statement

Lesson Objectives

- On completion of this lesson, you will be able to:
  - Use DD statement
  - Use DSN, DISP, UNIT, VOL, SPACE, LABEL, DCB, SYSOUT, SYSIN,SYSPRINT, DUMMY parameter
  - Use JOBLIB and STEPLIB statement
  - Use of JobCat and StepCat
  - Request storage dump

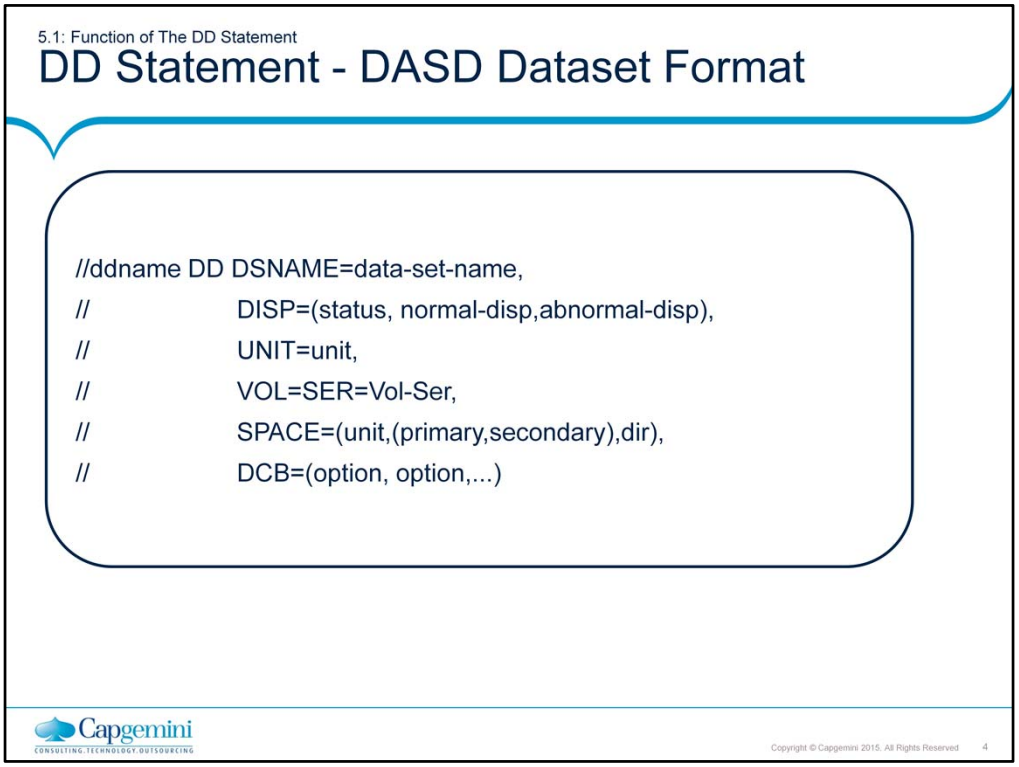
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2



Introduction:
A DD (Data Definition) statement must appear in a step when the executing program expects to read from or write to a dataset. In other words, DD statement describes the dataset.

The maximum number of DD statements in a step is 3273. The DD statement can be coded in any order and always appears after the EXEC statement with the exception of JOBLIB, JOBCAT, and PROCLIB DD statement.



As job executes the system performs device and space allocation for each ddname specified.
Each ddname must be unique & 1 to 8 characters long comprising of alphanumeric or national character & first character must be alphabetic or national character.If there are duplicate ddnames, even though the system will make the allocation specified, it will direct all related messages to the first ddname.
AVOID using ddnames that begin with ‘SYS’, ‘JOB’, ‘STEP’ are considered as reserve words for IBM products.JOBLIB, JOBCAT, SYSOUT, SYSIN, STEPLIB, STEPCAT, SYSUDUMP, STEPLIB, SYSABEND, SYSDBOUT
As job executes the system performs device and space allocation for each ddname specified.
Each ddname must be unique & 1 to 8 characters long comprising of alphanumeric or national character & first character must be alphabetic or national character.If there are duplicate ddnames, even though the system will make the allocation specified, it will direct all related messages to the first ddname.
AVOID using ddnames that begin with ‘SYS’, ‘JOB’, ‘STEP’ are considered as reserve words for IBM products.JOBLIB, JOBCAT, SYSOUT, SYSIN, STEPLIB, STEPCAT, SYSUDUMP, STEPLIB, SYSABEND, SYSDBOUT

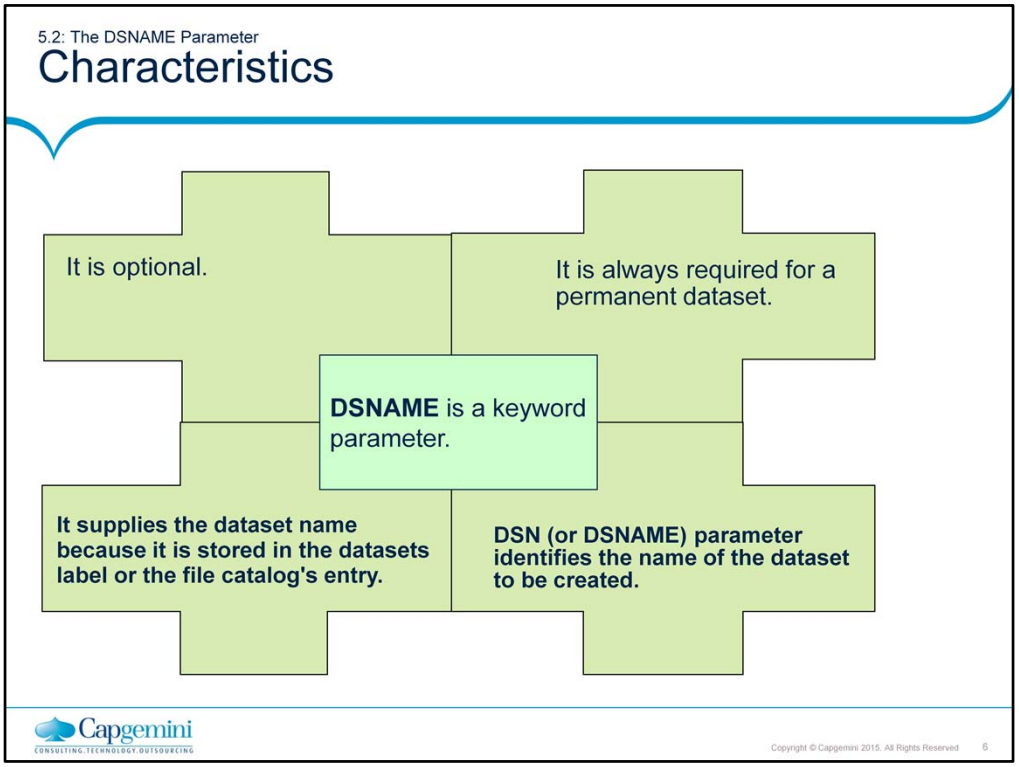
5.1: Function of The DD Statement

DD Statement

- Classified into Permanent and Temporary datasets
  - Permanent datasets are used as a data-store on disks and tapes.
  - Temporary datasets are used as work areas within a job; they last only for the duration of the job.
  - A Temporary dataset name starts with && followed by 1 to 6 characters - DSN=&&TEMPRY
    - Absence of DSN coding also denotes a temporary dataset.
  - A DSN coding without && indicates a Permanent dataset.
- If the dataset (DSN) is not found in the system catalog, the system throws a “JCL error”.
- Dataset names (DSN) cannot be duplicated in the VTOC. If duplicated, the system returns a “CC of 8”.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5



Characteristics:
The DSN (or DSNAME) parameter identifies the name of the dataset to be created or retrieved.
General Syntax

```
DSN=name| NULLFILE | referback          - keyword parameter
```

Name: This could be a qualified name. This name consists of two or more simple name separated by periods for a maximum of 44 characters.

For example (i)
```
DSN=DA0001T.CG.EMPFILE
```

For example (ii)
```
DSN=DA0001T.CG.COBOL(ASS1)
```

This describes a sequential dataset, that means, ASS1 is a member of PDS/library DA0001T.CG.COBOL

For example (iii)
```
DSN=&&name
```

A simple name preceded by two ampersands identifies a temporary dataset. It is considered temporary because it is not retained beyond job termination.

5.2: The DSNAME Parameter

Format and Example

- Format:

```
DSN=name
```

- Example:

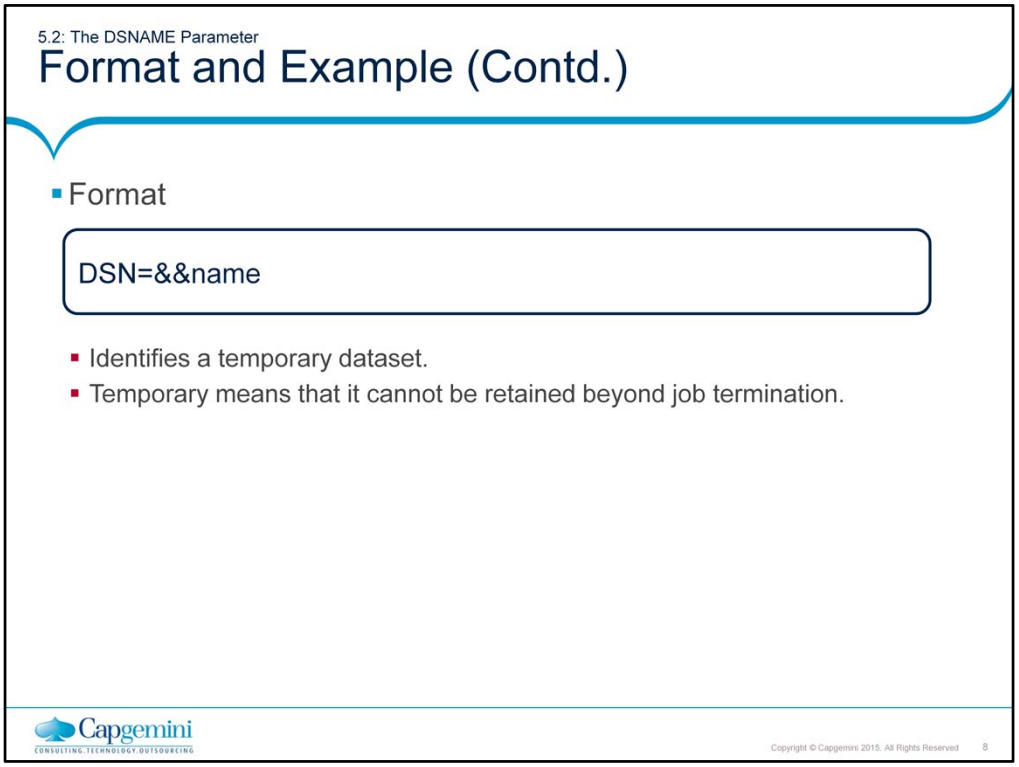
```
DSN=DA0001T.CG.PAYROLL.MASTER
```

- To refer to a member of a PDS:

```
//TRANFILE  DD DSN=DA0001T.CG.TRANFILE(mem1)
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7



Format and Example (Contd.):
The system generates a name with the following format:

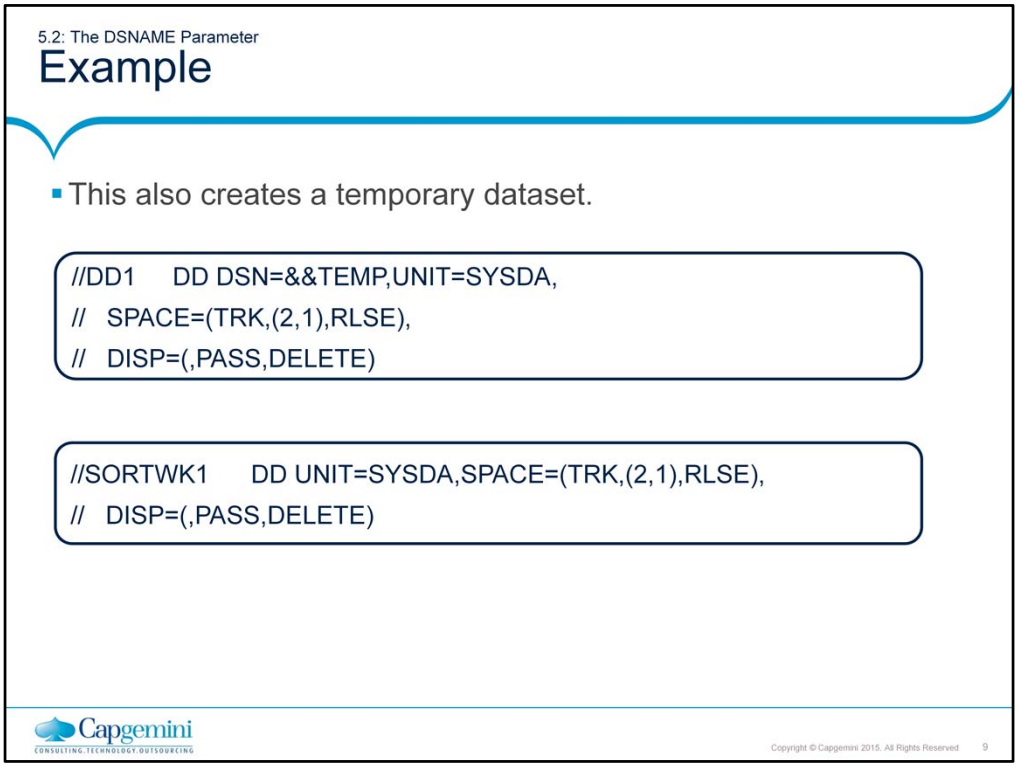
SYSyyddd.Thhmmss.RV001.jobname.name

yyddd – date as per Julian calendar;
hhmmss – uses 24-hour clock. It is the time of JOB initiation (beginning of JOB execution)

RV001 –system provided information in reference to the reader;
jobname – as it appears in the JOB statement;
name – whatever is coded after &&.

For e.g. DSN=&&temp. The system generates the following name:

SYS03173.T090000.RV001.DA0001TA.TEMP



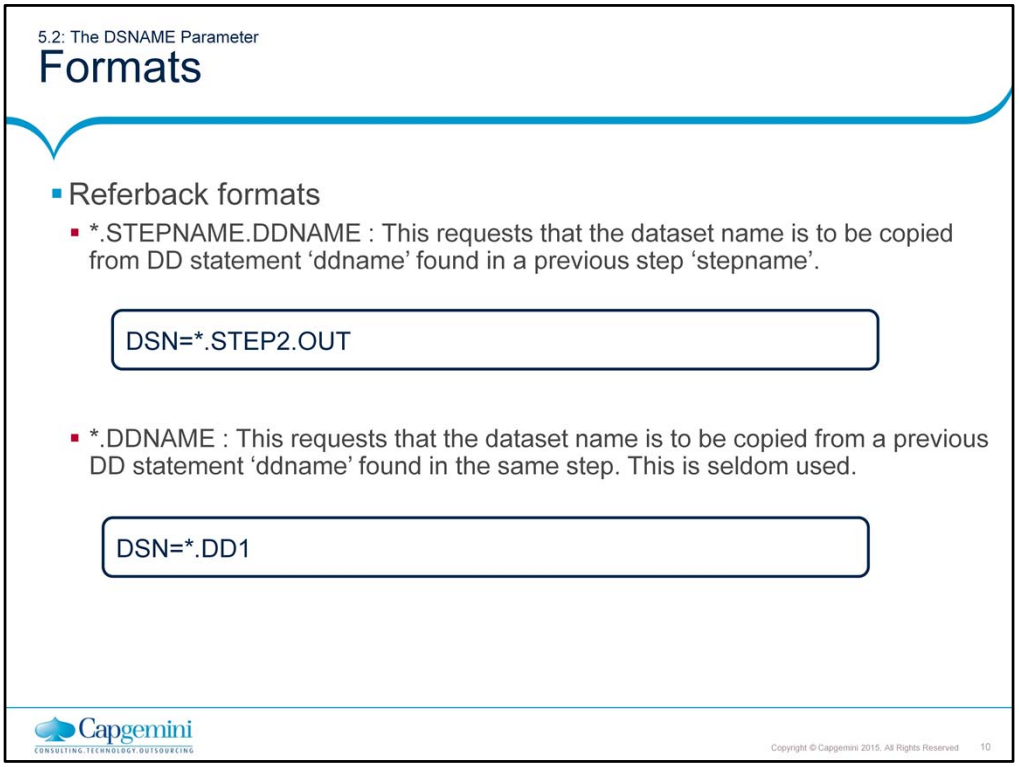
Example:

If the DSN name is omitted from a DD statement (except DD *, SYSOUT and DUMMY), it also indicates a temporary dataset. However, the system generates a name with the following format:

```
SYSyyddd.Thhmmss.RV001.jobname.R0000001

//SORTWK1     DD UNIT=SYSDA,SPACE=(TRK,(1,2),RLSE)
SYS00173.T100000.RV001.DA0001TA.R0000001
```

This form is basically used when a step requires a work dataset (a dataset created at the beginning of the step's execution and deleted at the end). Mostly used in utilities.

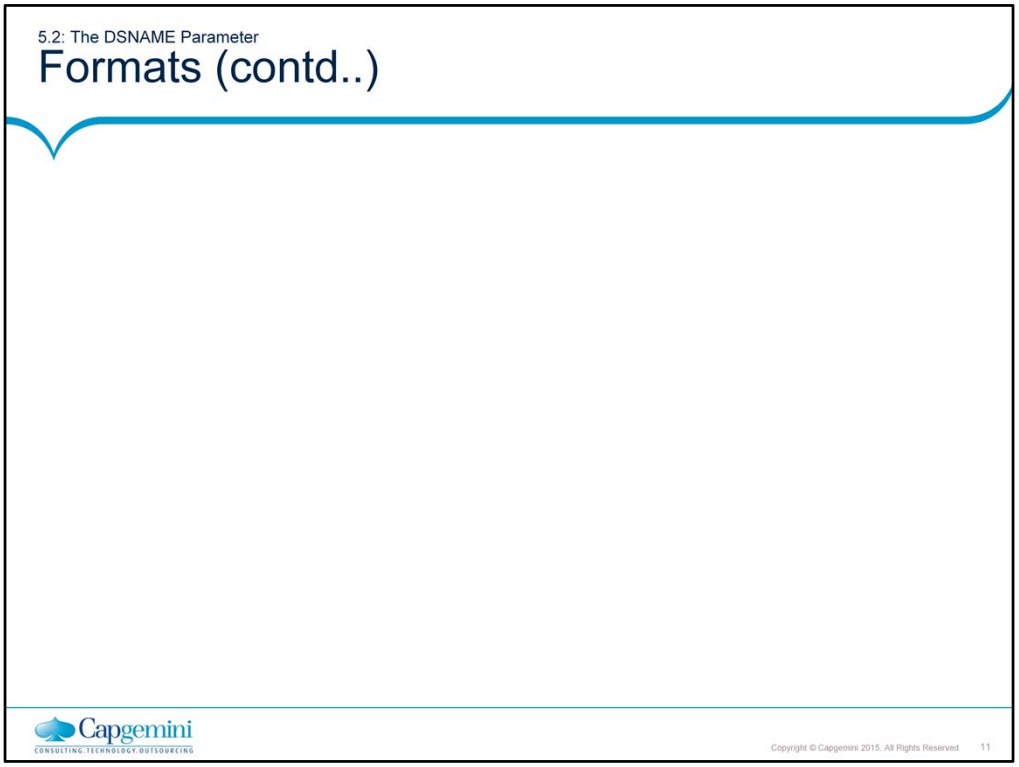


REFERBACK: Backward reference (Referback) is used to copy information from a previous DD statement (within the same job) thus simplifying JCL code.
PGM can be Referback. On the DD statement, this can be applied to three parameters:

DSN
VOL
DCB

Referback on a DD statement

```
//IGTRN01A    JOB     NOTIFY=DSRC012
//STEP010      EXEC    PGM=PGM1
//DD1           DD      DSN=EMP.PDS,DISP=(NEW,CATLG),
//       DCB=(LRECL=80,RECFM=FB,BLKSIZE=800),
//       ……..……
//DD2             DD        DSN=INP.PDS,DCB=*.DD1, ……..
//STEP2     EXEC     PGM=PGM2
//DD2        DD      DSN=STU.PDS, DCB=*.STEP010.DD1, …..
```



Referback on a PGM statement

```
//LKED   EXEC PGM=HEWL,REGION=1024K,PARM='XREF,LIST'
//SYSLIB   DD  DSNAME=CEE.SCEELKED,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//SYSLIN   DD  DSNAME=DSRC746.SATYA.OBJ(ADD1),DISP=
//SYSLMOD  DD  DSNAME=DSRC746.SATYA.LOAD(ADD1),DISP
//SYSUT1   DD  UNIT=SYSDA,SPACE=(TRK,(10,10))
//*
//GO       EXEC PGM=*.LKED.SYSLMOD
//SYSPRINT  DD  SYSOUT=A
//SYSOUT    DD  SYSOUT=A
//SYSIN     DD  *
10

/*
```

5.2: The DSNAME Parameter

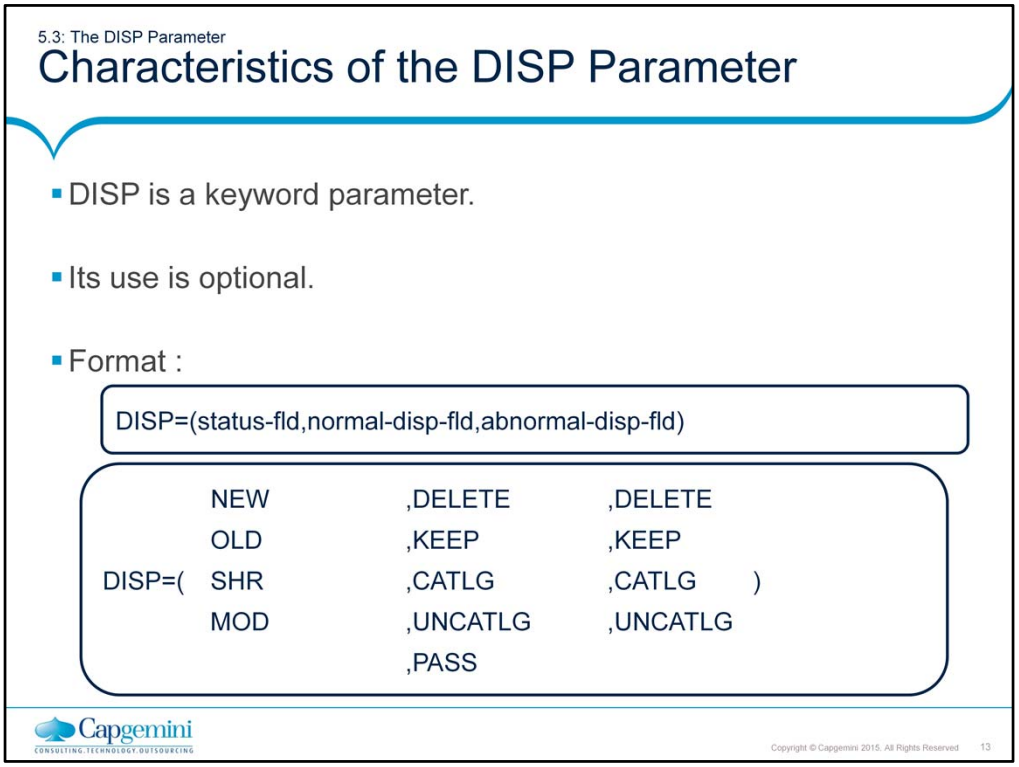
Formats (contd.)

- *.PROCEXEC.STEPNAME.DDNAME : Requests that the dataset name be copied from DD statement ‘ddname’ found in a previous step ‘stepname’ found within procedure ‘procexec’.

```
DSN=*.PS4.STEP2.OUT
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 12



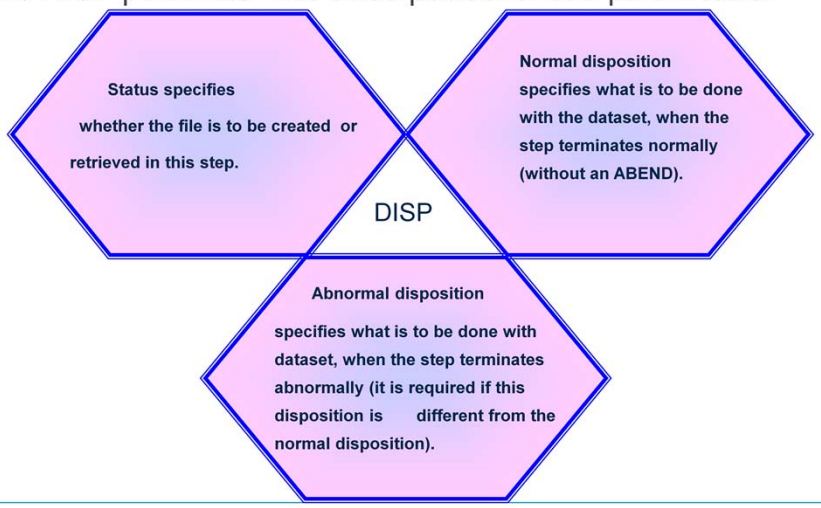
Characteristics of the DISP Parameter:

The DISP parameter specifies the following:
It specifies if the dataset is to be created or retrieved.
It indicates how to dispose off the dataset when the step terminates (normally or abnormally).

5.3: The DISP Parameter

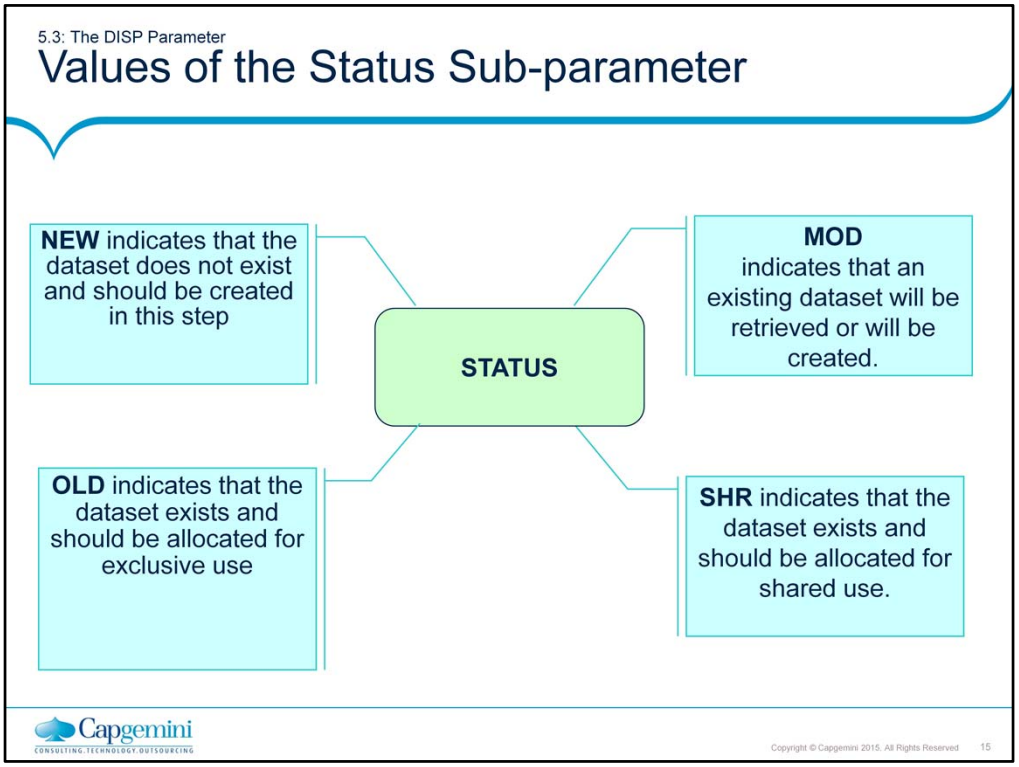
Characteristics of the DISP Parameter

- The DISP parameter has three positional sub-parameters:



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14



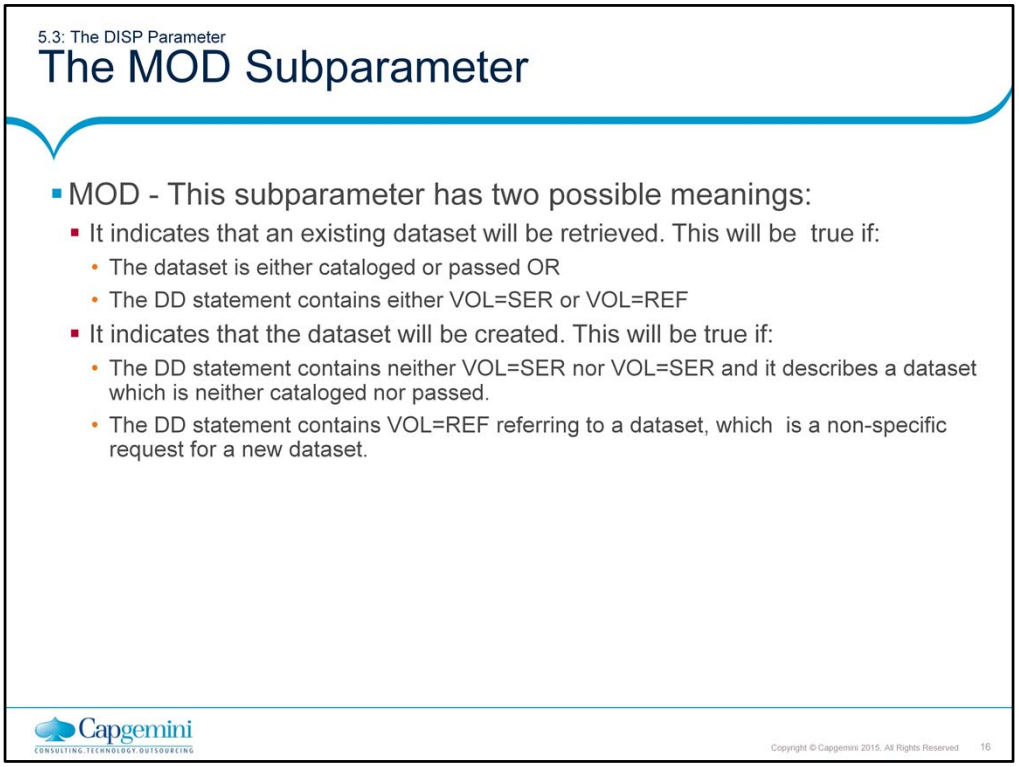
The status-field: This field tells the system whether the dataset is to be created or retrieved.

NEW – Indicates that the dataset will be created in this step

OLD - Indicates that an existing dataset will be retrieved and demands exclusive control

SHR - Indicates that an existing dataset will be retrieved. It also indicates that this dataset, if on disk, can be shared with one or more other users

(Mnemonic hint: NOMS; N:New, O: OLD, M: MOD, S: SHR)



The MOD Subparameter:
MOD - This sub parameter has two possible meanings:

Indicates that an existing dataset will be retrieved. This will be true if:

The dataset is either cataloged or passed.
The DD statement contains either VOL=SER or VOL=REF (a VOL VOL=REF referring to a DD statement, which is a nonspecific request for a new dataset, is not included).

Indicates that the dataset will be created. This is true if:

The DD statement contains neither VOL=SER nor VOL=REF and it describes a dataset which is neither cataloged nor passed.
The DD statement contains VOL=REF referring to a DD statement, which is nonspecific, a request for a new dataset.

Example 1:

```
//DD1    DD  DSN=DA0001T.EMPFILE,DISP=(MOD,CATLG),
//          UNIT=TAPE
```

Explanation:

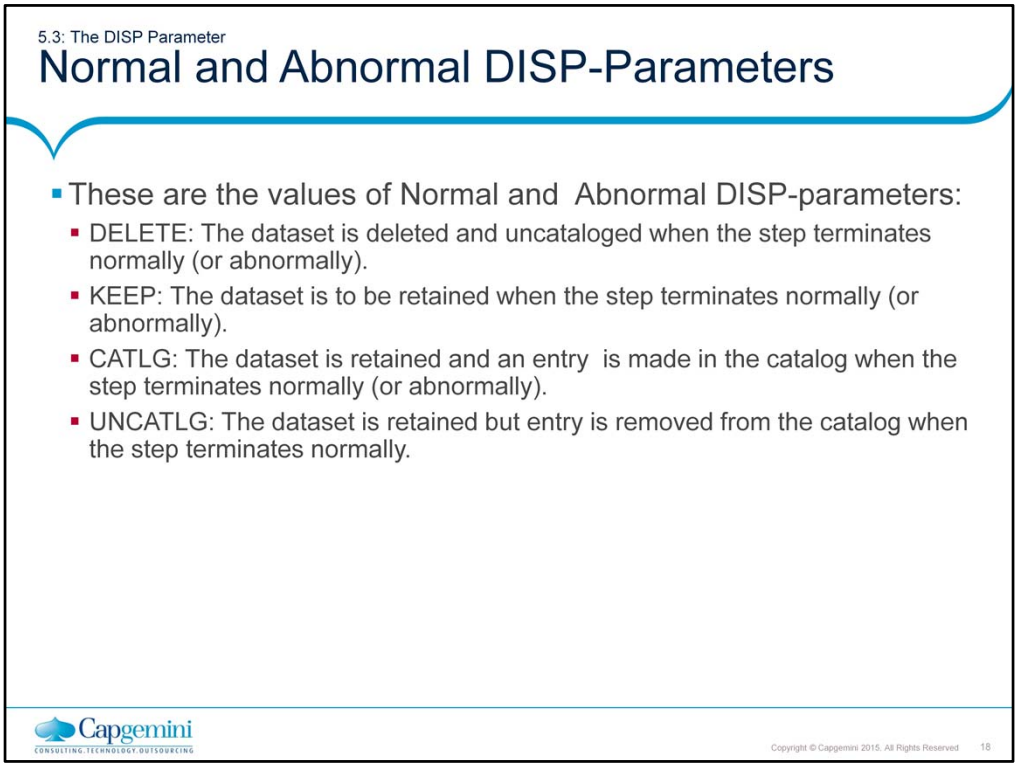
- The system assumes DA0001T.EMPFILE to be an existing dataset. Since the DD statement contains neither VOL=SER or VOL=REF, the system searches the catalog and gets volume information from the catalog entry. The volume having been found, the dataset will be treated as existing dataset.
- Had the dataset been neither cataloged nor passed, the system would have been unable to find the volume information and MOD will default to new.

Example 2:

```
//DD1    DD  DSN=DA0001T.EMPFILE,DISP=(MOD,CATLG),
//          UNIT=SYSDA,VOL=SER=BS3003,SPACE=(TRK,(1,2))
```

Explanation: Since VOL=SER is specified; the fate of MOD is sealed, whether or not it exists. It will be treated as OLD (with appropriate positioning). If the dataset exists on that volume no problem, however, if it does not exist the result will be S213-04 ABEND failure (i.e. dataset does not exist)

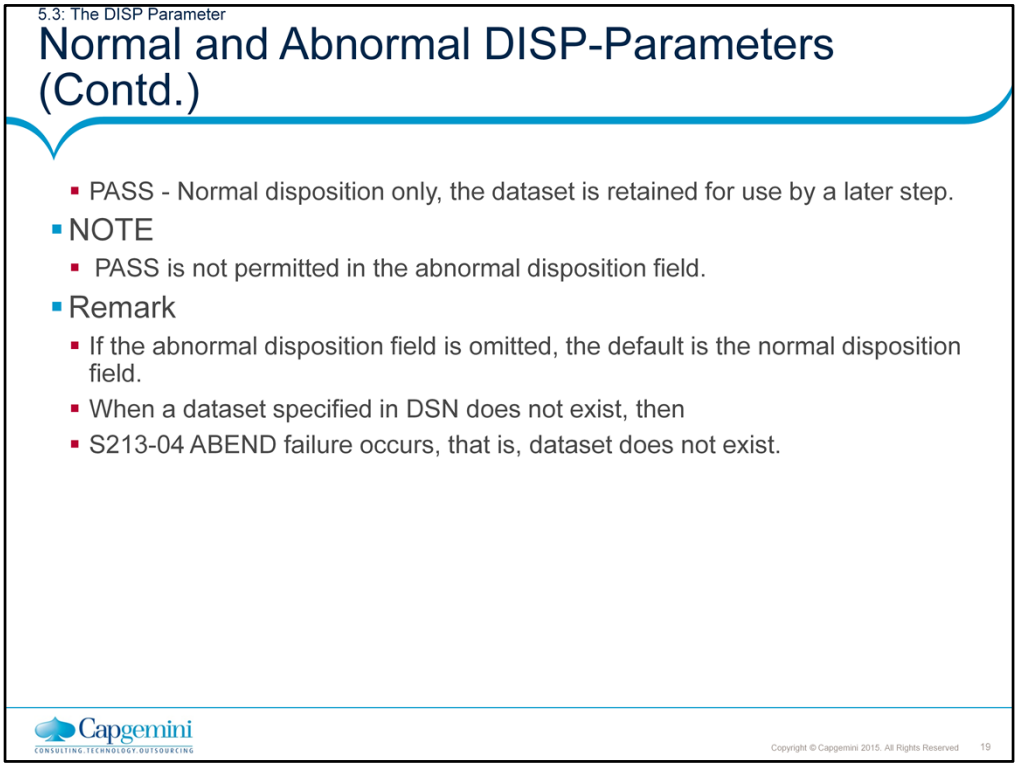
Note: When UNIT and VOL=SER is specified the system does not search the catalog to locate the dataset.



The normal disposition field: This field is used to tell the system how to dispose of the dataset when the step terminates normally (without an ABEND).

Normal and Abnormal DISP-Parameters:
Values of Normal and Abnormal DISP-parameters are as follows: (Mnemonic Hint: DUCK)
DELETE: This indicates that the dataset is to be deleted when the step terminates. For an existing dataset, OLD, SHR or MOD (not defaulting to NEW), the dataset is also uncataloged, if the catalog is used while retrieving the dataset. It only deletes the dataset if the catalog is not used during the retrieval. This means that for a cataloged dataset, if you specify UNIT and VOL=SER, the system does not search the catalog.

Note:
(i) When a tape dataset is deleted, nothing happens. A tape dataset cannot be deleted through the DISP parameter. It is effectively deleted when the dataset is written over.
(ii) A VSAM cluster cannot be deleted by coding DISP=(OLD, DELETE) as it defaults to DISP=(OLD,KEEP).
(iii) A member of PDS cannot be deleted because DISP applies to the entire PDS, and as a result, it deletes the entire PDS. Use either TSO or IEHPROGM utility.
The system always issues a message indicating " DELETED' or "NOT DELETED N" , where N indicates the reason for failing.



Normal and Abnormal DISP-Parameters (contd.):

KEEP: This Indicates that the dataset is to be kept when the step terminates. The system takes no action and issues a message indicating the dataset is kept. Also, the system issues a message “KEPT”. Note that “NOT KEPT” message does not exist.

Note: KEEP does not imply CATLG. As a result, DISP=(NEW,KEEP) should be rarely used because next time you retrieve the dataset, you need to specify UNIT and VOL=SER.

CATLG: This indicates that the dataset is to be kept and an entry for it placed in the catalog when the step terminates.

PASS: This indicates that an entry for the dataset (containing DSN, VOL=SER and UNIT information) be placed on a table in storage (Passed Dataset Queue). This entry is to be used in a subsequent step to “receive the passed dataset”. A message is displayed “PASSED”.

The Abnormal (or Conditional) Disposition Field:

This field is used to notify the system how to dispose off the dataset when the step terminates abnormally (ABENDs). It is required only if this disposition is different from the normal disposition.

DELETE, **KEEP**, **CATLG**, and **UNCATLG** have the same meaning as they have in the normal disposition. Note that **PASS** is not permitted in the abnormal disposition field.

The best example of using the abnormal disposition field is,

```
DISP=(NEW,CATLG,DELETE)
```

If there is ABEND, the dataset is to be deleted. This eliminates future manual intervention to delete and uncatalog the dataset in order to restart.

Defaults: Some defaults in the DISP parameter are fixed and others are variable.

- If the DISP parameter is omitted, the default is always **(NEW,DELETE)**.

```
//SYSUT1      DD  UNIT=SYSDA,SPACE=(TRK,(1,2))
//*     (NEW,DELETE)  IS THE DEFAULT
```

- If the status is omitted, the default is always **NEW**.
  DISP=(,CATLG) is same as DISP=(NEW,CATLG)

- If the normal disposition field is omitted:
  - If the status field is **NEW**, the default is **DELETE.**
  - If the status field is **OLD** or **SHR** and the dataset name is non-temporary:
    - If the DD statement is not receiving a passed dataset, the default is **KEEP**.

```
//DD1     DD DSN=DA0001T.EMPFILE,DISP=SHR
```

    - If the DD statement is receiving a passed dataset, which was created during the execution of the job and was never given a permanent disposition, the default is **DELETE**.

```
//S1        EXEC        PGM=PR
//DD1       DD          DISP=(, PASS), DSN=USER1.PDST,
//          UNIT=SYSDA,
//          DCB=(BLKSIZE=23440, LRECL=80, RECFM=FB),
//          SPACE=(TRK, (50,10),RLSE)
//S2        EXEC        PGM=PC
//DD2       DD          DISP=OLD, DSN=USER1.PD
```

In DD2, **DISP=OLD** defaults to **DISP=(OLD, DELETE)**.

The Abnormal (or Conditional) Disposition Field (Contd.):

- If the normal disposition field is omitted (Contd.):
  - If the status field is **OLD** or **SHR** and the dataset name is non-temporary (Contd.):
    - If the DD statement is receiving a passed dataset, which was created during the execution of the job but was given permanent disposition since being created, the default is **KEEP**.

```
//S1      EXEC    PGM=PR
//DD1     DD      DISP=(, PASS),
     DSN=USER1.PDST,UNIT=SYSDA,
//   DCB=(BLKSIZE=23440, LRECL=80, RECFM=FB),
//   SPACE=(TRK, (50,10), RLSE)
//S2      EXEC    PGM=PC, COND=(4, LT)
//DD2     DD      DISP=(OLD, CATLG), DSN=USER1.PDST
//S3      EXEC    PGM=PF, COND=(4,LT)
//DD3     DD              DISP=(OLD,PASS), DSN=USER1.PDST
//S4      EXEC    PGM=PK, COND=(4, LT)
//DD4     DD              DISP=OLD, DSN=USER1.PDST
```

In **DD4**, **DISP=OLD** defaults to **DISP=(OLD,KEEP)**

- If the DD statement is receiving a passed dataset, which existed before the job began execution, the default is **KEEP**.

```
//S1      EXEC    PGM=PR
//DD1     DD      DISP=(SHR,PASS), DSN=USER1.LONE
//S2      EXEC    PGM=PK, COND=(4,LT)
//DD2     DD              DISP=SHR,DSN=USER1.LONE
```

In **DD2**, **DISP=SHR** defaults to **DISP=(SHR,KEEP)**

Despite the several possible defaults for **DISP=OLD** or **DISP=SHR** their use is extremely common. When not receiving a passed data set, they always safely default to **DISP=(OLD,KEEP)** and **DISP=(SHR,KEEP)**, respectively.

- If the status field is **OLD** or **SHR** and the dataset name is temporary, the default is pass.

```
//DD1   DD DISP=OLD,DSN=&&TEMP
```

DISP=OLD defaults to **DISP=(OLD,PASS)** and the message is displayed as the output –"INVALID DISP FIELD – PASS SUBSTITUTED"

- If the status field is **MOD**, which defaults to an existing data set, **MOD** works the same as **OLD** and **SHR**.
- If the status field is **MOD**, which defaults to **NEW**, the default of the second field is **DELETE**.

The Abnormal (or Conditional) Disposition Field (Contd.):

DISP=MOD can default to **(MOD,KEEP)**, **(MOD,DELETE)**, **(MOD,PASS)**, and **(NEW,DELETE)**. In view of all these possibilities, it is recommended that defaults are not to be practiced with **MOD**.

Remark: The various fields of the DISP parameter stand for the PDS and not the member.

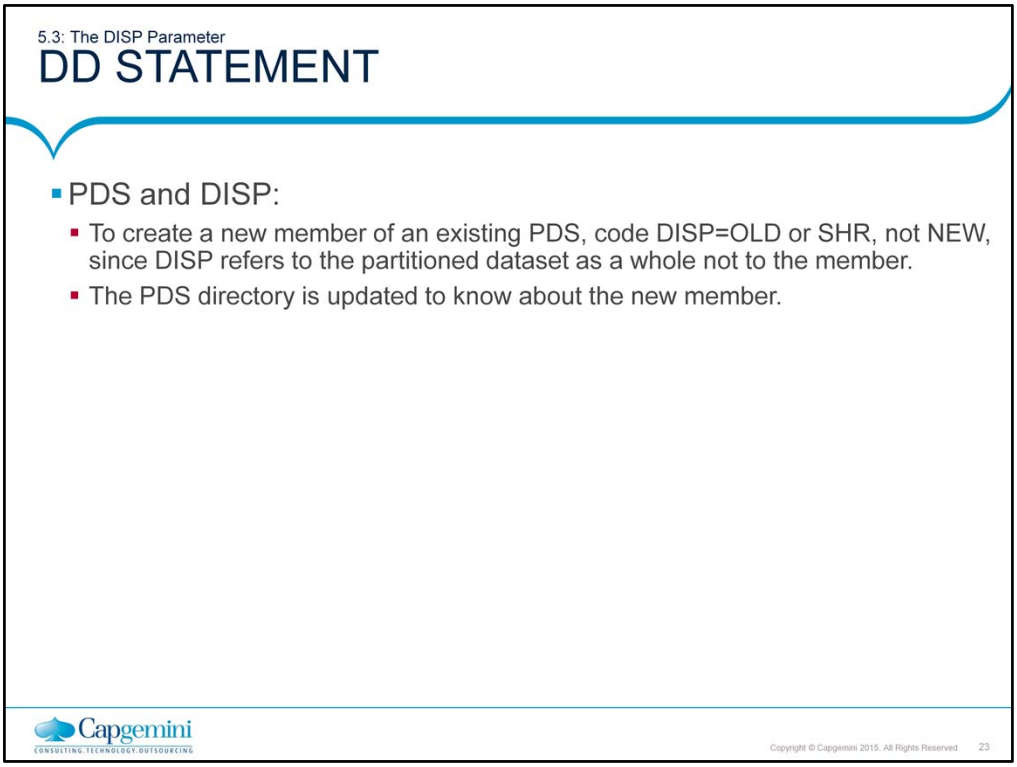
```
//DD1 DD DSN=USER1.LIB2(Z32), DISP=(OLD,DELETE,DELETE)
```

In the above case, the PDS and the member both are deleted. Both the PDS and the member should exist.
How a member is handled depends on whether or not it exists and whether the program opens for input or output. A summary of all possibilities is presented below.

```
//S1        EXEC      PGM=P1
//D1        DD        DSN=DA0001T.LIB(M12),DISP=SHR
```

M12 EXISTS
- In P1, if M12 is opened in I/P mode, for reading, M12 is read.
- In P1, if M12 is opened in O/P mode for writing, M12 is replaced (not in place).

M12 DOES NOT EXIST
- In P1, if M12 is opened in I/P mode for reading, ABEND (S013-18).
- In P1, if M12 is opened in O/P mode for writing; M12 will be created and written into.



If DISP parameters are not coded, the following defaults:

DISP Coded as : Defaults to :

If not coded ……(NEW,DELETE,DELETE)
(OLD)…….....(OLD,KEEP,KEEP)
(SHR)……..(SHR,KEEP,KEEP)
(,CATLG)……..(NEW,CATLG,CATLG)
(MOD)……If dataset does not exist :

(NEW,DELETE,DELETE)

If dataset exists :

(OLD,KEEP,KEEP)
(MOD,CATLG,)……..(NEW/OLD,CATLG,)
DELETE) DELETE)

Disposition of OLD deletes existing records from the dataset. So, to append records, code MOD.

5.3: The DISP Parameter

Normal and Abnormal DISP-Parameters (Contd.)

- If
- M12 EXISTS

```
//S1     EXEC   PGM=P1
//D1     DD     DSN=DA0001T.LIB(M12),
//                     DISP=SHR
```

  - P1 opened in I/P mode for reading, M12 is read.
  - P1 opened in O/P mode for writing, M12 will be replaced (not in place)
- If M12 DOES NOT EXIST
  - P1 opened in I/P mode for reading, ABEND (S013-18).
  - P1 opened in O/P mode for writing, M12 will be created and written into.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 24

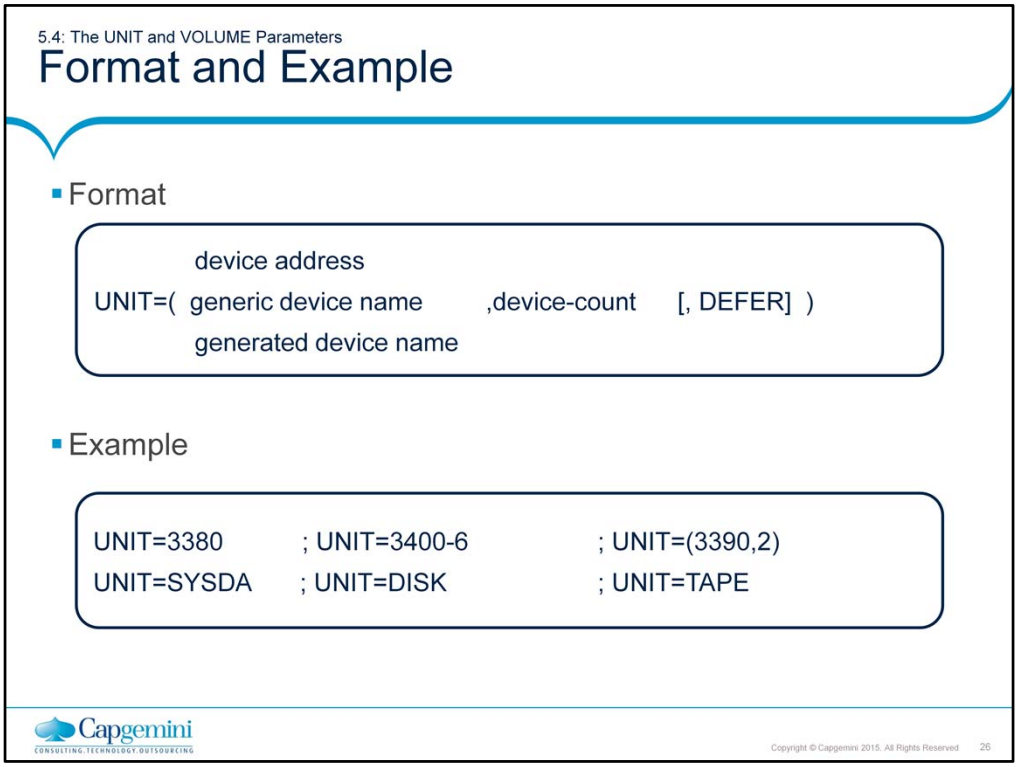
5.4: The UNIT and VOLUME Parameters

Characteristics

- UNIT and VOLUME parameters work together to specify the location of the dataset.
- UNIT indicates the following:
  - The device type or device address on which the dataset resides on a volume
  - The number of devices to be allocated to the dataset
  - The instance when the mount message is to be shown to the operator
- There is no default for UNIT.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 25



Format:

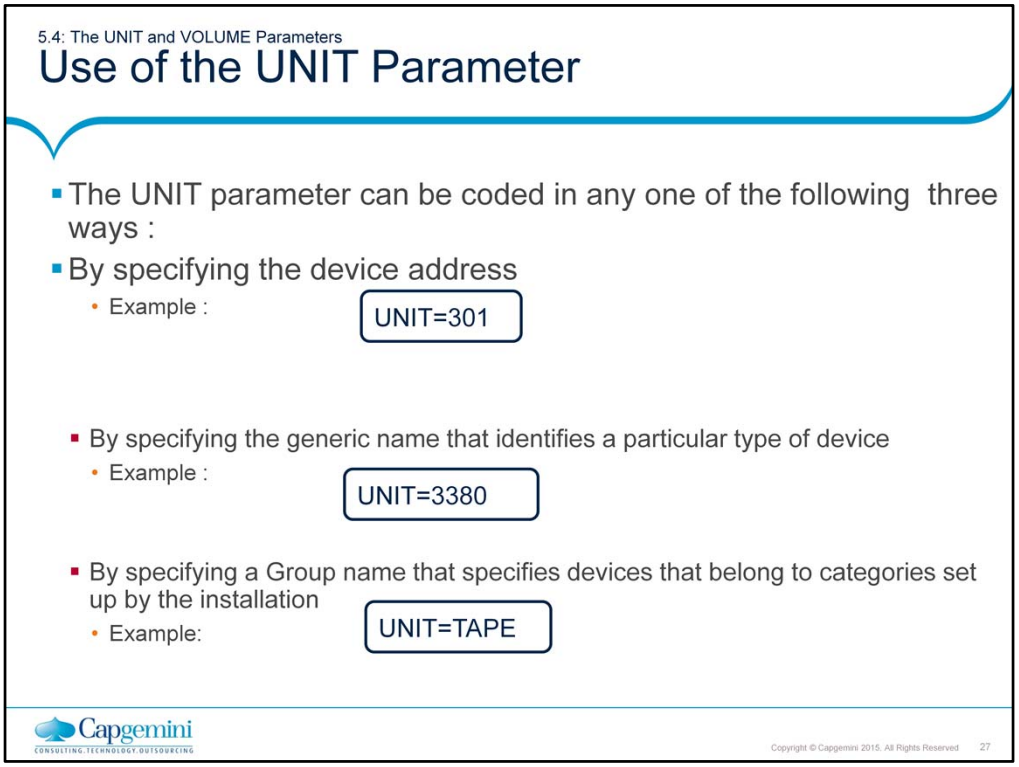
device address: Identifies the exact device address. This notation is almost never used.

generic device name: Identifies the device type using a universal system-supplied name.
For example: UNIT=3390; UNIT=3400-5; UNIT=3480

generated device name: Identifies the device type using an installation-defined name.
For example: UNIT=SYSDA; UNIT=DISK; UNIT=TAPE
The generated names can be made to mean whatever an installation requires them to mean. For example, UNIT=SYSDA can mean all 3380 devices of any density, or single density only, or a subset of double density devices or a combination of 3380 and 3390 device. Their definition can vary from installation to installation.

device count: Specifies the number of devices to be allocated for the dataset. The limit is 59 devices. If omitted, default is 1 except when DD statement describes a disk multi-volume dataset. In such case, device count=number of volumes.

Of the four, the generated device name is used most commonly.



Use of the UNIT Parameter:
(i) For example, UNIT=(SYSDA,5) , UNIT=(TAPE,2)

(ii) For example, 2 UNIT=SYSDA is same as UNIT=(SYSDA,1) because of default.

(iii) For example, 3

```
//DD1    DD  DSN=DA0001T.EMPFILE,DISP=(,CATLG,DELETE),
//           UNIT=SYSDA,VOL=SER=(BS3001,BS3002,BS3003),
//           SPACE=(TRK,(1,2)),DCB=(LRECL=80,RECFM=FB,
//           BLKSIZE=800)
```

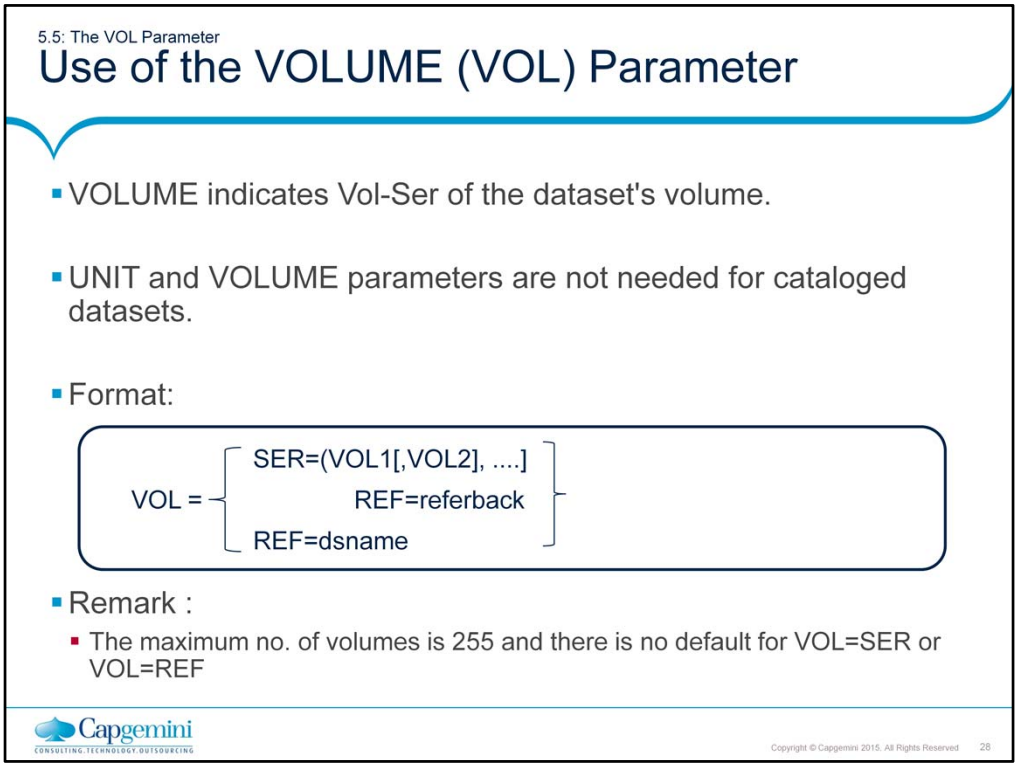
In this example, UNIT =SYSDA defaults to UNIT=(SYSDA,3)

Note: UNIT=(,2) can also be used if the device is being supplied by the catalog.

Default:

There is no default for device name. If it is not coded in the UNIT parameter and it is also not supplied by the catalog, by the Passed dataset Queue, or by the VOL=REF, the result is a JCL error. The message is

"IEF210I JOBNAME STEPNAME DDANAME – UNIT FIELD SPECIFIES INCORRECT DEVICE NAME", which is misleading. It means that the device name was needed but not coded.



Use of the VOLUME (VOL) Parameter:
The main function of the VOL (or VOLUME) is to identify the volume(s) by serial number where an existing dataset resides or where a new dataset will reside.

SER=(vol1,vol2….) – Specifies the serial number(s) of the volume(s) to be used. The maximum number of volumes is 255.

A volume serial is a combination of alphabetic, numeric, and national characters ($ @ #) up to 6. A hyphen is also permitted. In a real (or production) environment, the number of characters is almost never less than 6.

For example.

```
VOL=SER=BS3001 or
VOLUME=SER=BS3001
VOL=SER=(BS3013,BS3014)
```

The VOL parameter must be coded:

(i) While retrieving a dataset, which is neither cataloged nor passed.

(ii) While retrieving a dataset, which is cataloged, but the catalog must not be used.

(iii) While creating a dataset, which must reside on a particular volume.

Use of the VOLUME (VOL) Parameter:

- **REF=referback**

  Referback can have three formats:

  - ***.stepname.ddname**: This requests that the volume is to be the same as the volume for DD statement **ddname** found in the previous step **stepname**

    VOL=REF=*.STEP2.DD1

  - ***.ddname**: This requests that the volume is to be the same as the volume for previous DD statement **ddname** found in the same step **stepname**.

    VOL=REF=*.DD1

  - ***.procexec.stepname.ddname**: This requests that the volume is to be the same as the volume for DD statement **ddname** found in the previous step **stepname** found within a procedure **procexec** (name of the **EXEC** statement invoking the procedure

    VOL=REF=*.PR1.STEP2.DD1

  Referbacks are not encouraged. They are to be used only when they are necessary. A **referback** with a **stepname** causes a JCL error if the referenced step does not execute. Such **referbacks** must be avoided where restart is required.

- **REF=dsname**: This requests that the volume is to be the same as the one where dataset **dsname** resides. The dataset must be cataloged or passed. The dataset does not even have to exist, as long as it is cataloged or passed. The name of the referenced dataset need not appear anywhere else in the job.

  For example, VOL=REF=DA0001T.EMPFILE

  Remark: When **VOL=REF** (**referback** or **dsname**) is used, the system supplies the volume as well as the unit information. Therefore, the **UNIT** parameter is usually unnecessary.

Default:

There is no default for **VOL=SER** or **VOL=REF**. However, if both are omitted, no JCL error results. Instead, the meaning of DD statement changes. For example, while retrieving when **VOL=SER** or **VOL=REF** is coded, the catalog is not used. If neither is coded, the catalog is used.

VOLUME=(v1,v2,v3,v4,v5)

To specify the volume where a dataset is stored
- V1=PRIVATE, requests for private volume for the dataset to reside
- V2=RETAIN, the volume must not be dismounted after dataset is closed at the end of the job step
- V3=VOLUMN SEQ NO, In multi volume dataset it specifies the volume to begin processing with. Volume sequence number values ranges from 1 thru 255
- V4=VOLUMN COUNT, refers to the maximum number of volumes required for output data sets. This is used with tape data sets and maximum volumes allowed for tape data sets is 8
- V5=SER indicates the serial numbers of the tape or disk volume the dataset is on.It takes maximum of 6 characters.

UNIT and VOLUME parameters not needed for cataloged dataset.

```
VOL=SER=(VOL1[,VOL2], ....]
```

While creating New datasets, if UNIT is not coded:
Defaults to installation-defined devices based on purpose:

	Unit	Volume
For Development datasets	3390	ZTSO01
For Testing datasets	3391	ZRSO06

For SMS managed datasets, defaults to STORAGE CLASS:

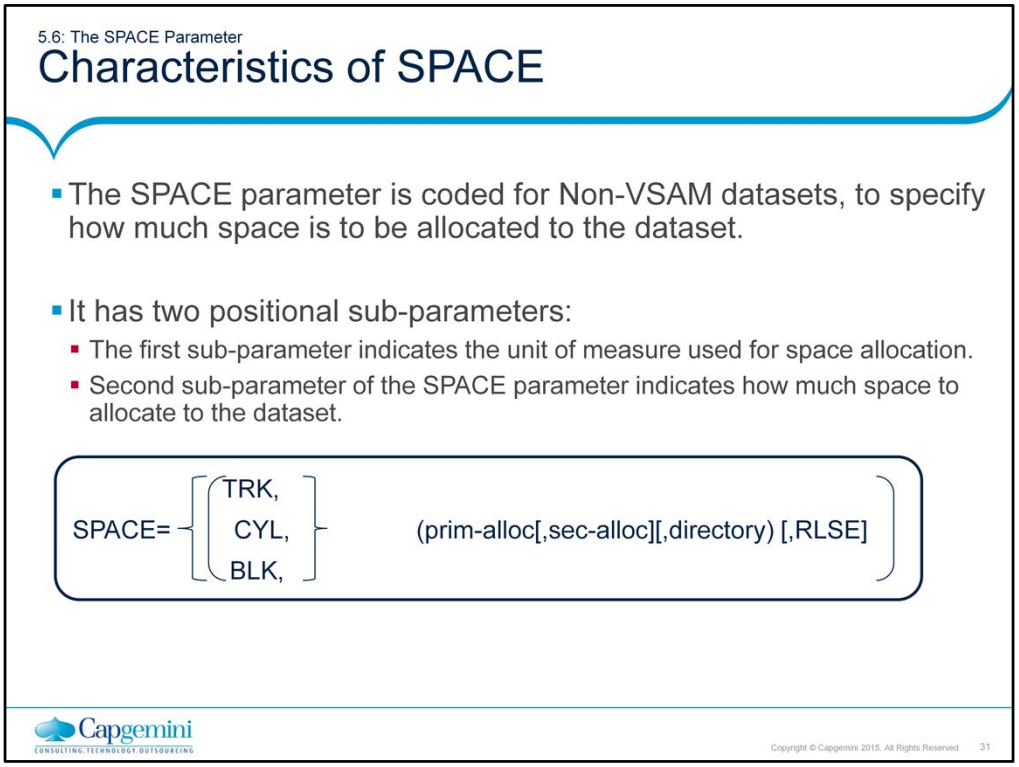
Storage Class	Unit	Volume
SCTSO	3390	ZTSO03

Common problems with UNIT
If you forget to code DISP at the dataset that you are reading,

```
//DDRd    DD    DSN=IGTRN12.OLD.DS
```

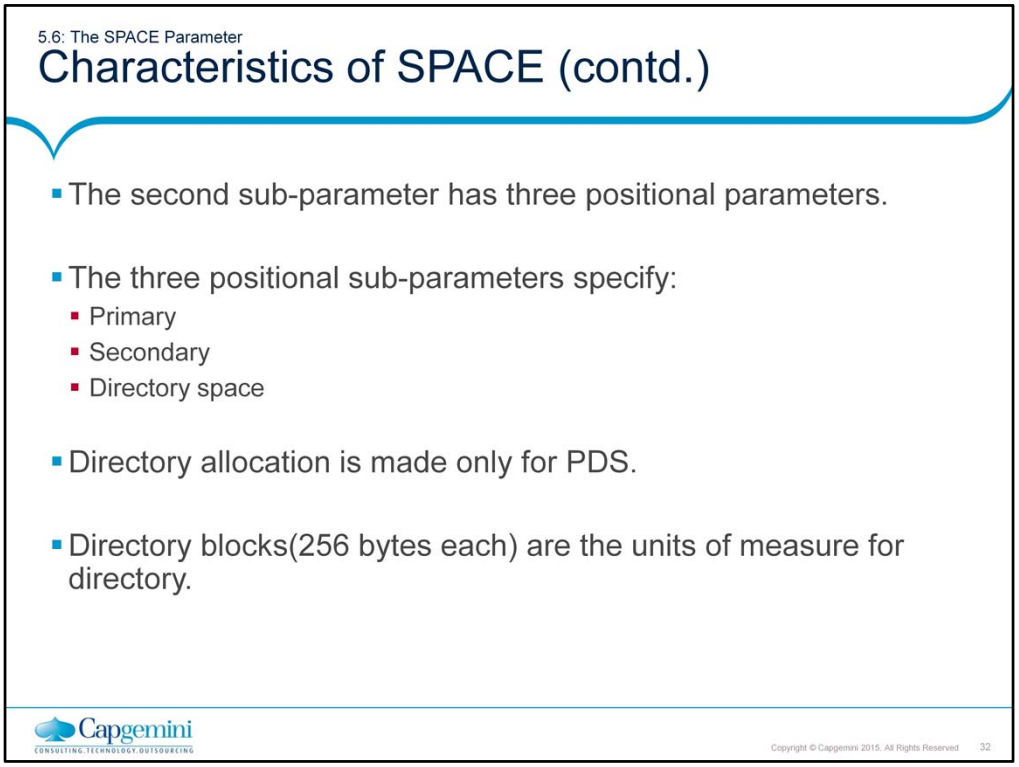
you will get an error message SPACE NOT SPECIFIED FOR DATASET or INCORRECT DEVICE TYPE SPECIFIED.

Correct this problem by coding DISP=SHR that you forgot, not UNIT.
You will get this same error message if you code an incorrect UNIT with a Cataloged dataset.



Characteristics of the SPACE Parameter:
The SPACE parameter must be included in a DD statement at the following instances:
A new disk dataset is created.
An old dataset needs to alter its entitlement to additional space. That is, it needs to request additional disk space for an old dataset when available space is exhausted.
An old disk dataset must make all unused space available.

TRK: This requests that space be allocated in tracks.
CYL: This requests that space be allocated in cylinders.
Blksize: This specifies the average blocksize of the dataset. The system translates it to tracks.



Characteristics of the SPACE Parameter:
Directory: This specifies the number of directory blocks (256 bytes each) to be assigned to the directory of a PDS.

The directory quantity, if not coded, defaults to zero; therefore, the directory quantity must be specified for a new PDS. If it is so, S013-14 ABEND failure does not occur if an attempt is made to add the first member to a PDS.

This identifies the number of tracks (if TRK is coded) or cylinders (if CYL is coded) or the number of blocks (if blksize is coded) that must be allocated during the allocation process for a new dataset before the step begins execution. The system allocates the requested space in one extent. If this is not possible (and CONTIG is not coded), two extents are used, then three and so on up to five extents. If as many as five extents cannot satisfy the request, the result is the following allocation JCL error:

IEF257I jobname stepname ddname –SPACE REQUESTED NOT AVAILABLE.

If the request is nonspecific (no VOL=SER or VOL=REF), needing a storage volume, the JCL error message will be different:

IEF257I jobname stepname ddname –INSUFFICIENT SPACE ON STORAGE VOLUMES.

Remark: The system always allocates the primary quantity in the least number of extents possible on a single volume. The primary quantity cannot be split over multiple volumes. The primary allocation cannot be omitted (coding 0 is allowed). It is ignored if the dataset is old.

For example, SPACE=(TRK,3)
For example, SPACE=(CYL,4)
For example, SPACE=(23440,100)
For example, SPACE=(TRK,0)
The primary allocation cannot be omitted (coding 0 is allowed). It is ignored if the dataset is old.

Characteristics of the SPACE Parameter (Contd.):

- **sec-alloc - Secondary allocation or secondary quantity**:

This identifies the number of tracks (if **TRK** is coded) or cylinders (if **CYL** is coded) or the number of blocks (if **blksize** is coded) that are to be allocated when all the available space is exhausted while writing to a dataset. The system allocates the secondary quantity in the least number of extents possible, and just like the primary quantity; it can be given in as many as five extents, if necessary.

The system always supplies the specified secondary allocation when one is needed unless one of the two events occurs:

- The allocated volume does not have enough space to satisfy the secondary allocation and no other volumes are allocated.
- The needed secondary allocation, if granted, causes the dataset to exceed 16 extents on the volumes and no other volumes are allocated.

If either of these two conditions arises, the result is a **SB37-04** ABEND failure (normally for a sequential dataset). For a PDS, the ABEND can also be **SE37-04**.

Please note that a PDS is confined to a single volume, while a sequential dataset can extend into a maximum of 59 volumes. The 16-extent-per-volume limit for a dataset is system-supplied and cannot be altered.

The secondary allocation is optional. If omitted, defaults to 0. When no secondary allocation is coded and the primary allocation is exhausted, the result is an SD37-04 ABEND failure.

Characteristics of the SPACE Parameter (Contd.):

- **sec-alloc - Secondary allocation or secondary quantity (contd.)**:
  Remark: The directory quantity is taken away from the beginning of the primary allocation if **TRK** or **CYL** is coded in the **SPACE** parameter. When **blksize** is coded, the system adds the directory blocks to the data blocks and then computes the amount of primary space.

  For example, SPACE=(TRK,(20,5,5)) OR SPACE=(TRK,(20,,5))

  If no secondary

  For example, SPACE =(CYL,(20,5,5)) OR SPACE =(CYL,(20,,5))
  if no secondary

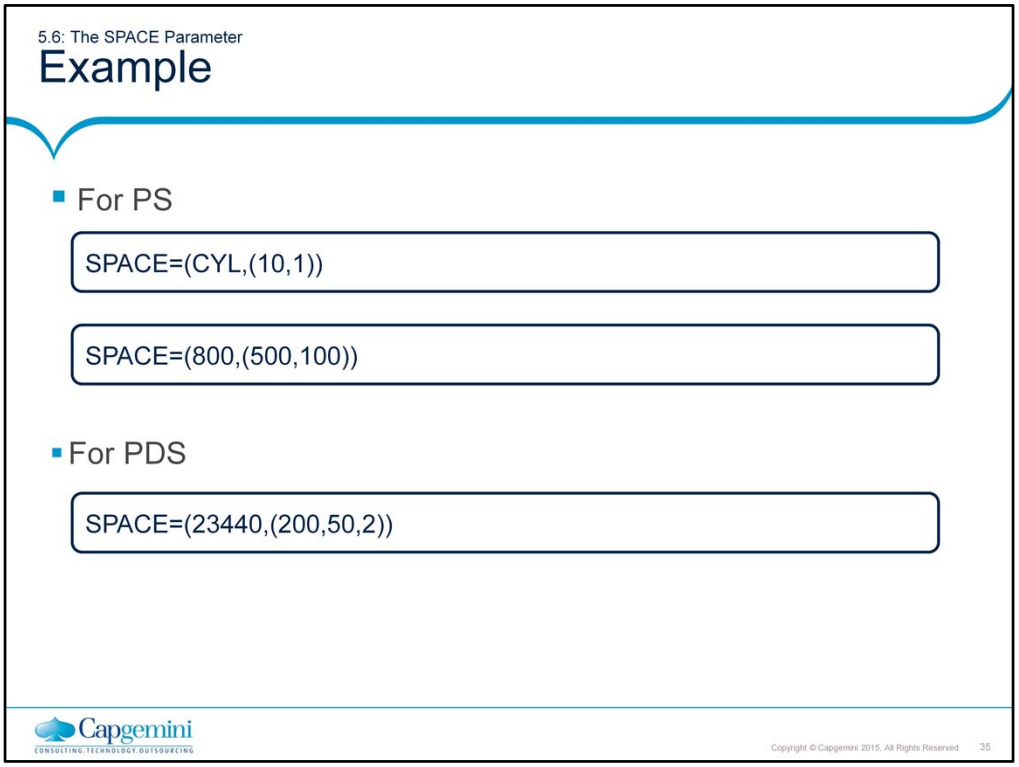
  For example,

  SPACE =(23440,(200,50,5)) OR SPACE =(23440,(200,,5))
  if no secondary

- **RLSE:** This requests that any unused space is to be freed when the dataset is closed. This works for both new and old datasets, provided they were opened for output. Space is released on the boundary used in the **SPACE** parameter. If tracks (or cylinders) are allocated, unused tracks (or cylinders), are released.

  Remark: Using **RLSE** is highly recommended for datasets not intended for future expansions. Temporary datasets are ideal candidates. For datasets that expand in future runs, **RLSE** can result in a larger number of extents, and, possibly, a premature SB37-04 ABEND failure. RLSE will be ignored if the dataset is opened by another user (or shared by another job) or the step ABEND's.

  For example, SPACE=(TRK,(5,1),RLSE)



SPACE PARAMETER
Specifies how much space to allocate for a new disk (DASD) dataset.
Syntax:

```
SPACE=(S1,(S2,S3,S4),S5)
```

S1 - the Unit of space in Cylinders (CYL), Tracks (TRK), Blocks of records (a number representing block size), Record length
S2 – Primary (initial) quantity of units
S3 - Secondary quantity of units if primary is exceeded; but allocated only when the dataset expands.
(S3 X 15 or 16 or 123 secondary allocations - called Extents)
S4 - Number of Directory blocks (applicable for PDS only)
S5 - Releases the unused space requested

EXAMPLE:
SPACE=(TRK,(5,2)) You get 5 tracks + (2 X 15) tracks = 35 tracks

Disk surfaces are divided into Tracks of recording space
Group of tracks make a Cylinder
Many cylinders make a disk Volume
Different models of IBM Mainframe disks have different track capacities, number of cylinders and device capacities.

Disk Volume Model	Bytes/Track	Tracks/Cylinder	Total Cylinders	Total Bytes
3390-1	56664	15	1113	0.946 Gb
3390-2	56664	15	2226	1.892 Gb

Space (EXTENTS)

The smallest unit of space on mainframe disks is one whole track. The OS cannot split tracks for use by other datasets

One or more contiguous tracks or cylinders allocated to a dataset are called an EXTENT

An EXTENT is a piece of disk space.
Your primary space allocation is 1 extent.
When you request SPACE=(TRK,(5,2)) you get:
1 primary extent and 2*15 secondary extents tracks

Space (Block size)

You can specify space request by Block size instead of Tracks or Cylinders.

EXAMPLE:
SPACE=(3840,200) You get 200 blocks each holding
3840 bytes = 768,000 bytes on the disk
If you code a secondary allocation:
SPACE=(3840,(200,60)) …plus (60 X 3840 bytes X 15
Extents) as secondary allocation

Since one track is the minimum unit of space, if block size requested works out smaller than one track, you still get one track of disk space.

SPACE (Record Length)

Applicable only for SMS managed datasets
Code AVGREC parameter along with Space parameter
Do not code BLKSIZE in the DCB keyword parameter\

```
//DDN          DD          DSN=DSRC012.DS,
//                          DISP=(NEW,CATLG,DELETE),
//                          UNIT=SYSDA,
//                          AVGREC=U,
//                          SPACE=(133,10000)
//                          RECFM=FB,
//                           LRECL=133
```

With AVGREC=U:
The 133 in Space parameter is the average Record length in bytes
=U means that the 10000 is a “unit” estimate of the quantity of records to be written.

Example

```
//DDN     DD     DSN=DSRC012.DS,
//             DISP=(NEW,CATLG,DELETE),
//             UNIT=SYSDA,
//             AVGREC=U,
//             SPACE=(133,10)
```

You get (10 records X 133 bytes) of space.
For Secondary allocation request: SPACE=(133,(10,2))
AVGREC can support record quantities by thousands (AVGREC=K) and by millions (AVGREC=M)

```
//             AVGREC=M,
//             SPACE=(133,10)
```

You get (10,000,000 records X 133 bytes) of space.

SPACE (Directory Blocks)

While creating a PDS, the SPACE parameter is to include a request for Directory Blocks

Space=(TRK,(1,2,7))	You get 1 + (2 X 15) = 31 tracks and create 7 Directory blocks. Each directory block can store information about 5 members; implies 35 members can be allocated.
Space=(TRK,(1,,7))	When you do not want secondary space for the PDS.

EXAMPLES:

Space=(TRK,(5,,1))	Allocates a PDS with no secondary space and 1 directory block.
Space=(3840,(200,60))	Allocates (200 X 3840) + (60 X 3840 X 15) of disk space.

```
//          //DDN  DD  DSN=……..,
//          DISP=…..,
//          UNIT=3390,
//          Space=(TRK,10)
```

5.6: The SPACE Parameter

Characteristics

- When the directory is exhausted, the result is an S013-14 ABEND failure.
- The directory quantity, if not coded defaults to zero, that is, a sequential dataset.
- When the secondary space is exhausted, the result is SB37-04 ABEND failure for a sequential dataset and SE37-04 ABEND failure for a PDS.
- There is no default for primary allocation (coding 0 is allowed) and secondary and directory space if omitted, defaults to zero.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 38

You receive the secondary space requested for your dataset only as the dataset expands as you use it .You can get up to 16 extents of secondary space per dataset per disk volume for a PS If you use more than one disk volume for a PS dataset, it can spread up to 16 secondary extents on each volume A PDS cannot span disk volumes The limit of 16 extents per dataset per disk volume does not apply to VSAM datasets, which can exist with up to 123 extents

SPACE (RLSE)
Causes all of the whole unused tracks to be freed for use by other datasets

Syntax:

SPACE=(6200,300,RLSE)	Releases unused tracks.
SPACE=(6200,(300,60),RLSE)	Releases unused tracks in the last active secondary.
SPACE=(80,1,RLSE)	No space is released since space is less than a track for any disk volume model.

RLSE does not work if a step abends
Does not work if the dataset is not closed normally
Does not work when the dataset is shared

5.7: The LABEL Parameter

Characteristics

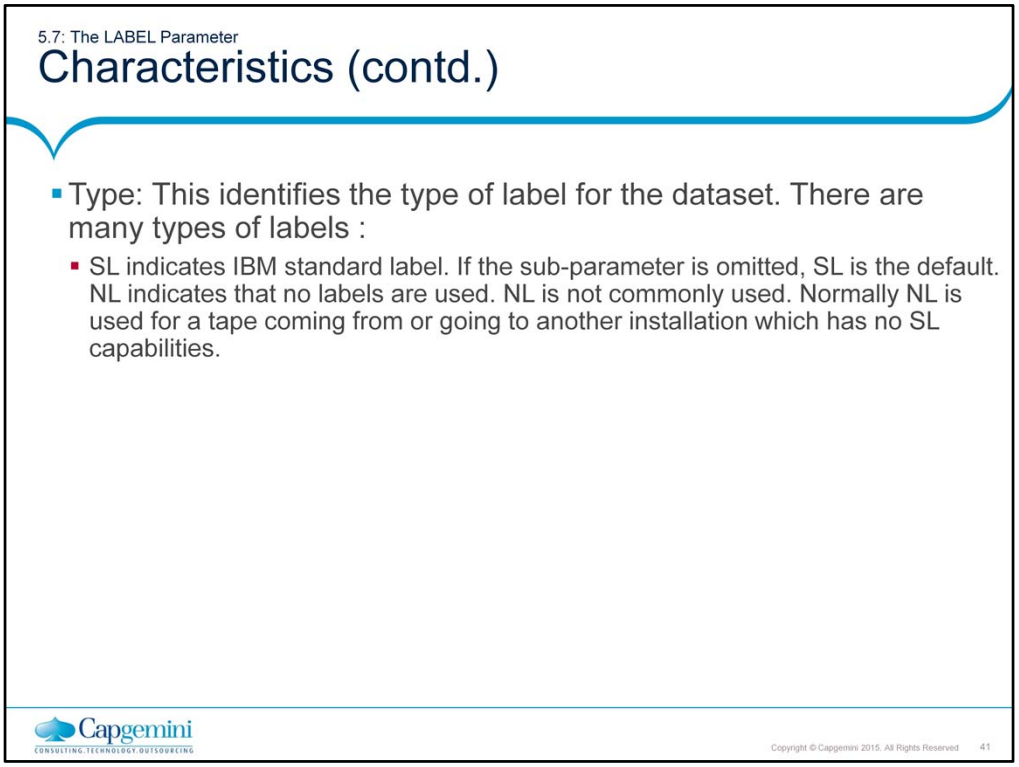
- The LABEL parameter can specify the following:
  - The sequence of a tape dataset on a volume
  - The type of label of the dataset
- Format :

LABEL=([seq-no][, type])

  - seq-no: This identifies the sequence number of the dataset on a tape volume. It can have 1 to 4 digits. If omitted, it defaults to 1. If 0 is coded, it defaults to 1. maximum: 9999
  - seq-no (Contd.): The sequence number is ignored in the following situations:
    - Retrieving a tape dataset through the catalog
    - Receiving a passed tape dataset

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 40



Characteristics of LABEL:
There are many types of labels. To name a few, which are important from project perspective.

SL: This indicates IBM standard label. If the sub parameter is omitted, SL is the default.
NL: This indicates no labels are used. NL is not commonly used. Normally, NL is used for a tape coming from or going to another installation, which has no SL capabilities.
BLP or Bypass Label Processing: This indicates that labels are not to be recognized and are to be treated as ordinary files. BLP is used as a last resort when neither SL nor NL can accomplish what is required.

Label Verification: While retrieving an SL tape dataset, both the volume serial and the dataset name are verified. While creating an SL tape dataset with VOL=SER or VOL=REF, only the volume serial is verified.

While retrieving an NL tape dataset, neither the volume serial nor dataset name can be verified. However, only an NL tape volume can be mounted. An SL volume is rejected.

5.7: The LABEL Parameter

Examples

```
LABEL=(2, SL)
```

```
LABEL=(, NL)
```

- Defaults : If omitted, the LABEL parameter defaults to (1, SL).
- There are four ways to supply the same information:
  - Omit the label parameter
  - Code LABEL=(1, SL)
  - Code LABEL=(, SL) 1 is the default
  - Code LABEL=1 SL is the default

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 42

5.7: The LABEL Parameter

Examples (contd.)

- Creating labeled tape dataset:

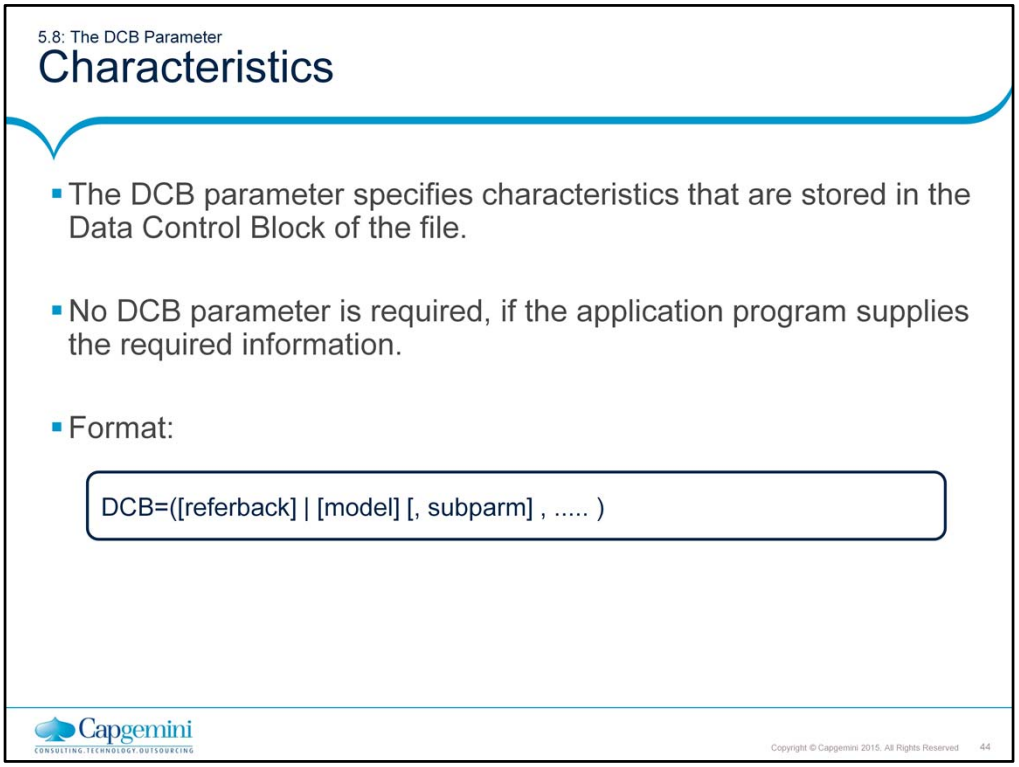
```
//OUT     DD      DSN=DA0001T.TAPE,
//              DISP=(, CATLG), UNIT=TAPE,
//              DCB=(BLKSIZE=32720, LRECL=80,
//              RECFM=FB)
//* LABEL not supplied - the default is (1, SL)
```

- Creating non-labeled tape dataset:

```
//OUT     DD      DSN=DA0001T.TAPE,
//              DISP=(, KEEP), UNIT=TAPE, LABEL=(, NL),
//              DCB=(BLKSIZE=32720, LRECL=80,
//              RECFM=FB)
//* LABELs are not used
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

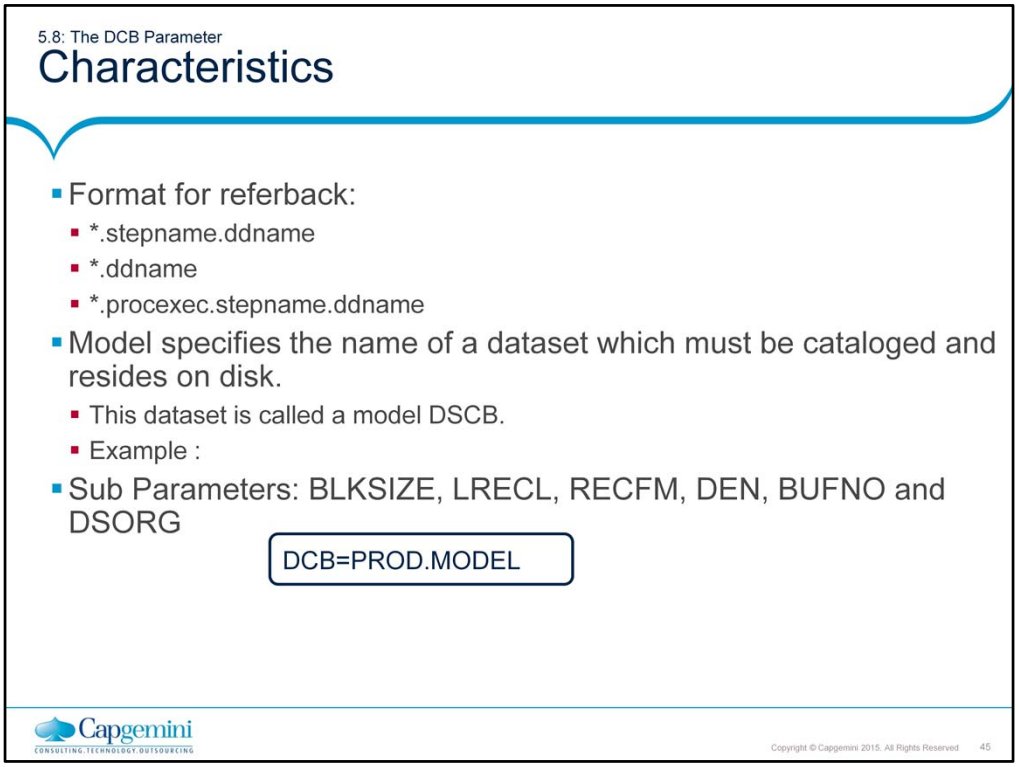
 43



Characteristics of DCB:

The DCB parameter specifies values to be used to complete the Data Control Block (DCB) when a dataset is opened. A DCB is constructed by the language processor (compiler or assembler), based on the appropriate instructions of the language being used, and resides inside the code of the program. The compiler collects this information and defaults from various parts of the program (For example, in COBOL, RECORD CONTAINS 80 CHARACTERS; BLOCK CONTAINS 10 RECORDS and so on) and constructs the DCB. Note that the DCB exists only for non-VSAM datasets and is checked by the OPEN routines (for input or output). Certain values must be “hard-coded” in the DCB by the program. Others can be left out, giving the user the option of supplying these values via the DCB parameter (as well as other means).

There are three suppliers of DCB information:
Values supplied by the program, referred to as hard-coded. When a value is hard-coded, it cannot be changed unless the program is changed.
Values coded in the DCB parameter of the DD statement. These values are ignored if they are already hard-coded.
Values from the standard label of the dataset. The values supplied by the label are limited to: BLKSIZE, LRECL, RECFM, DSORG, and so on. Values from the label are not used if they are hard-coded inside the program or coded in the DCB parameter of the DD statement.



Characteristics of DCB:

DCB=([referback] | [model][,subparameter],…… keyword parameter

Referback: This can have three formats:
*.stepname.ddname: This requests that the DCB parameter is to be copied from the DD statement “ddname” found in the previous step “stepname”.

DCB=*.STEP2.DD1

*.ddname: This requests that the DCB parameter is to be copied from a previous DD statement “ddname” found in the same step “stepname”.

DCB=*.DD1

*.procexec.stepname.ddname: This requests that the DCB parameter is to be copied from DD statement “ddname” found in the previous step “stepname” found within a procedure “procexec” (name of EXEC statement invoking the procedure).

DCB=*.PR1.STEP2.DD1

Remark: The DCB referback copies the DCB parameter as opposed to the DSN and VOL=REF referbacks which acquire the dataset name and the VOL=SER respectively, whether or not the DSN and VOL parameters are present in the referenced DD statement. If the DCB referback refers to a DD statement, which contains no DCB, nothing is copied and no message appears.

Sub-parameters: Mnemonic Hint:BLaBbeReDD (B: BLKSIZE, L: LRECL, B: BUFNO, R:RECFM, D:DEN, and D:DSORG).

Models:

This specifies the name of the dataset which has following characteristics:

- It must be cataloged. If it is not, the result is a JCL error:
  IEF2121 jobname stepname ddname –DATASET NOT FOUND
- It must be on disk (Tapes not allowed).
- It must reside on a volume that is accessible (online).

This dataset is called a model DSCB. The DCB information from the label of the model is extracted and can be used.

For example, DCB=DA0001T.EMPFILE

For example, In case you want to override some of the subparameters, the overriding subparameters must follow the DCSB model dataset name.
DCB=(DA0001T.EMPFILE,LRECL=100,BLKSIZE=800)

Models are generally used, during the creations of GDGs and dummying the PDS.

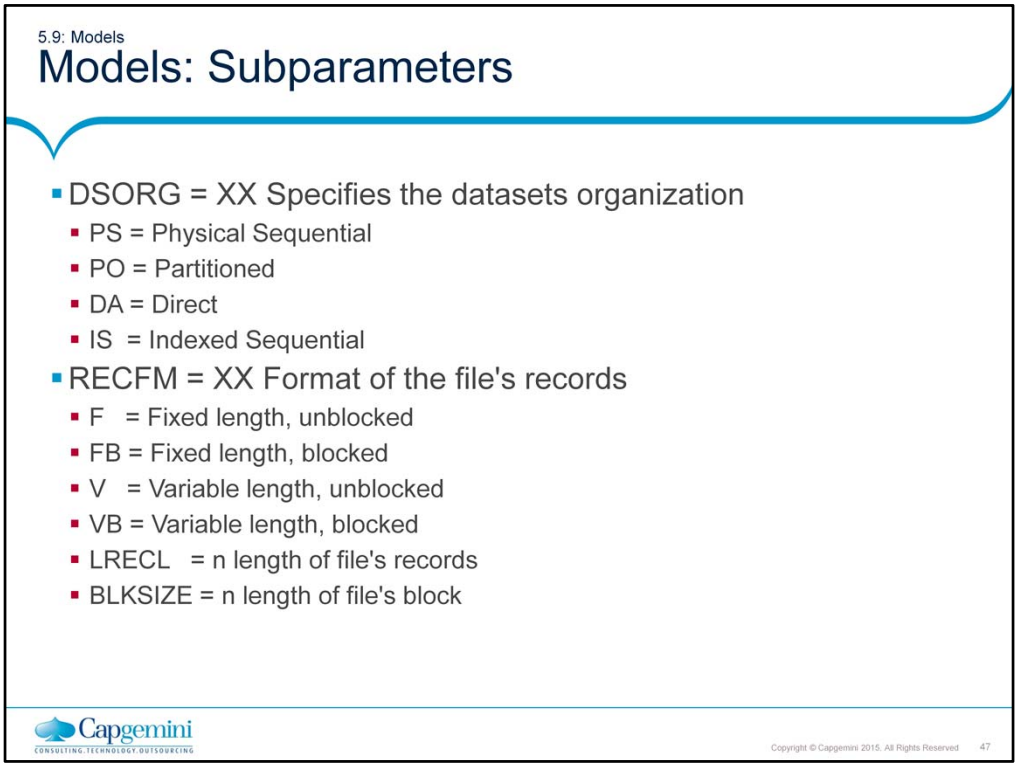
Subparameters: There is a vast number of subparameters, the great majority of which are seldom or never used.

- **BLKSIZE**: This specifies the size of the block (also known as the physical record). For **RECFM=FB**, the blocksize must be multiple of the logical record length, and it identifies the exact size of the block. For **RECFM=VB**, the blocksize can be any value up to the limit but atleast 4 bytes larger than the logical record length. For **RECFM=U**, the blocksize can be any value up to the limit.

  - **Remark:** There is no default for **BLKSIZE**. Coding **BLKSIZE=0**, the system computes the optimum blocksize based on the device type.
    For exampl
  e, DCB=BLKSIZE=800

- **LRECL**: This specifies the size of the logical record. The maximum size is 32,760, and it cannot be larger than blocksize, unless **RECFM=VBS** is used.

  For example, DCB=(LRECL=80,BLKSIZE=800)



Models: Subparameters (Contd.):
RECFM: Specifies the record format. There are several values (or combinations of values) that can be coded:

F: This indicates that all blocks and all logical records are fixed in size.
V: This indicates that blocks as well as logical records are of variable size. The first four bytes of each block (and logical record) describe its length.
B: This indicates that one or more logical records reside in each block. B cannot be coded alone. It is used in conjunction with F or V. For example FB or VB.
U: This indicates that blocks are of variable size. There are no logical records. Mainly used with Load Library.
S: For fixed-size records, this indicates that no short blocks are permitted anywhere but the end of the data. For variable-size records, it indicates that a logical record can span more than one block. S cannot be coded alone. It must follow F, V, FB or VB.
A: This indicates that the first character of each record is an ANSI control character to be used for printer carriage control. A cannot be coded alone. It must follow F, V, FB, VB or U.

For example, DCB=(LRECL=80,RECFM=FB,BLKSIZE=800)

Models: Subparameters (Contd.):

If **RECFM** is not supplied through any means, **U** is the default.

- **DEN:** This identifies the density of the tape. **DEN=3(or 4)** indicates 1600 (or 6250) BPI density.
- **BUFNO:** This identifies the number of buffers to be allocated in virtual storage by the **OPEN** routines, which will contain the blocks to be read in or written out. If omitted, default is 5. The maximum is 255. Coding for **BUFNO** a number greater than 5 may require that the **REGION** parameter be increased. However, default of 5 is more than adequate in most cases of dataset processing.

```
For example,
//INFILE     DD DSN=DA00011.EMPFILE,DISP=SHR,DCB=BUFNO=8
```

- **DSORG**: This identifies the organization of the datasets.
  - **PS:** This specifies physical sequential organization. Mostly **QSAM** and sometimes **BSAM**.
  - **PO:** This specifies partitioned organization (or **BPAM**).
  - **DA:** This specifies direct organization (or **BDAM**).
  - **IS**: This specifies indexed sequential organization (or **ISAM**).

It is important to understand which of these often-used parameters are normally hard-coded and which are not:

- **BLKSIZE**: This is seldom hard-coded. The **BLKSIZE** is unrelated to the logic of the program and hard-coding its value would cause unnecessary changes whenever the **BLKSIZE** is changed. In COBOL, **BLOCK CONTAINS 0 RECORDS** must be coded to avoid hard-coding the **BLKSIZE**. Omitting this clause causes a default of 1 to be used. The result is a hard-coded **BLKSIZE** equal to **LRECL**. Many installation standards disallow hard-coding the **BLKSIZE** for sequential and partitioned datasets.
- **LRECL**: This is frequently hard-coded. The logic of any ordinary program is dependent on the **LRECL** and, as a result, the **LRECL** cannot be changed without changing the logic of the program. Many high-level languages such as COBOL always hard-code the **LRECL**.
- **RECFM**: This is frequently hard-coded. The logic of any ordinary program is dependent on the **RECFM** and, as a result, the **RECFM** cannot be changed without changing the logic of the program. Many high-level languages like COBOL always hard-code the **RECFM.**

5.9: Models

DD STATEMENT

- LRECL, indicates the logical record length of dataset in bytes
  - For fixed length records, the dataset contains records all of the same length and hence LRECL = the actual length of the data-bytes.
    - LRECL for fixed length records can range from 1 to 32,760 bytes. You can access all 32,760 bytes.
  - For variable length records, the length of the records vary from an average to a maximum. So, the LRECL = longest data-bytes length + 4 bytes; the 4 bytes called the RDW(record descriptor word) is pre-pended to each record and indicates the actual length of the record in its first two bytes
    - LRECL can be as high as 32,760 but only 32,752 are accessible.
  - For Undefined format, code LRECL=0. Conventionally used to store load modules.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 49

5.9: Models

DD STATEMENT

- BLKSIZE, specifies the maximum length of a block. The value must be the multiple of LRECL and 4 bytes greater than longest record for variable length record.
  - For fixed length blocked records (RECFM=FB)
    - Block size is multiples of LRECL.
    - Must not be larger than 32,760 bytes.
    - Example: DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000)
  - For variable length blocked records (RECFM=VB)
    - Block size is multiples of LRECL plus 4 bytes, for Block descriptor word (BDW) which is pre-pended to the block of records. The first two bytes of the BDW indicates the length of the block.
    - Example: DCB=(RECFM=VB,LRECL=84,BLKSIZE=8404).
  - For Undefined format (RECFM=U) records, let the OS determine the block size.
  - DCB=(RECFM=U,LRECL=0)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 50

5.9: Models

Models: Example

- Example

```
DCB = (DSORG = PS, RECFM = FB, LRECL = 80,
                                BLKSIZE = 800)
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 51

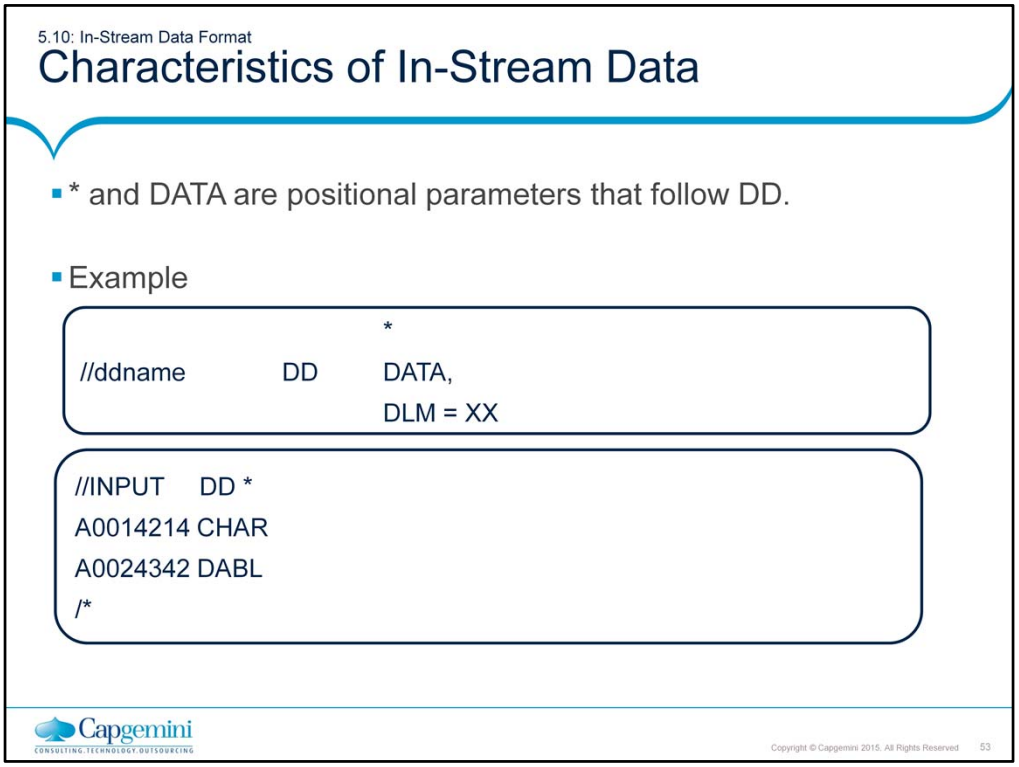
5.9: Models

Models: Remarks

- A DCB exists for every non-VSAM dataset to be opened by the program (for input or output). Certain values must be hard-coded by the program itself. Others can be left out, giving the user the option of supplying these values via the DCB parameter as well as by other means.
- ABEND failures while supplying inconsistent DCB values:
  - S013-20 ABEND when RECFM=FB is used but the LRECL is not an exact multiple of the block size.
  - S013-34 ABEND when RECFM=FB is used and the LRECL is greater than the BLKSIZE.
  - S013-34 ABEND when RECFM=VB is used and the LRECL is greater than the BLKSIZE-4.
  - S001-04 ABEND when BLKSIZE in the DCB parameter is smaller than the actual blocksize and a multiple of the LRECL in the DSCB of the dataset.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 52

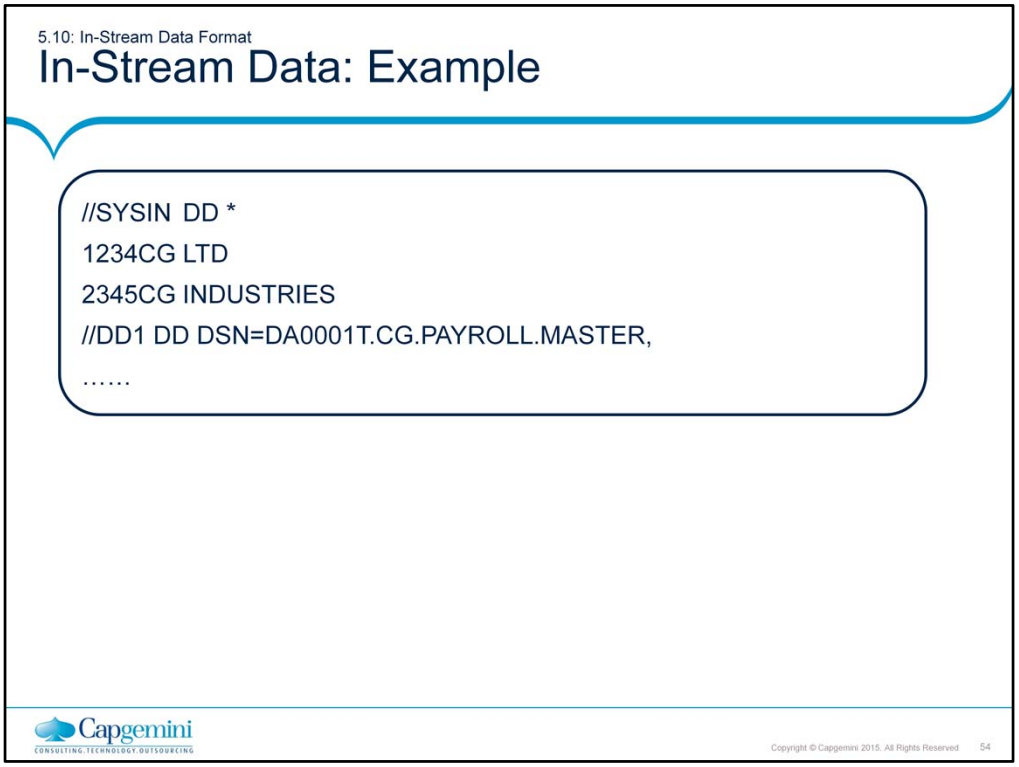


Characteristics of In-Stream Data:
The input stream submitted to the system for execution consists of two possible parts:
JCL: The mandatory part of the input stream
Data mixed in with JCL in the input stream: This data is known as SYSIN data or input stream data. It is an optional part of the input stream and always has a logical record length of 80. Any records encountered in the input stream which are not JCL statements are treated as the SYSIN data.

SYSIN data must be preceded by a DD statement such as follows:
SYSIN data encountered by JES2 or JES3 following a DD * statement saved on the SPOOL volume for future use. This is known as input spooling. The SYSIN is delimited (the spooling stops) by the following:
A /* (delimiter) statement found
A valid JCL statement
An end-of-file condition on reading device

```
//DDNAME      DD  *
  data
/*
```

SYSIN data must be preceded by a DD statement such as follows:
SYSIN data encountered by JES2 or JES3 following a DD * statement saved on the SPOOL volume for future use. This is known as input spooling. The SYSIN is delimited (the spooling stops) by the following:
A /* (delimiter) statement found
A valid JCL statement
An end-of-file condition on reading device



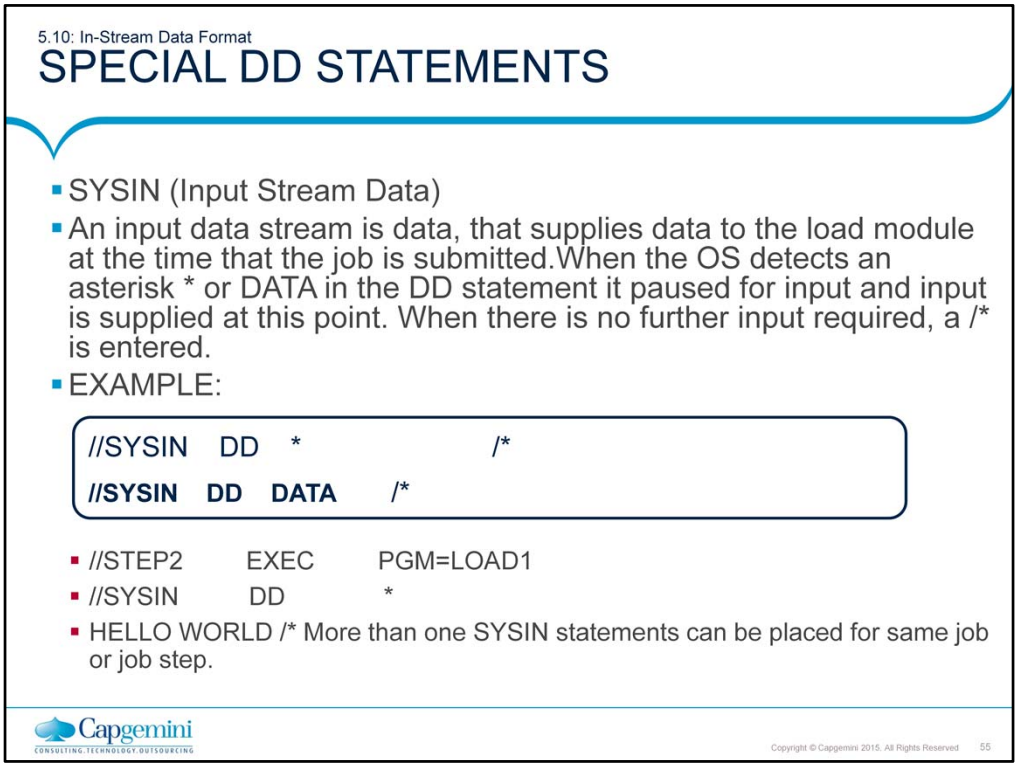
In-Stream Data: Example
The asterisk (*) is a positional parameter. The DD* is a special statement which is under complete JES2 or JES3 control.

SYSIN is a very common ddname used by many vendor-written programs to pass control information to the utility. For example, SORT, IEBGENER, and IDCAMS utilities.

In user written programs, if you use COBOL ACCEPT statement, then in the run JCL one of the DD statements is SYSIN DD statement.

```
//SYSIN   DD *
1234
/*
```

Remark: If SYSIN data is not preceded by DD*, the system generates a statement and place it in front of the SYSIN data.



Instream data to a Cobol program:

In the JCL

```
//STEP010     EXEC     PGM=LM of a Cobol program
      //SYSIN          DD         *
      001 JOHN …..
      /*
```

In the Cobol program:

```
IDENTIFICATION DIVISION.
DATA DIVISION.
WORKING-STORGAE SECTION.
77 DATA1-IN PIC X(80).
PROCEDURE DIVISION.
      ACCEPT  DATA1-IN
```

In-Stream Data: Example

Note: A line with blanks is the most common offender. It is invisible to the user but it will be treated as data by the system this may or may not cause problem. Let us look at the following example.
The system will interpret the above JCL in the following way:
Conclusion: If there are two or more DD statements by the same name in the same step, this is not an error condition. When the program opens for SYSIN the first of the two be used. The other will be allocated and ignored.

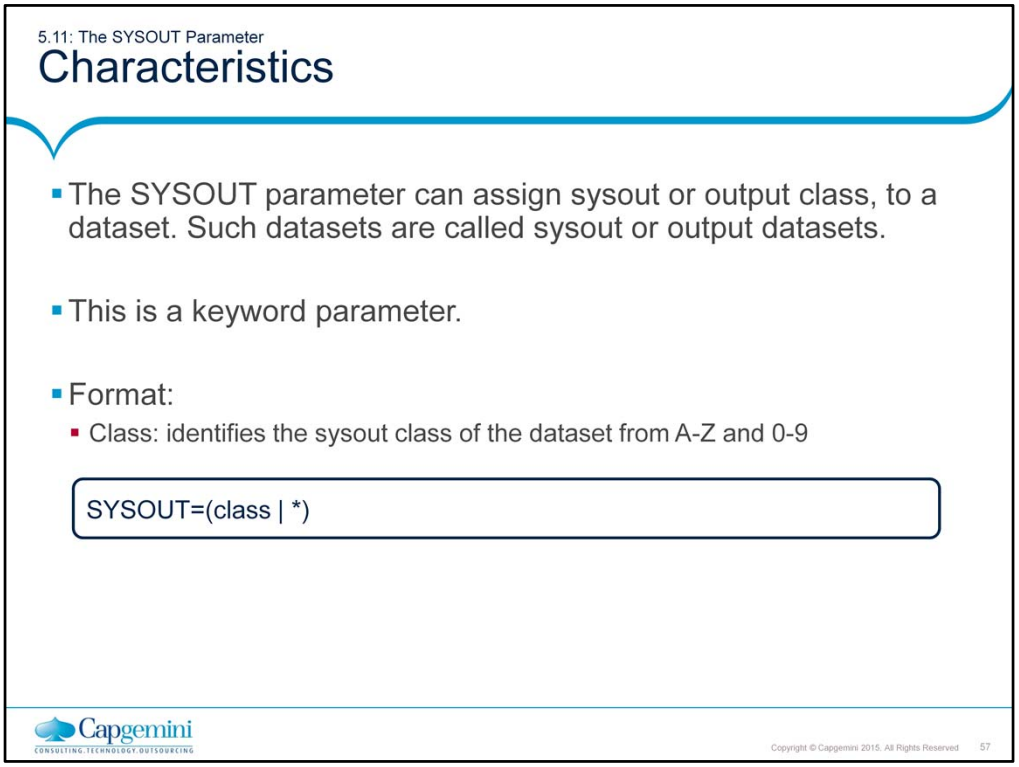
```
//DA0001TA JOB LA2719,…..
//S1        EXEC PGM=ASS1
//STEPLIB   DD …
1234
//DD1       DD …
```

```
//DA0001TA JOB LA2719,…..
//S1            EXEC PGM=ASS1
//STEPLIB   DD …
//SYSIN       DD  *     (generated statement)
1234
//DD1         DD …
```

```
//DA0001TA JOB LA2719,…..
//S1    EXEC PGM=ASS1
//STEPLIB   DD …
//SYSIN  DD *
1234
//DD1  DD …
```

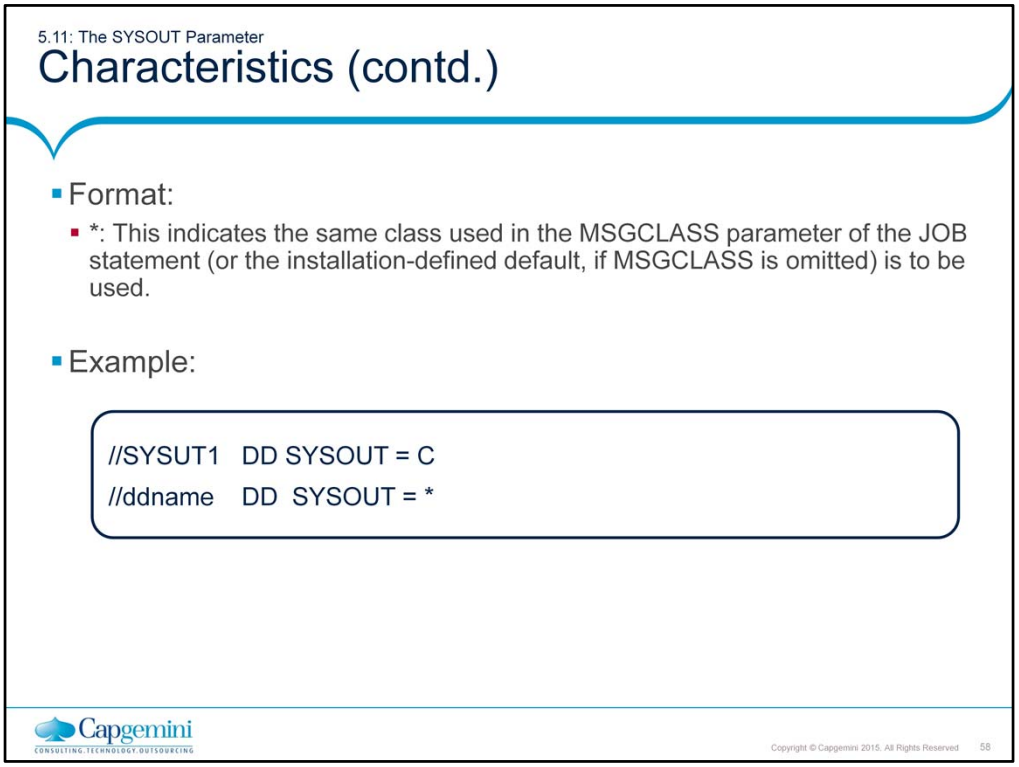
```
//DA0001TA JOB LA2719,…..
//S1    EXEC PGM=ASS1
//SYSIN       DD *

//STEPLIB   DD …
//SYSIN  DD *     (generated statement)
1234
//DD1  DD …
```



Characteristics:
Print records generated by a program are not normally routed directly to a physical printer (theoretically it is possible; but in practice, it is seldom done). Instead, they are written on the SPOOL pack, and saved there for later viewing on a terminal or printing (or both). This is called output spooling, and is under the control of JES2 or JES3, which later can use one of their print routines to print the dataset. These print routines must schedule the datasets for printing, and message classes are used for this purpose. All print routines (called printers or writers) are associated with one or more classes (in all 36 classes) and each dataset to be printed must also be assigned classes. The printer routines select datasets for printing in a very similar way as initiators selects jobs for executions. Use S.ST option of the ISPF menu to view the output dataset.
The SYSOUT parameter can assign this class, known as sysout or output class, to a dataset. Such datasets are called sysout or output datasets.
SYSOUT

```
 //MFCVT01  JOB (),MSGCLASS=A
//STEP01   EXEC PGM=PROGRAM1
//SYSPRINT DD SYSOUT = *
//
//MFCVT01  JOB (),CLASS=A
//STEP01   EXEC PGM=PROGRAM1
//DD1    DD   SYSOUT = A
//
//MFCVT01  JOB (),CLASS=A
//STEP01   EXEC PGM=PROGRAM1
//DD1   DD  DSN=FILE1,SYSOUT = A
//DD2   DD DSN=FILE2,SYSOUT = B
//
```



E.g. 1

```
//SYSOUT      DD SYSOUT=A
```

E.g. 2

```
//SYSPRINT    DD SYSOUT=*
```

This DD statement is used for printing system messages generated by JES2 or JES3. Each step must have the SYSPRINT DD statement. Absence causes “SYSPRINT DD STATEMENT MISSING” message in the sysout.

E.g. 3

```
//SYSOUT DD SYSOUT=*  (or any sysout class may be
                       assigned)
```

This DD statement is used when you have the COBOL DISPLAY clause in your program.
SYSPRINT

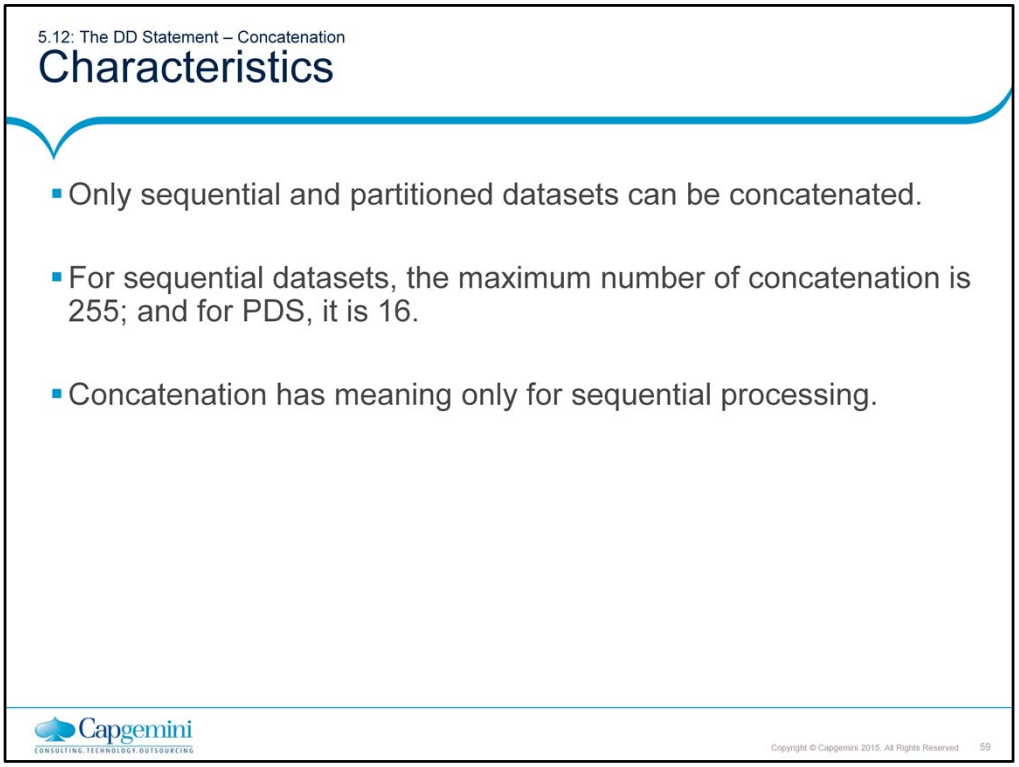
Specifies that an execution report of the load module (PGM) is required. It defines the output file containing the execution messages.

EXAMPLE:

```
//STEP1         EXEC      PGM=IEBGENER
//SYSPRINT    DD          SYSOUT=*
//SYSUT1……..
//
```

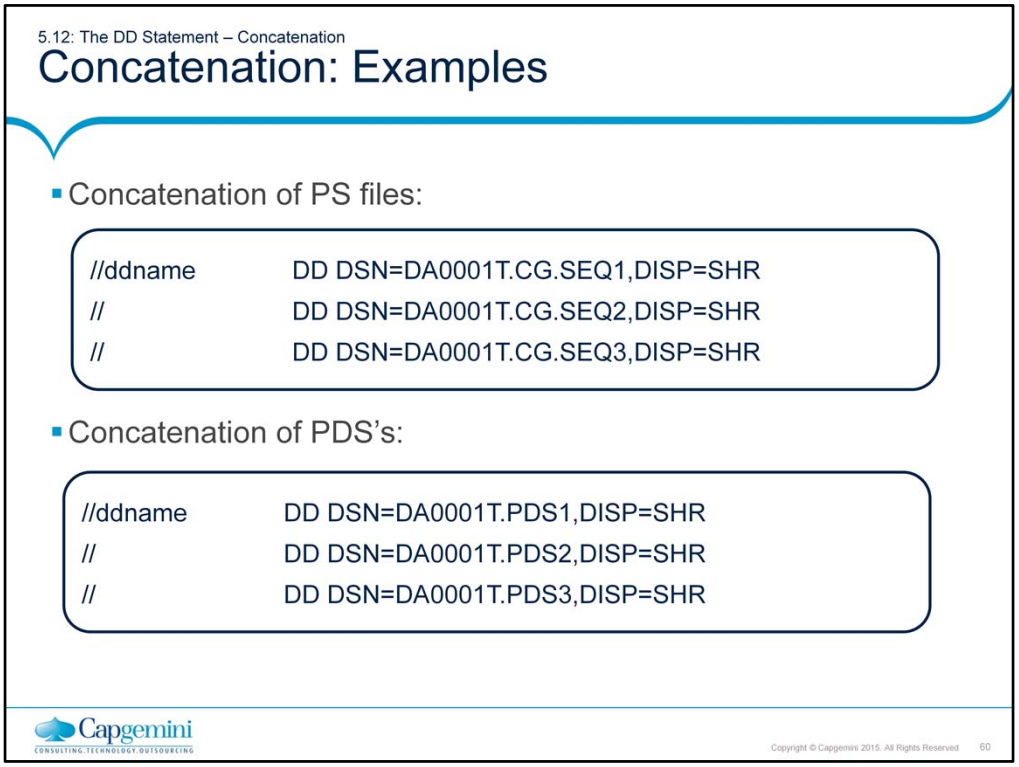
Note:

The SYSOUT parameter specifies the output stream dataset. It routes the report to the class (device) mentioned. SYSOUT=CLASS-code, is the syntax. The class can be any alphanumeric character or an asterisk(*), which refers to the device coded in the MSGCLASS parameter of JOB statement.



Characteristics:
At times, program may have to read in sequence several input datasets as if they were one. This can be accomplished without physically putting the data in one datasets. This is done by concatenating the datasets in JCL code with comparable DCB characteristics without programming changes.

Note that only sequential and partitioned datasets can be concatenated. For sequential datasets, the maximum number of concatenations is 255 and for PDS it is 16. Concatenation has meaning only for sequential processing.

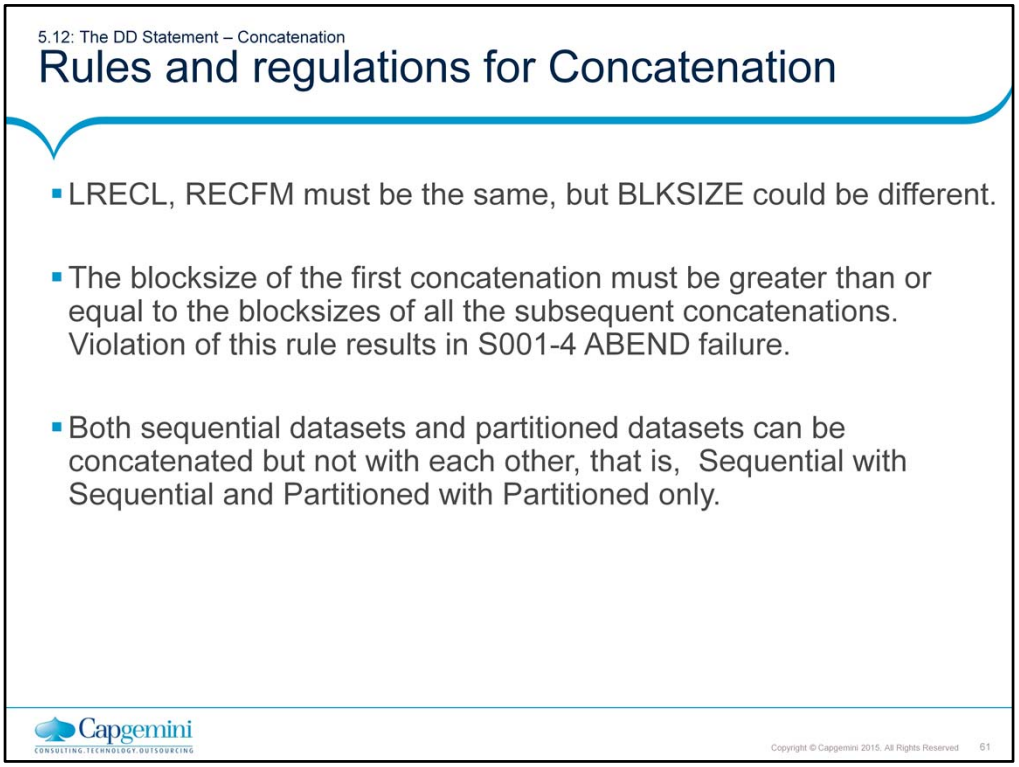


Example:Concatenate datasets by coding a DD statement for each dataset

```
//IN      DD      DSN=Jan.DS,DISP=SHR
//              DD      DSN=Feb.DS,DISP=SHR
//              DD      DSN=Mar.DS,DISP=SHR
```

DDname should be coded for the first DD statement only
If DUMMY dataset is coded, the rest of the datasets down are not processed

```
  //DD1   DD      DSN=A1.PDS,DISP=SHR
//      DD      DUMMY
//      DD      DSN=C1.PDS,DISP=SHR
```



Rules and Restrictions for Concatenation:
There are number of rules and restrictions for concatenations:
The first concatenation is the only one with a ddname.
The logical record length and the record format of concatenated datasets must be the same. However, the blocksizes need not be.
The blocksize of the first concatenation must be greater than or equal to blocksizes of all subsequent concatenation. Violation of this rule results in S001-04 ABEND failure.

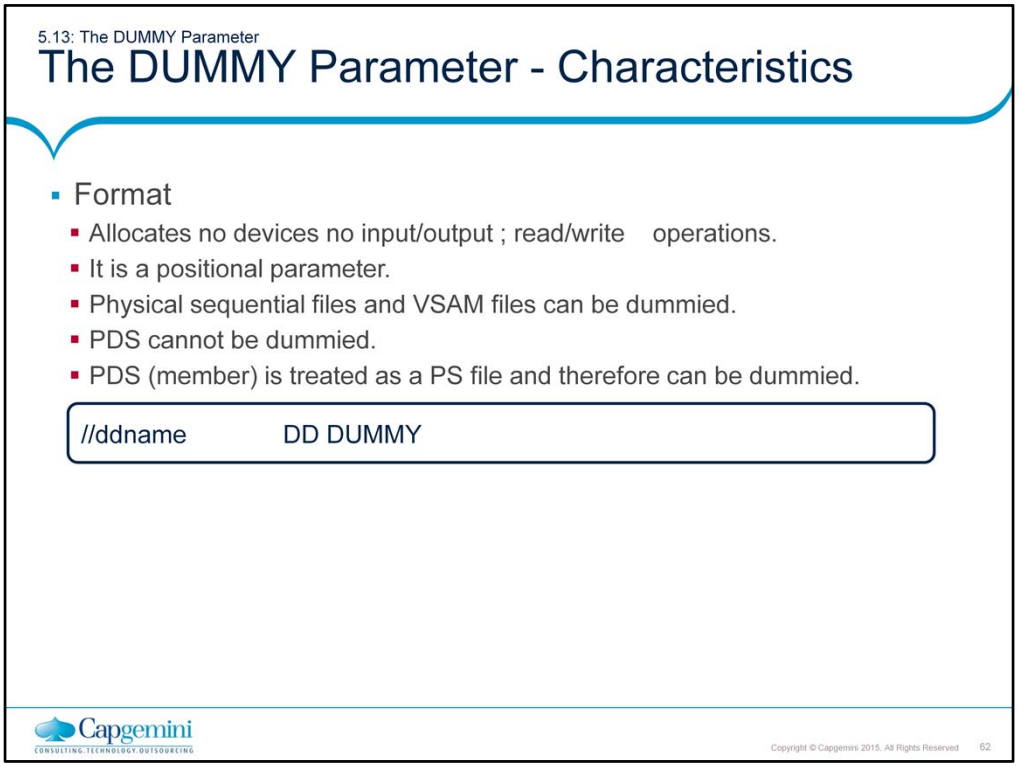
For example, assume that in the JCL below, the first concatenation has a blocksize of 800, the second has a blocksize of 800- and the third has a blocksize of 23400.

Rules and Restrictions for Concatenation (Contd.):
Both sequential datasets and partitioned datasets can be concatenated, but not with each other – sequential with sequential and partitioned with partitioned only.
Member of a PDS is treated as sequential dataset and thus can be concatenated with sequential dataset.

For example,

Disk as well as tape datasets can be concatenated but not with each other. Only like devices should be concatenated, disk with disk and tape with tape.

```
//INFILE  DD DSN=DA0001T.CG.GROUP1,DISP=SHR,DCB=23400
//         DD  DSN=DA0001T.CG.GROUP2,DISP=SHR
//         DD  DSN=DA0001T.CG.GROUP3,DISP=SHR
```



The DUMMY Parameter – Characteristics:

The DUMMY parameter is a positional parameter. At times, one might want to execute a program but suppress read or write operations in certain jobs, For example. not print a report. At other times, one might want to test a program without actually processing data. At times, a DD statement referring to a dataset may be coded in the in a JCL in production region .The DUMMY parameter may be coded in a test region when the same JCL is to be executed.

The DUMMY parameter specifies the following:
No device or external storage be allocated.
No disposition processing is performed.
No input or output operations are performed for sequential access methods.

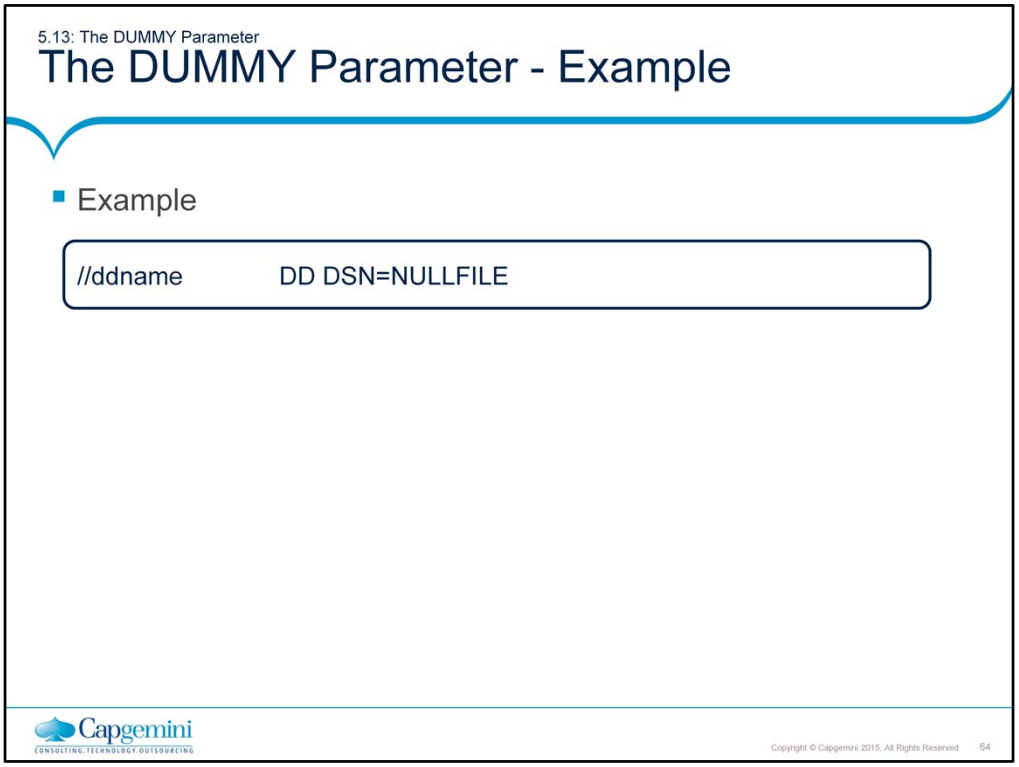
5.13: The DUMMY Parameter

The DUMMY Parameter - Remarks

- When an attempt to dummy a PDS is made, it causes an S013-64 ABEND failure.
- The DCB parameter may be required while coding dummy. Failure to do so may cause an S013-10 ABEND failure.
- NULLFILE is a keyword parameter and is same as DUMMY.
- DUMMY provides a safe way to eliminate I/O activity when required.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 63


5.13: The DUMMY Parameter
The DUMMY Parameter - Example
Example
//ddname DD DSN=NULLFILE
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved 64

5.14: The JOBLIB DD Statement

Characteristics

- The JOBLIB DD statement identifies the program library where the programs to be executed throughout the job reside.
- It must be placed between the JOB and the first EXEC statement.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 65

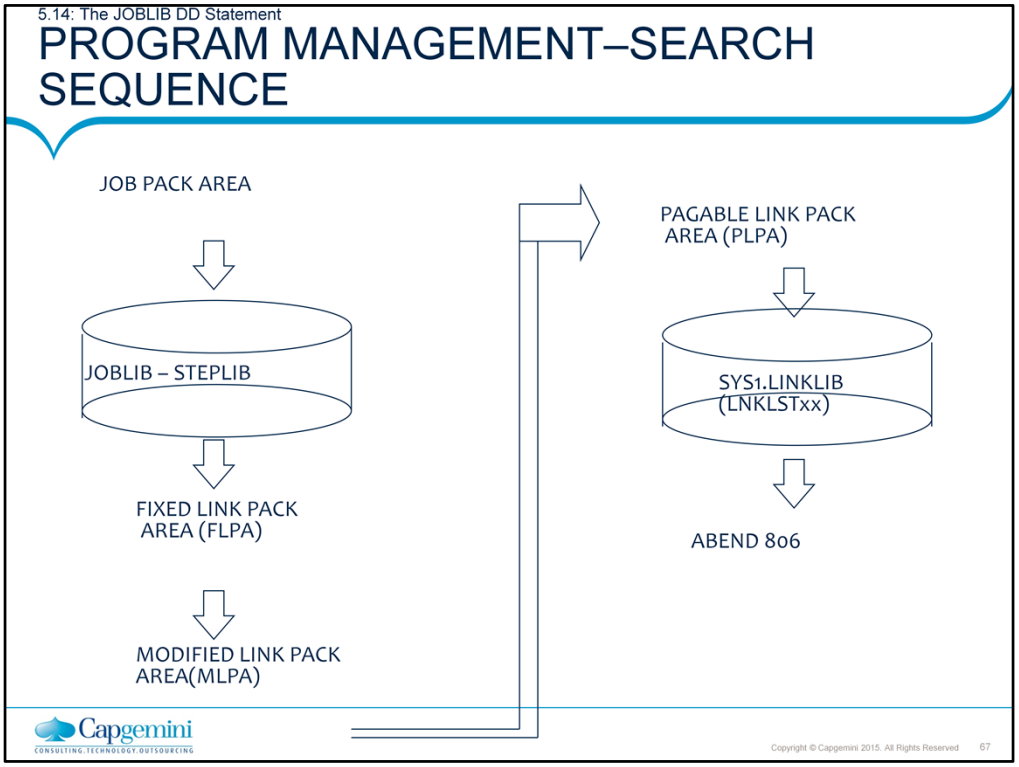
5.14: The JOBLIB DD Statement

SPECIAL DD STATEMENTS

- JOBLIB & STEPLIB:
  - It is not enough to know the program name to be executed, the system needs to know where that program resides.
  - The EXEC statement identifies only the member name only so its location in the system has to be specified by the JOBLIB or STEPLIB statement
  - Load modules will be checked first in this library and then in the system libraries and if it is not found in both the places, then JOB would ABEND with S806 code.
  - Both are DD statements.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 66


5.14: The JOBLIB DD Statement
PROGRAM MANAGEMENT–SEARCH SEQUENCE
JOB PACK AREA
JOBLIB – STEPLIB
FIXED LINK PACK AREA (FLPA)
MODIFIED LINK PACK AREA(MLPA)
PAGABLE LINK PACK AREA (PLPA)
SYS1.LINKLIB (LNKLSTxx)
ABEND 806
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
67

5.14: The JOBLIB DD Statement

PROGRAM MANAGEMENT – SEARCH SEQUENCE

```
//DA0001TA    JOB LA2719,CG,CLASS=A
//JOBLIB      DD    DSN=DA0001T.LIB.LOAD, DISP=SHR
// S1         EXEC  PGM = PROGA
// S2         EXEC  PGM = PROGB
```

- The PROGA Program is expected to reside in DA0001T.LIB.LOAD as a member of a library and the system searches the directory. If not found, system searches certain predefined libraries. If it is still not found, S806 ABEND failure occurs.

- A JOBLIB DD statement can have several concatenations (maximum : 16).

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 68

5.14: The JOBLIB DD Statement

The JOBLIB DD - Example 2

- All concatenations may be searched to locate a program. If, however, the program is found in a concatenation other than the last one, other concatenations will not be used.
- When duplicate member names exists in different concatenations, the user can decide which one is to be executed by determining the sequence of the **concatenations**

```
//DA0001TA      JOB LA2719,CG,CLASS=A
//JOBLIB        DD    DSN=DA0001T.LIB.LOAD, DISP=SHR
// S1           EXEC  PGM = PROGA
// S2           EXEC  PGM = PROGB
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 69

5.15: The STEPLIB DD Statement

Characteristics

- The STEPLIB DD statement identifies the library, where the program to be executed for the step where STEPLIB is found resides.
- It can be placed anywhere after the EXEC statement.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 70

5.15: The STEPLIB DD Statement

Example1

- Program PROGA is expected to reside in PROD.LOADLIB1 as a member of the library.
- If not found, default libraries are searched

```
//PROD213   JOB   SH21, CLASS=P
//S1        EXEC  PGM=PROGA
//STEPLIB   DD    DSN=DA0001T.LIB.LOAD1,
//                DISP=SHR
//S2        EXEC  PGM=PROGB
//STEPLIB   DD    DSN = DA0001T.LIB.LOAD2,
//                DISP=SHR
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 71

5.15: The STEPLIB DD Statement

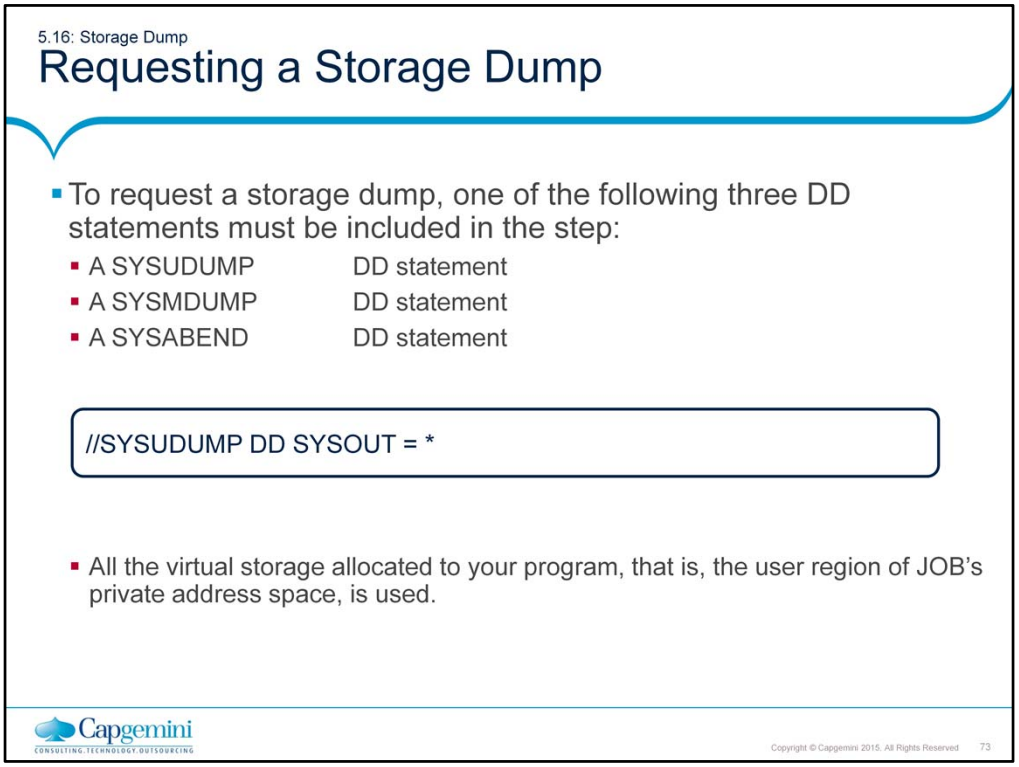
Example2

- A STEPLIB DD statement has the effect of negating the JOBLIB DD statement for a particular step.

```
//DA0001TA    JOB     LA2719,CG,CLASS=P
 //JOBLIB     DD      DSN=DA0001T.LIB.LOAD1,DISP=SHR
 //S1         EXEC    PGM=PROGA
 //S2         EXEC    PGM=PROGB
 //STEPLIB    DD      DSN =DA0001T.LIB.LOAD2,
 //                   DISP=SHR
 //S3         EXEC    PGM=PROGC
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 72



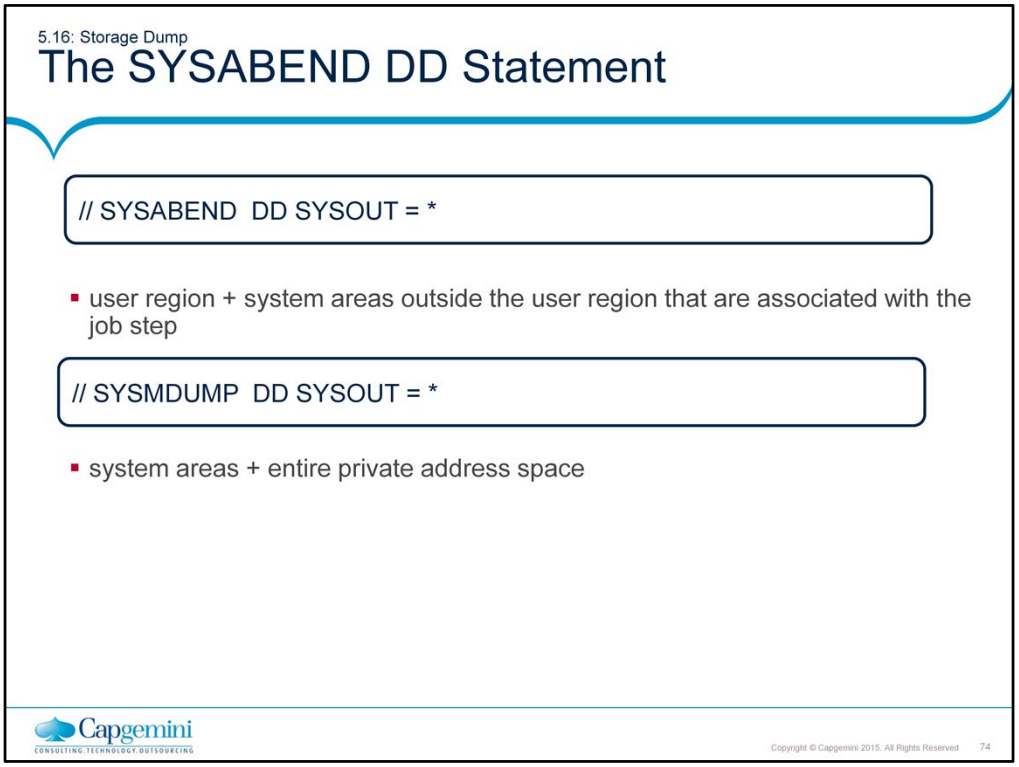
Requesting a Storage Dump:
When a step encounters an ABEND failure, it is often advantageous to request a virtual storage dump, which can then be helpful in determining
the cause of an ABEND. To request a storage dump, one of the following three DD statements must be included in the step:
A SYSUDUMP DD statement
A SYSMDUMP DD statement
A SYSABEND DD statement

```
//SYSUDUMP   DD SYSOUT=*
```

All virtual storage allocated to your program, that is, user region of job's address space, is used. It is a formatted dump. SYSUDUMP usually writes to sysout. It can, however, write to a disk dataset, providing a way to preserve the SYSUDUMP information for later viewing and analysis.

No DCB is required.

```
//SYSUDUMP DD DSN=DA0001T.DUMPFILE,SPACE=(TRK,(0,5),RLSE),
//                  DISP=(,DELETE,CATLG),UNIT=SYSDA
```



The SYSABEND DD Statement:
This is same as SYSUDUMP DD statement except for the fact that the dump is unformatted. This type of dump is very difficult to analyze
unless it is saved on a disk and then processed by the PRDUMP service aid.
SYSMDUMP is seldom used.

```
//SYSMUDUMP   DD SYSOUT=*
```

When a SYSUDUMP DD statement is included in a step which ABEND's, a formatted virtual storage dump will be provided. This dump also
includes information about the failed step, as well as most of the MVS storage-resident information, which is of no use to the average user.
SYSABEND is intended for system programmer.

```
//SYSABEND   DD SYSOUT=*
```

Remark:
If neither a SYSUDUMP nor a SYSMDUMP nor a SYSABEND statement is coded within a JCL of an ABENDing step, a small amount of information is provided. This information is seldom useful in resolving the problem that caused the ABEND failure.

5.16: Storage Dump

The SYSABEND DD Statement - Example

- Remark:
  - When more than one of the above statements is included in the JCL of a step, only the last one is used.

```
//SYSDUMP    DD    SYSOUT=*
//SYSDUMP    DD    DSN=DA0001T.DUMP1,
//                 DISP=(,DELETE,CATLG),UNIT=SYSDA,
//                 SPACE=(CYL,(0,5),RLSE)
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 75

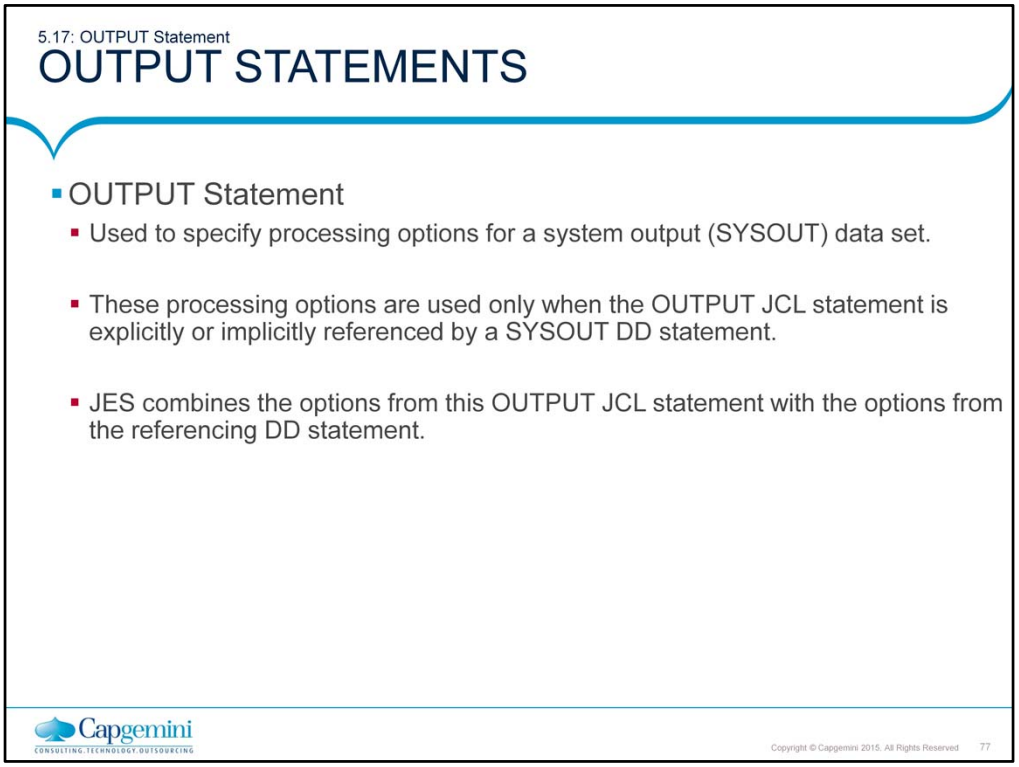
5.16: Storage Dump

SPECIAL DD STATEMENTS

- JOBCAT & STEPCAT DD STATEMENT
  - The dataset used in the step are first checked in the STEPCAT (ICF or VSAM catalog) before checking in system catalog.
  - If no STEPCAT in the step and there is a JOBCAT, then the datasets are first searched in JOBCAT before checking in the system catalog.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 76



OUTPUT JCL statements are useful in processing the output of one SYSOUT data set in several ways.
For example, a SYSOUT dataset can be sent to a distant site for printing, as shown in statement OUT1, while it is also printed locally, as shown in statement OUT2:

```
//OUT1 OUTPUT DEST=STLNODE.WMSMITH
//OUT2 OUTPUT CONTROL=DOUBLE
//DS DD SYSOUT=C,OUTPUT=(*.OUT1,*.OUT2)
```

5.17: Storage Dump

Lab

- Day 2 Lab

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 78

Summary

- DD statement must appear in a step to define the resources used by that specific step.
- DSN parameter identifies the name of the dataset to be created or retrieved.
- DISP parameter specifies how to dispose of the dataset when the step terminates (normally or abnormally).
- UNIT and VOLUME parameters work together to specify the location of the dataset.
- The SPACE parameter is used to allocate or alter space to a dataset.
- The LABEL parameter is used to specify sequence on a volume and type of label of a tape dataset.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Summary

- The DCB parameter specifies characteristics that are stored in the Data Control Block of the file.
- The SYSOUT parameter can assign the sysout or output class to a dataset.
- Only sequential and partitioned datasets can be concatenated.
- The DUMMY parameter allocates no devices no input/output and read/write operations.
- The JOBLIB and STEPLIB DD and statement identifies the program library.
- The STORAGE DUMP is used place a request for a storage dump.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 80

Review Question

- Question 1. Which of the following is/are positional parameter?
  - Option1 : DUMMY
  - Option2 : STORAGE DUMP
  - Option3 : JOBLIB

- Question 2. Which of the following statements is/are true?
  - Option1 : A JOBLIB statements can have several concatenations.
  - Option2 : A STEPLIB negates the effect of JOBLIB DD statement.

