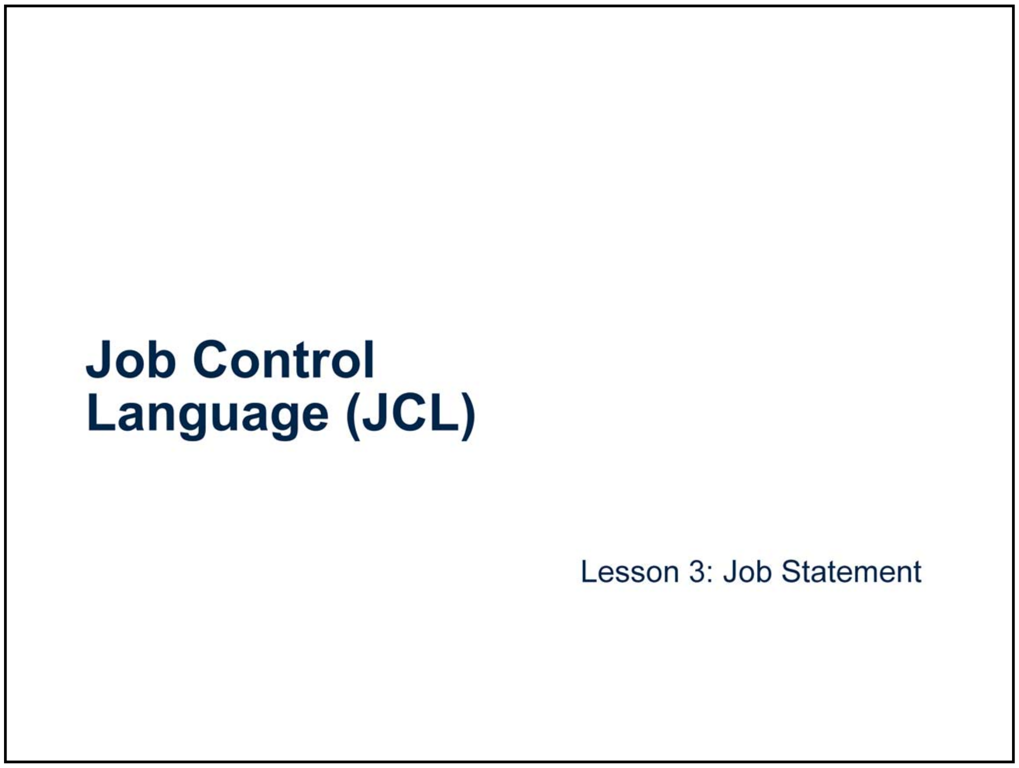
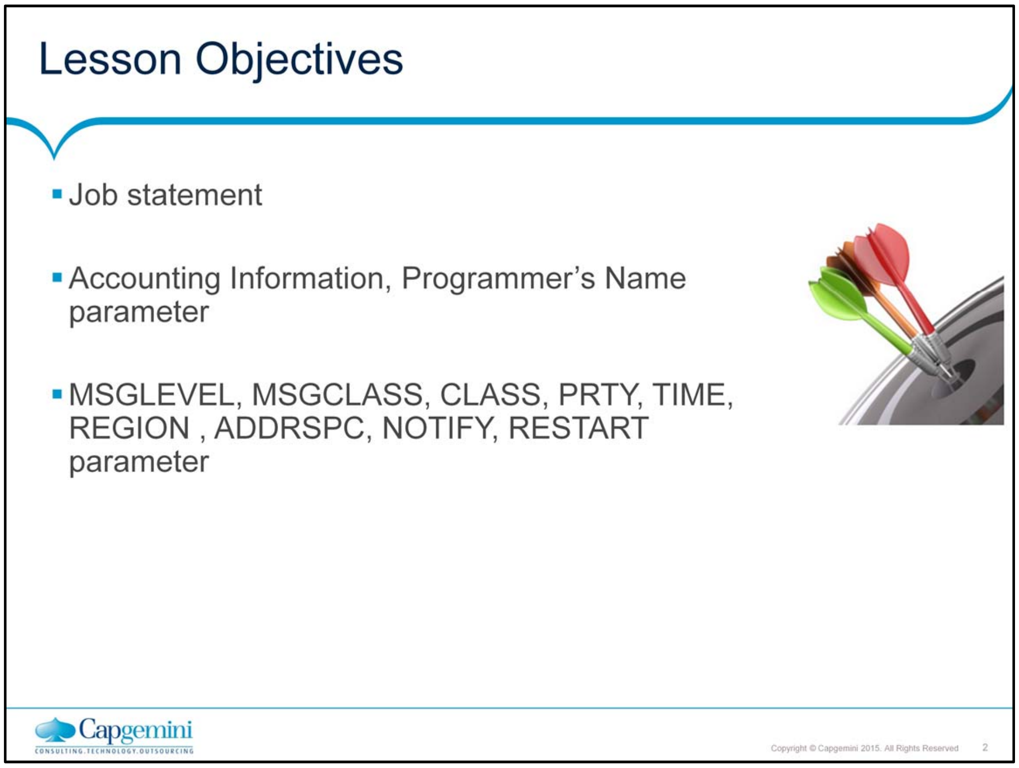


# Job Control Language (JCL)

Lesson 3: Job Statement

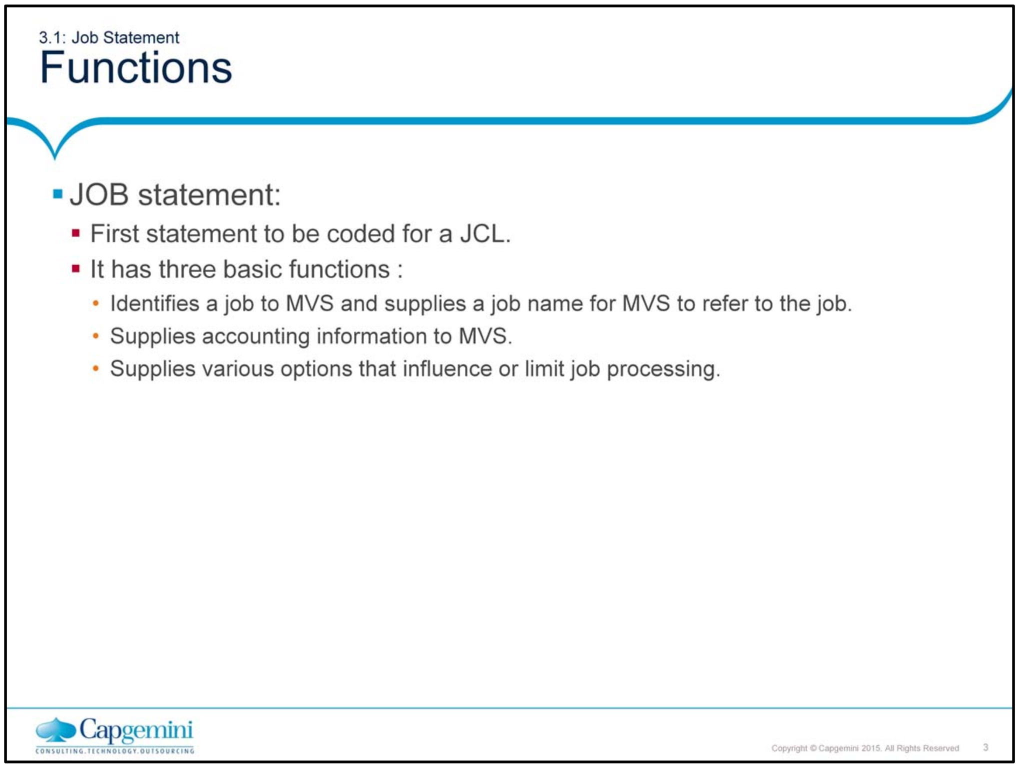


## Lesson Objectives

- Job statement
- Accounting Information, Programmer’s Name parameter
- MSGLEVEL, MSGCLASS, CLASS, PRTY, TIME, REGION , ADDRSPC, NOTIFY, RESTART parameter

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

2



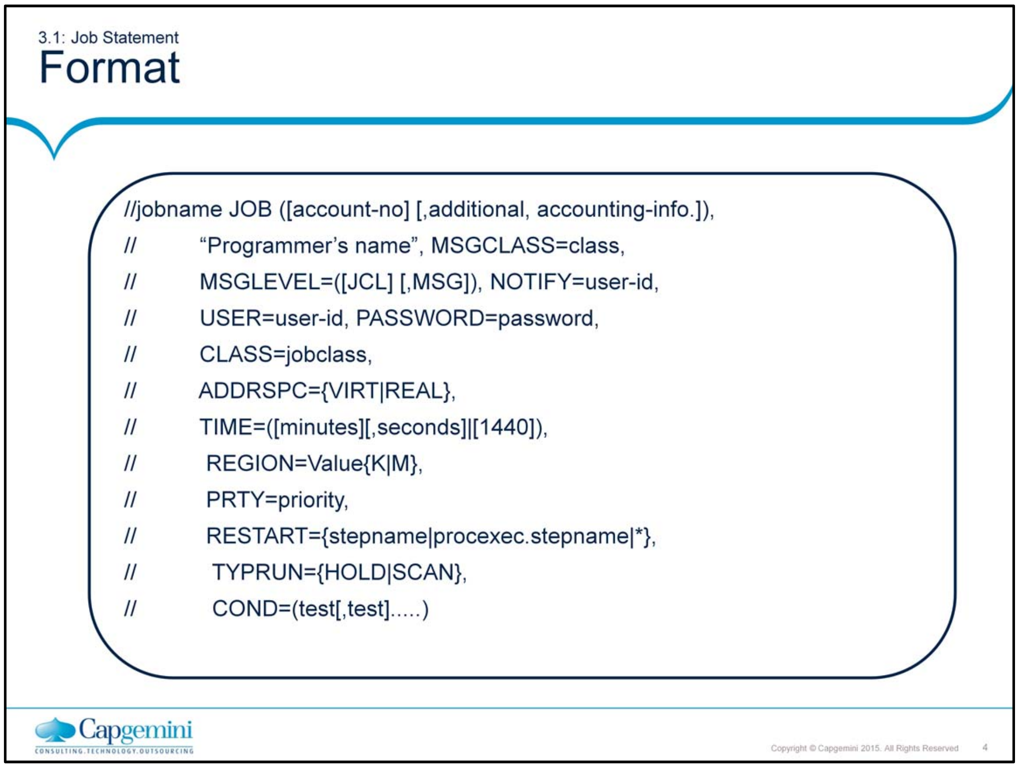
3.1: Job Statement

## Functions

- JOB statement:
  - First statement to be coded for a JCL.
  - It has three basic functions :
    - Identifies a job to MVS and supplies a job name for MVS to refer to the job.
    - Supplies accounting information to MVS.
    - Supplies various options that influence or limit job processing.

A JOB statement must be at the beginning of every job submitted to the system for execution. It must have a name. The absence of a job name will result in a JCL error. JOB statement identifies a job to the operating system with the job name operand. There are three possible delimiters for a job during a reading process:

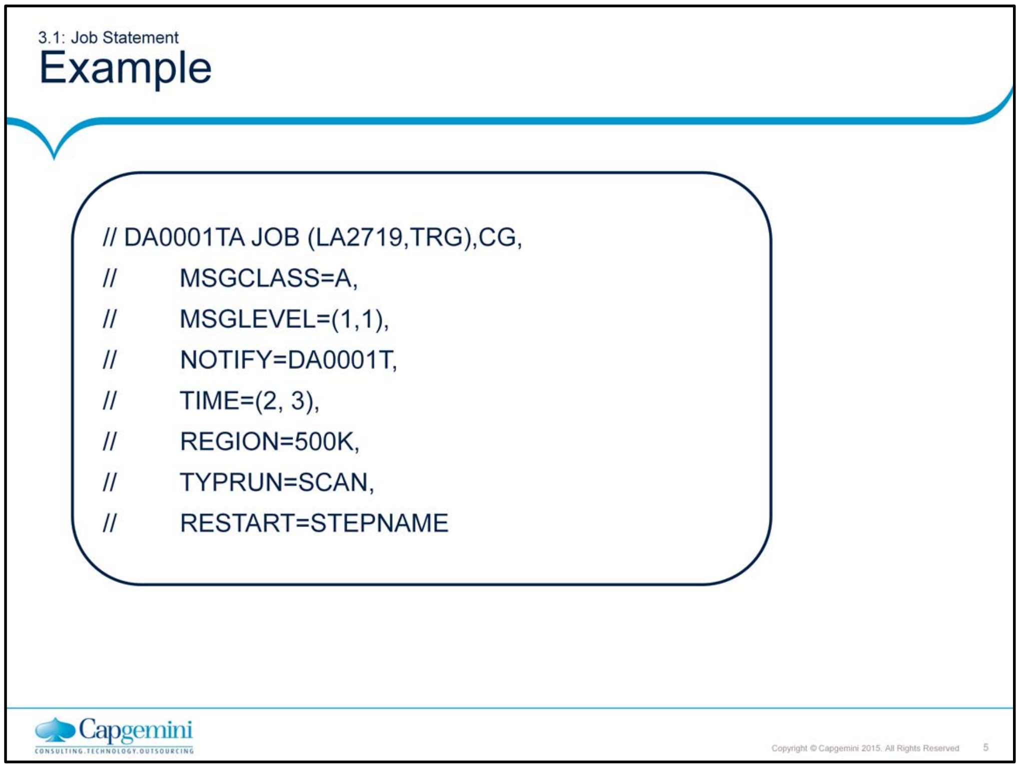
Another job statement in the input stream. It signals the end of (reading) one JOB and the beginning of (reading) another
A null statement. Following a null statement all JCL statements except a JOB statement will be ignored
End-of-file on the reading device, meaning there are no more statements to be read in



3.1: Job Statement

## Format

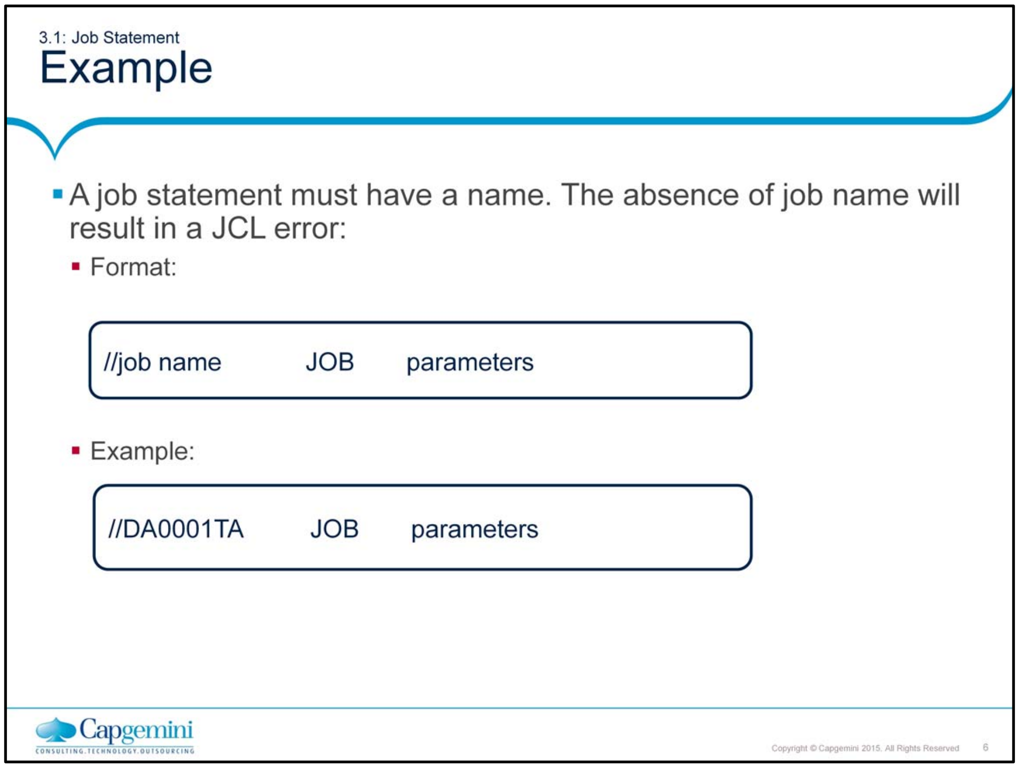
```
//jobname JOB ([account-no] [,additional, accounting-info.]),
//         “Programmer’s name”, MSGCLASS=class,
//         MSGLEVEL=([JCL] [,MSG]), NOTIFY=user-id,
//         USER=user-id, PASSWORD=password,
//         CLASS=jobclass,
//         ADDRSPC={VIRT|REAL},
//         TIME=([minutes][,seconds]|[1440]),
//          REGION=Value{K|M},
//          PRTY=priority,
//          RESTART={stepname|procexec.stepname|*},
//           TYPRUN={HOLD|SCAN},
//           COND=(test[,test].....)
```



3.1: Job Statement

## Example

```
// DA0001TA JOB (LA2719,TRG),CG,
//         MSGCLASS=A,
//         MSGLEVEL=(1,1),
//         NOTIFY=DA0001T,
//         TIME=(2, 3),
//         REGION=500K,
//         TYPRUN=SCAN,
//         RESTART=STEPNAME
```



3.1: Job Statement

## Example

- A job statement must have a name. The absence of job name will result in a JCL error:
  - Format:

```
//job name          JOB      parameters
```

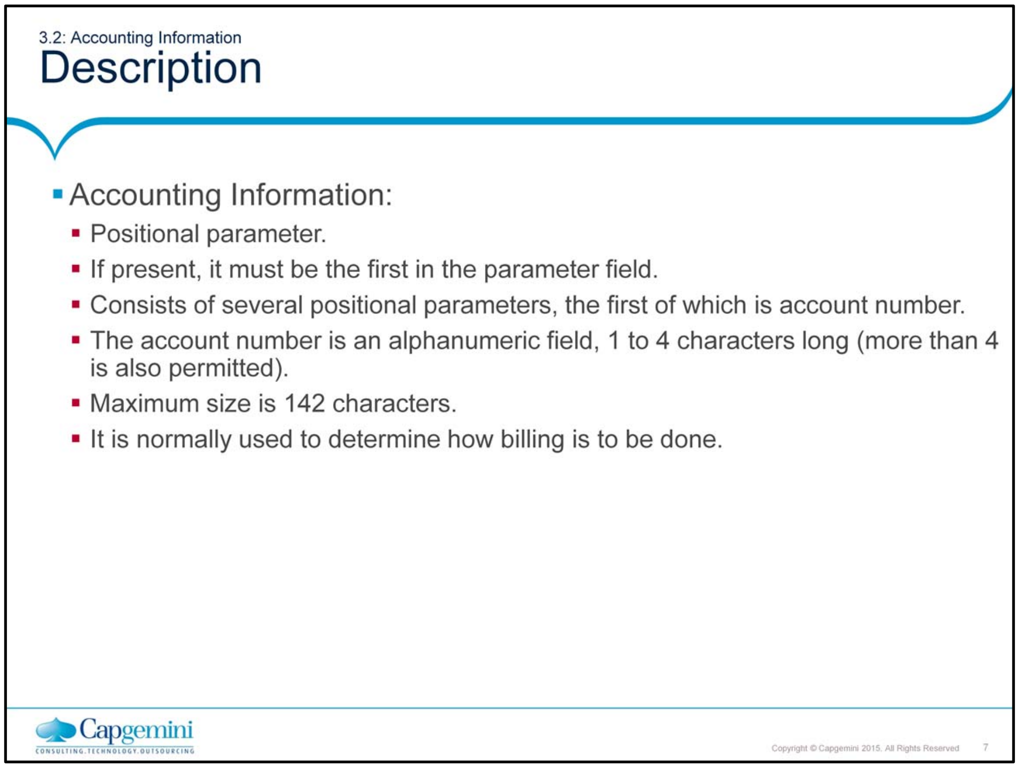
  - Example:

```
//DA0001TA        JOB      parameters
```

The job name cannot exceed 8 characters and is usually the user id (login id). Login id is generally 7 characters, so you need to add a suffix else the system prompts you to enter the character when a job is submitted to the system for execution.

When a job is submitted to the system, a job number is also assigned so that the job can be further identified. This way jobs with the same name can be uniquely identified. Jobs with the same name cannot execute simultaneously. If several jobs with the same name are submitted they execute sequentially even if additional jobs could be executing. Jobs waiting to run because of this time conflict are shown in hold status.

The rest of the JOB statement contains positional parameters followed by keyword parameter.



3.2: Accounting Information

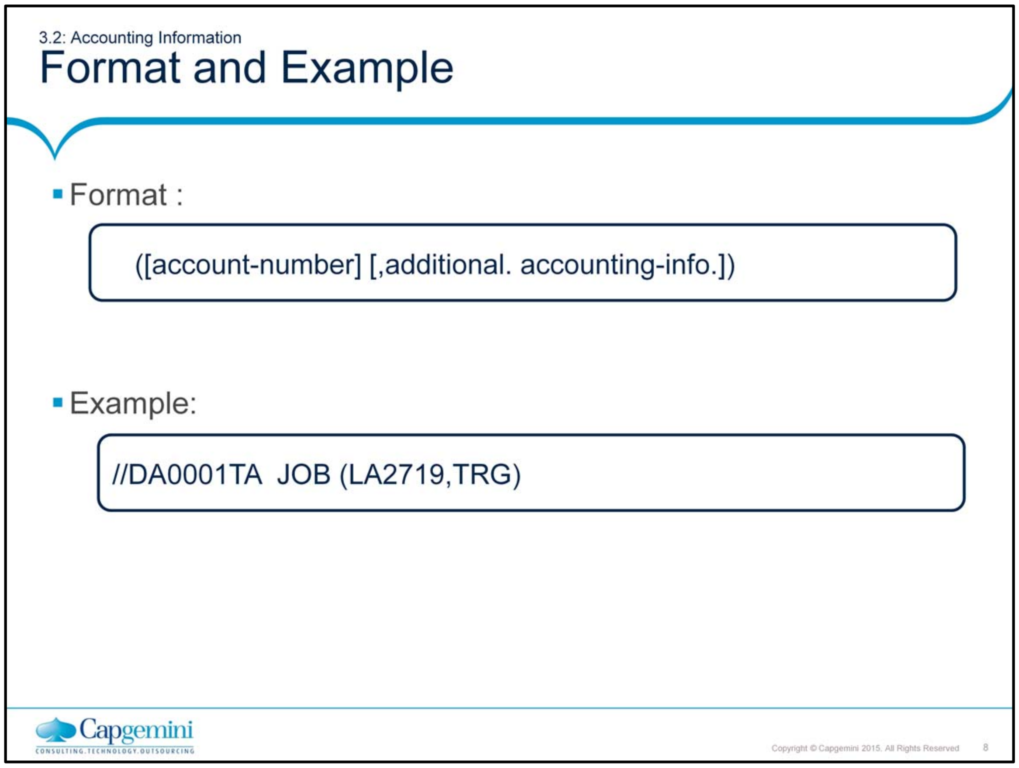
## Description

- Accounting Information:
  - Positional parameter.
  - If present, it must be the first in the parameter field.
  - Consists of several positional parameters, the first of which is account number.
  - The account number is an alphanumeric field, 1 to 4 characters long (more than 4 is also permitted).
  - Maximum size is 142 characters.
  - It is normally used to determine how billing is to be done.

If a positional parameter is present (it normally is), it must be the first in the parameter field. It can have a maximum of 142 characters (including parentheses and commas but not apostrophes). It is used to tie the resources used by the job to the appropriate account.

The account-number is an alphanumeric field from 1 to 4 characters long (many installations permit the use of more than 4 characters).

Additional-accounting-information is installation dependent. Many of the fields are not very important and are not often used.



3.2: Accounting Information

## Format and Example

- Format :

```
([account-number] [,additional. accounting-info.])
```

- Example:

```
//DA0001TA  JOB (LA2719,TRG)
```

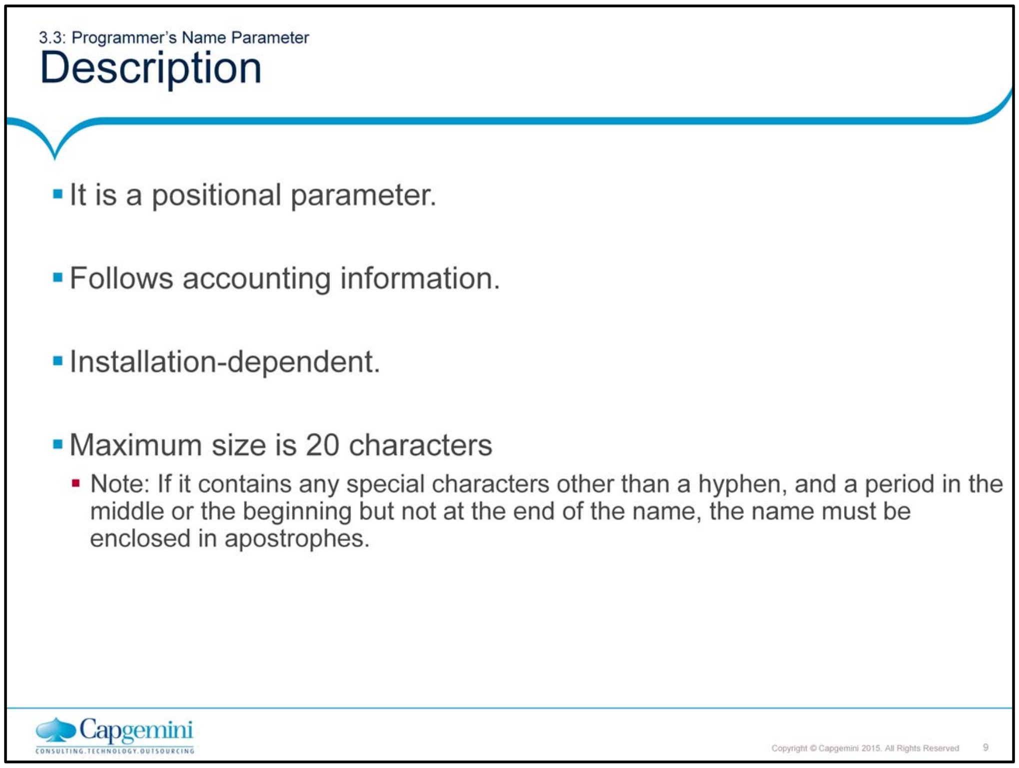
Example:

```
//DA0001TA JOB LA2719, parameters..
```

LA2719 is the account number for the training dept. This varies from one project to another.

Remark: An installation has the option of making the account number mandatory and most installations do. If so, its absence causes a JCL error.

If the account number is incorrectly specified, in this case its not JCL error. However when job is submitted to the system for execution, we get the message “JOB NOT RUN” in the SYSOUT.



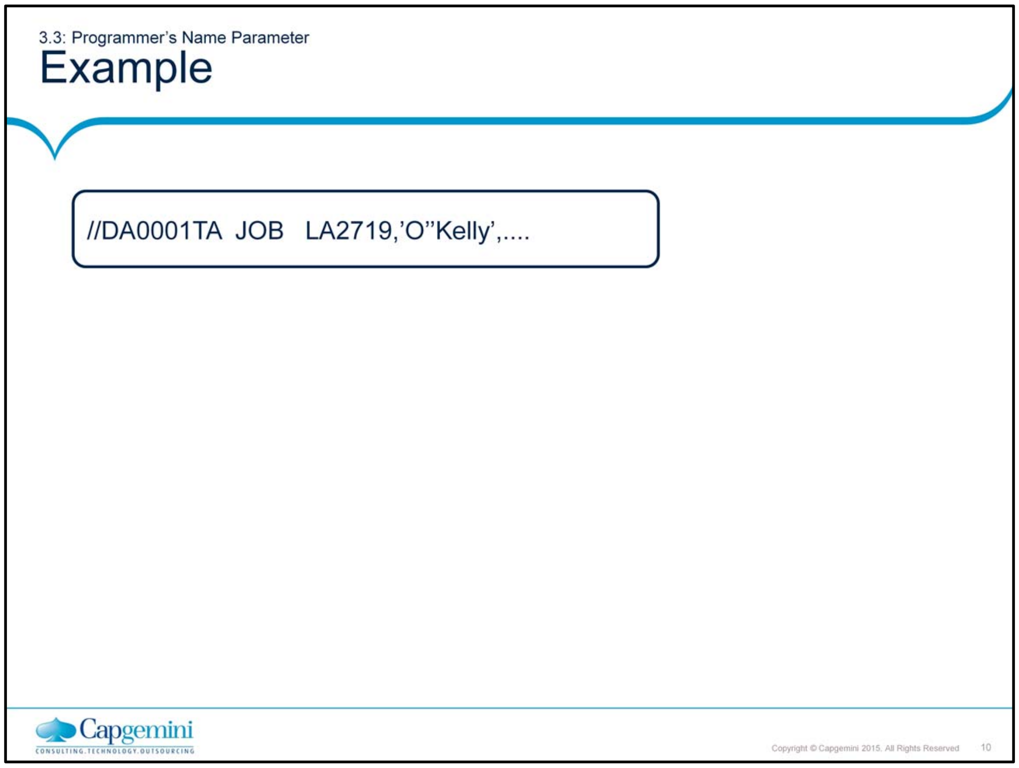
3.3: Programmer's Name Parameter

## Description

- It is a positional parameter.
- Follows accounting information.
- Installation-dependent.
- Maximum size is 20 characters
  - Note: If it contains any special characters other than a hyphen, and a period in the middle or the beginning but not at the end of the name, the name must be enclosed in apostrophes.

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

Following the accounting information parameter, another positional parameter, the programmer's name can be coded. The installation determines if this parameter is required or not. If required, it must be coded immediately after the accounting information, and its omission will cause a JCL error. The programmer's name cannot exceed 20 characters. If it contains any special characters other than a hyphen and a period in the middle of the beginning (but not at the end) of the name, the name must be enclosed in apostrophes. The apostrophes are not added to the length of the name.



3.3: Programmer's Name Parameter

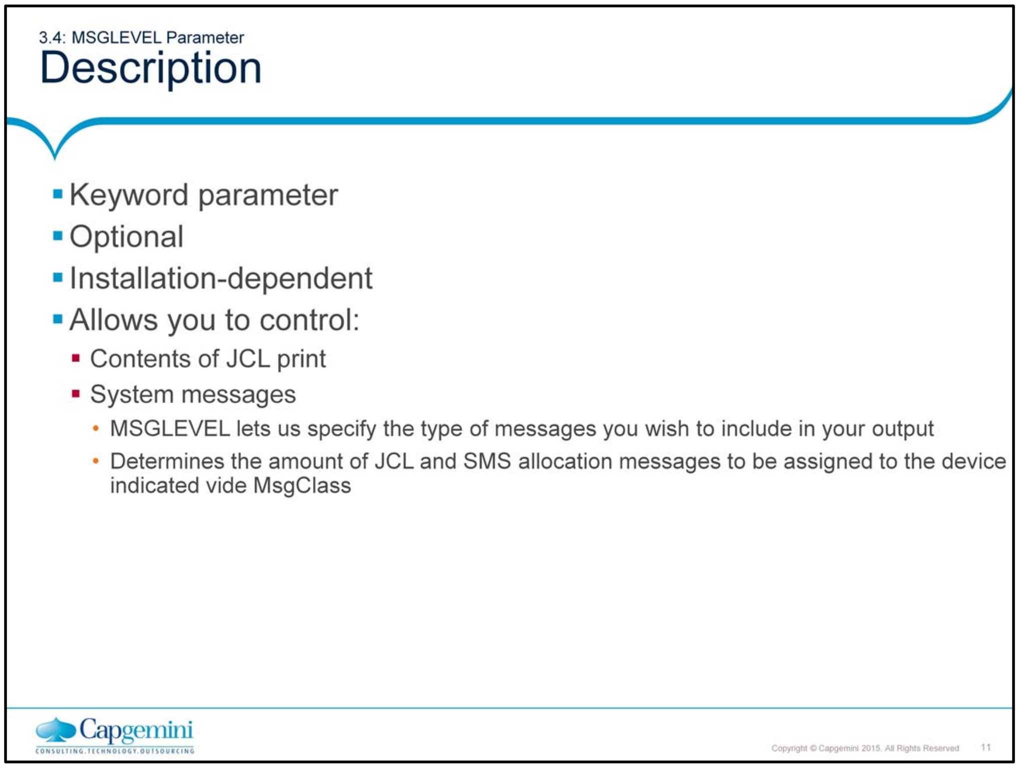
## Example

```
//DA0001TA  JOB   LA2719,'O''Kelly',....
```

If a name contains an apostrophe (e.g., D'COSTA), two apostrophes must be coded. They count as one character in the length of the name.

Example:

```
//DA0001TA JOB LA2719,Sheela,parameters

//DA0001TA JOB LA2719,'D''COSTA',parameters
```

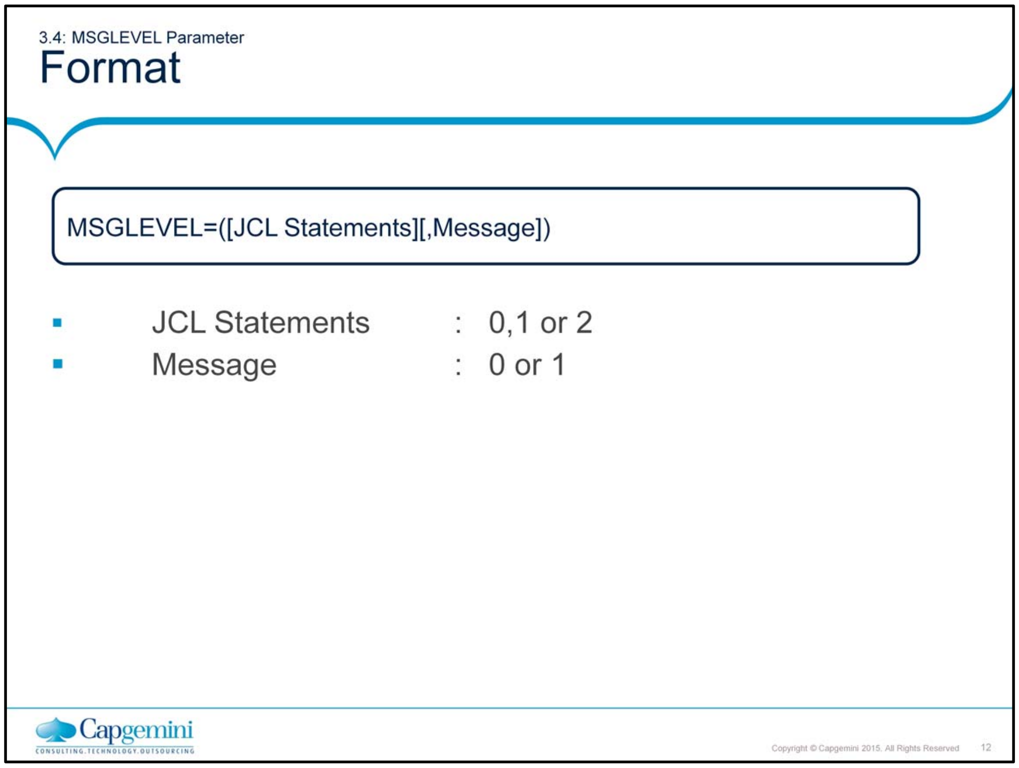


3.4: MSGLEVEL Parameter

## Description

- Keyword parameter
- Optional
- Installation-dependent
- Allows you to control:
  - Contents of JCL print
  - System messages
    - MSGLEVEL lets us specify the type of messages you wish to include in your output
    - Determines the amount of JCL and SMS allocation messages to be assigned to the device indicated vide MsgClass

This parameter specifies whether the submitted JCL or JCL-related messages should be shown on the job's output.



3.4: MSGLEVEL Parameter

## Format

MSGLEVEL=([JCL Statements][,Message])

| | | |
|---|---|---|
| JCL Statements | : | 0,1 or 2 |
| Message | : | 0 or 1 |

General syntax:

```
MSGLEVEL=([jcl][,messages])        keyword parameter
```

jcl - 0, 1, or 2

0 - Only the JOB statement will be shown

1 - All JCL is shown

Instream

Expanded cataloged procedures

Symbolic parameter substitutions

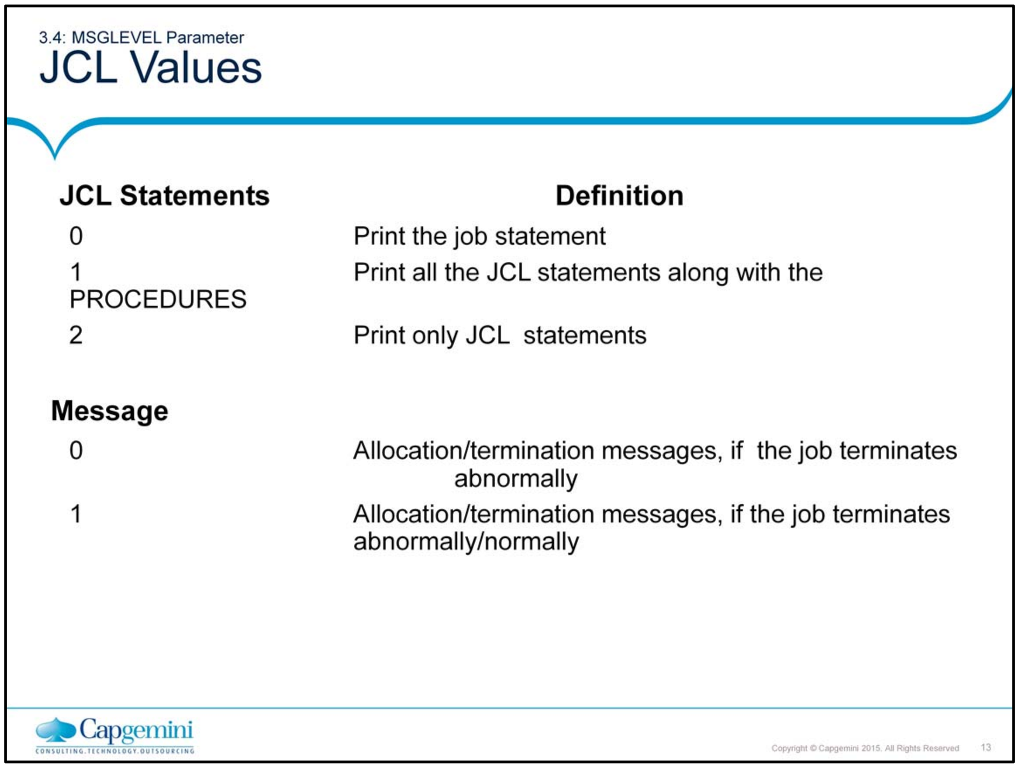
- All JCL is shown, but not expanded procedure listing.

messages – 0 or 1

0 - No messages will be shown i.e. information about step completion.

– All messages will be shown viz. allocation and termination messages.

The messages sub-parameter can be thought of as On (1) or Off (0).
The default in majority of the installations is (1,1).



3.4: MSGLEVEL Parameter

## JCL Values

| JCL Statements | Definition |
|---|---|
| 0 | Print the job statement |
| 1 | Print all the JCL statements along with the PROCEDURES |
| 2 | Print only JCL statements |

**Message**

|  |  |
|---|---|
| 0 | Allocation/termination messages, if the job terminates abnormally |
| 1 | Allocation/termination messages, if the job terminates abnormally/normally |

Remark:

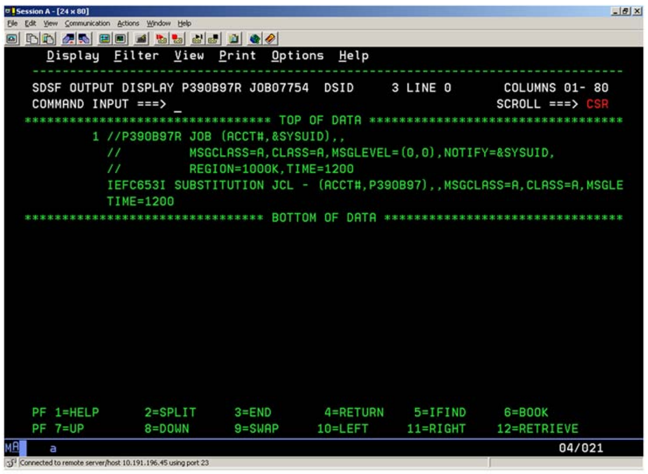
1. If the entire parameter or either of the two fields is omitted, an installation-defined default is assumed.
   MSGLEVEL=1 ------------------ MSGLEVEL=(1,default)
   MSGLEVEL=(,1) ------------------ MSGLEVEL=(default,1)
   Parameter omitted ------------------ MSGLEVEL=(default,default)

2. If the job encounters an ABEND failure, the second field always defaults to 1 even if coded as 0.

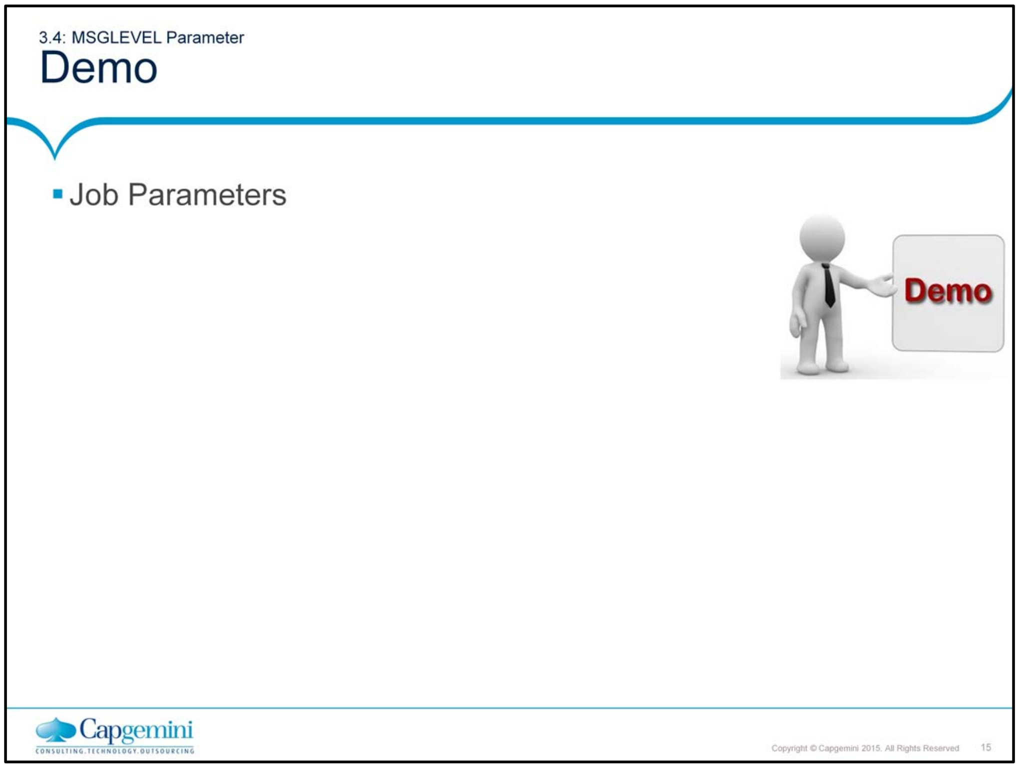
3.4: MSGLEVEL Parameter

## The JOB Statement – MSGLEVEL (0,0)

- JESJCL: Shows only JOB statement details



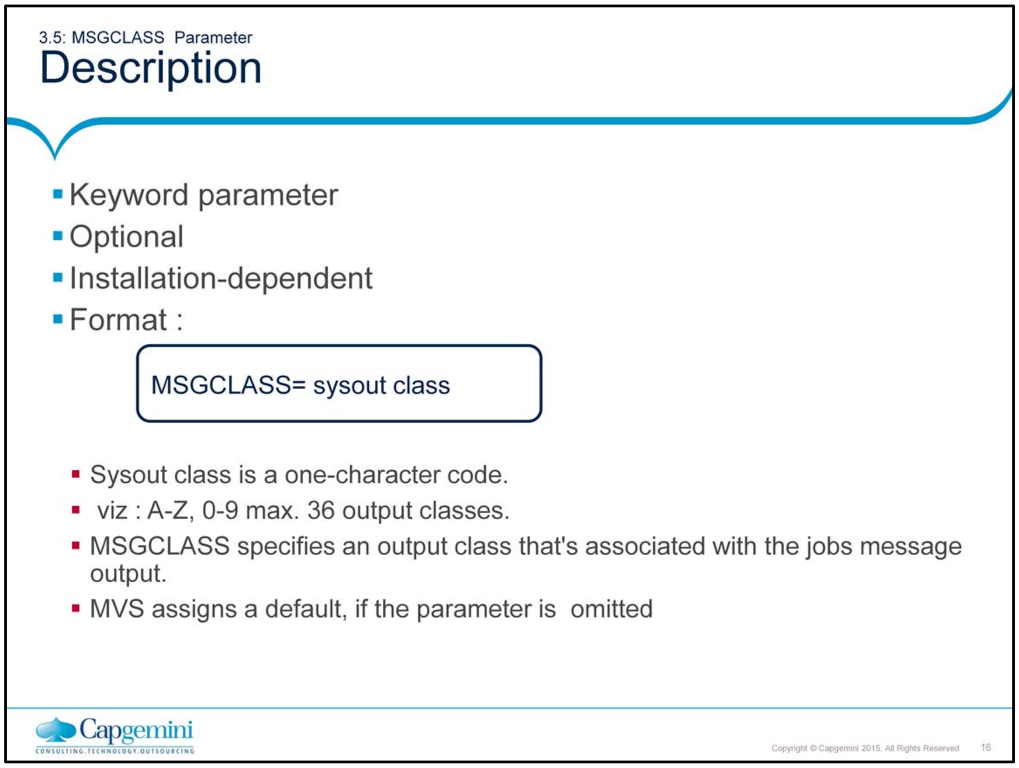
```
    Display  Filter  View  Print  Options  Help
 -------------------------------------------------------------------------------
 SDSF OUTPUT DISPLAY P390B97R JOB07754  DSID     3 LINE 0       COLUMNS 01- 80
 COMMAND INPUT ===> _                                          SCROLL ===> CSR
********************************* TOP OF DATA **********************************
        1 //P390B97R JOB (ACCT#,&SYSUID),,
          //         MSGCLASS=A,CLASS=A,MSGLEVEL=(0,0),NOTIFY=&SYSUID,
          //         REGION=1000K,TIME=1200
          IEFC653I SUBSTITUTION JCL - (ACCT#,P390B97),,MSGCLASS=A,CLASS=A,MSGLE
          TIME=1200
******************************** BOTTOM OF DATA ********************************




 PF 1=HELP      2=SPLIT     3=END       4=RETURN    5=IFIND     6=BOOK
 PF 7=UP        8=DOWN      9=SWAP     10=LEFT     11=RIGHT    12=RETRIEVE
```



3.4: MSGLEVEL Parameter

## Demo

- Job Parameters



3.5: MSGCLASS Parameter

## Description

- Keyword parameter
- Optional
- Installation-dependent
- Format :

```
MSGCLASS= sysout class
```

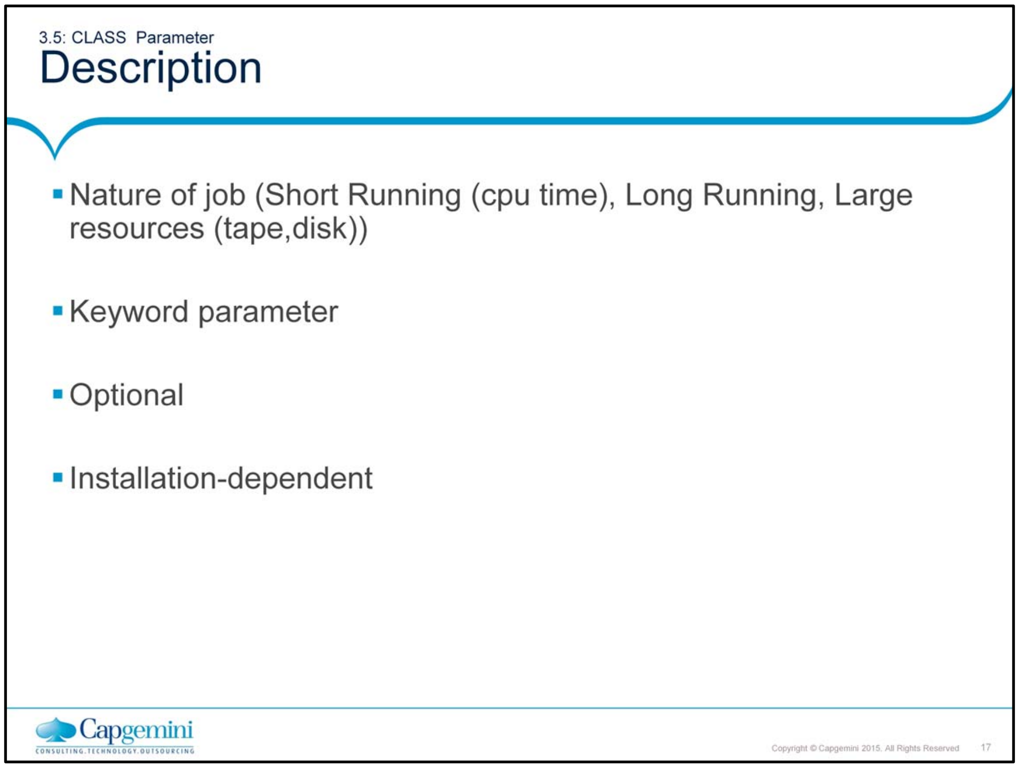
  - Sysout class is a one-character code.
  - viz : A-Z, 0-9 max. 36 output classes.
  - MSGCLASS specifies an output class that's associated with the jobs message output.
  - MVS assigns a default, if the parameter is omitted

This parameter assigns a sysout class to the Job log. The job log consists of what is what is known as system or JES datasets:

JES2 or JES3 log
JCL and its associated messages
Allocation and Termination messages

MSGCLASS - indicates the format of output
- Specifies output class for
Job log (collection of all operations)
List (collection of all printed output like compiled
class - A character from A to Z or a number from 0 to 9 (in all 36 classes)
MSGLEVEL parameter indicates whether or not one wishes to print the JCL statements and allocation messages. The MSGLEVEL can save paper. After a job is debugged, there may be no need to print all the JCL and allocation messages each time it runs. To reduce printing to a minimum, one may wish to MSGLEVEL=(0,0).

Remark: If the MSGCLASS parameter is omitted, an installation-defined default will be used.



3.5: CLASS Parameter

## Description

- Nature of job (Short Running (cpu time), Long Running, Large resources (tape,disk))
- Keyword parameter
- Optional
- Installation-dependent

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

This parameter assigns a class to a job.
General Syntax:

```
CLASS=jobclass          Keyword parameter
```

jobclass – A letter from A to Z or a number from 0 to 9 ( in all 36 classes).

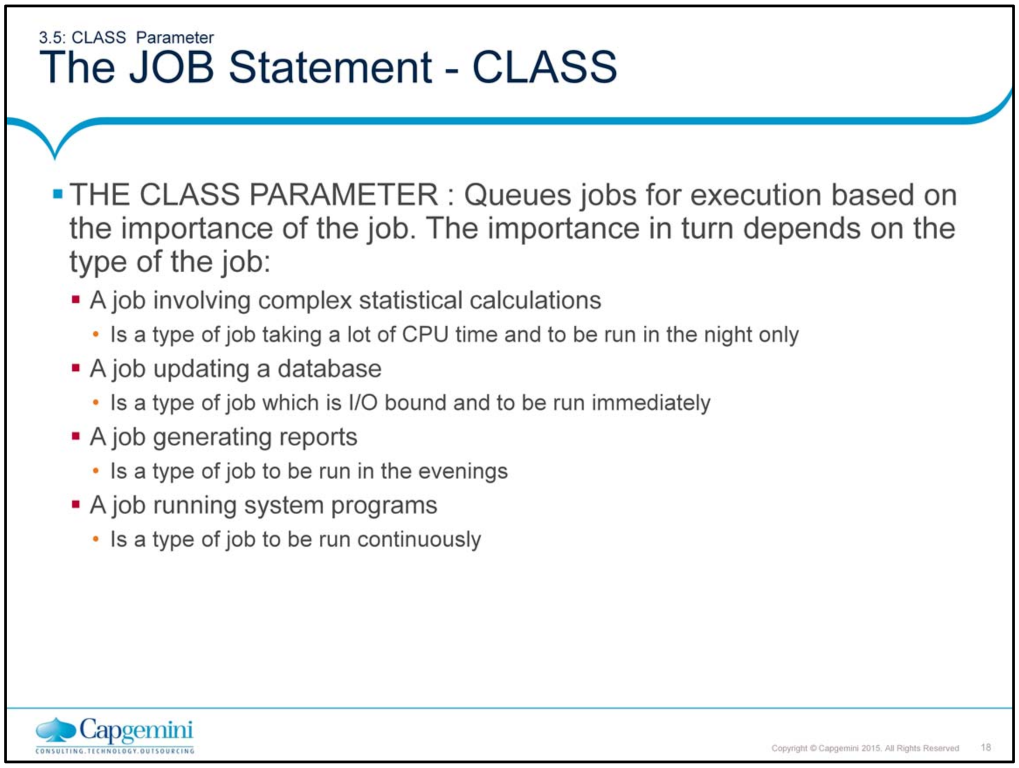
The job class affects job's processing in these ways:

When job is submitted, it is placed in an input queue where it waits to be executed. Queues can be thought of as waiting lines for jobs. Each job class has its own input queue

Job waits in the input queue until it is selected by an initiator to be processed. Each initiator is set to a list of job classes that it can select from.

Simply put, Jobclass identifies the nature of the job:

- Short running or long running
- Resource utilization



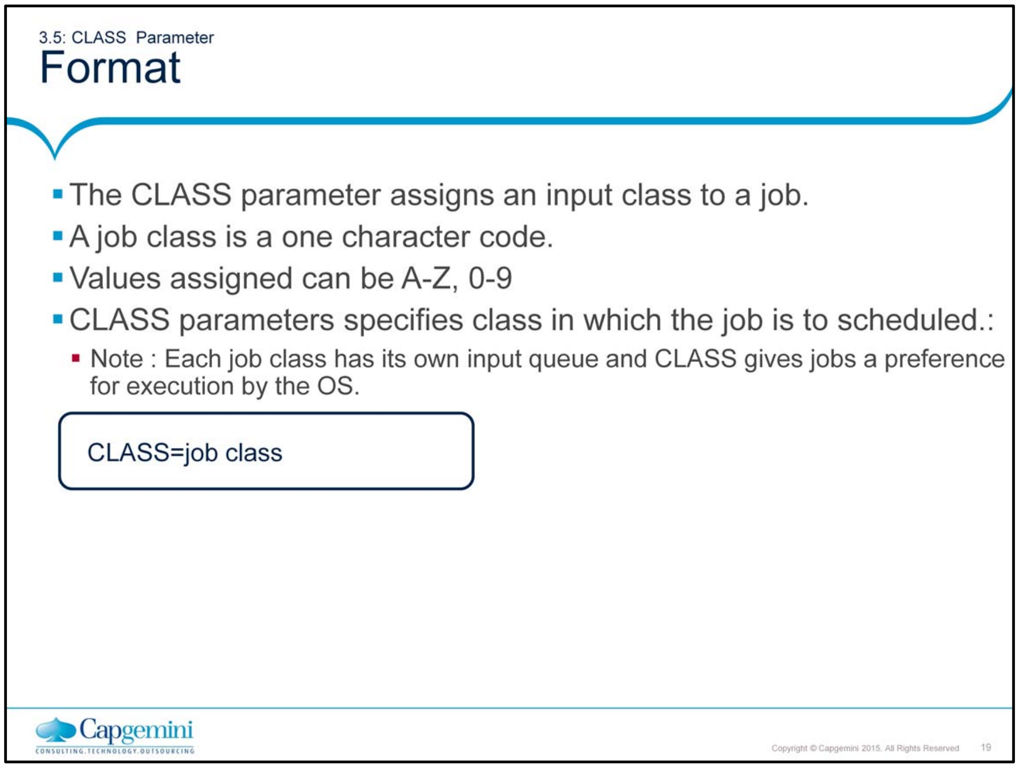
3.5: CLASS Parameter

## The JOB Statement - CLASS

- THE CLASS PARAMETER : Queues jobs for execution based on the importance of the job. The importance in turn depends on the type of the job:
  - A job involving complex statistical calculations
    - Is a type of job taking a lot of CPU time and to be run in the night only
  - A job updating a database
    - Is a type of job which is I/O bound and to be run immediately
  - A job generating reports
    - Is a type of job to be run in the evenings
  - A job running system programs
    - Is a type of job to be run continuously

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

18



3.5: CLASS Parameter

## Format

- The CLASS parameter assigns an input class to a job.
- A job class is a one character code.
- Values assigned can be A-Z, 0-9
- CLASS parameters specifies class in which the job is to scheduled.:
  - Note : Each job class has its own input queue and CLASS gives jobs a preference for execution by the OS.

```
CLASS=job class
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Each installation group jobs that have like characteristics into classes. By segregating jobs with similar characteristics, an installation can maintain a good mix of the jobs running at a given moment. This maintains system throughput and efficient use of resources.

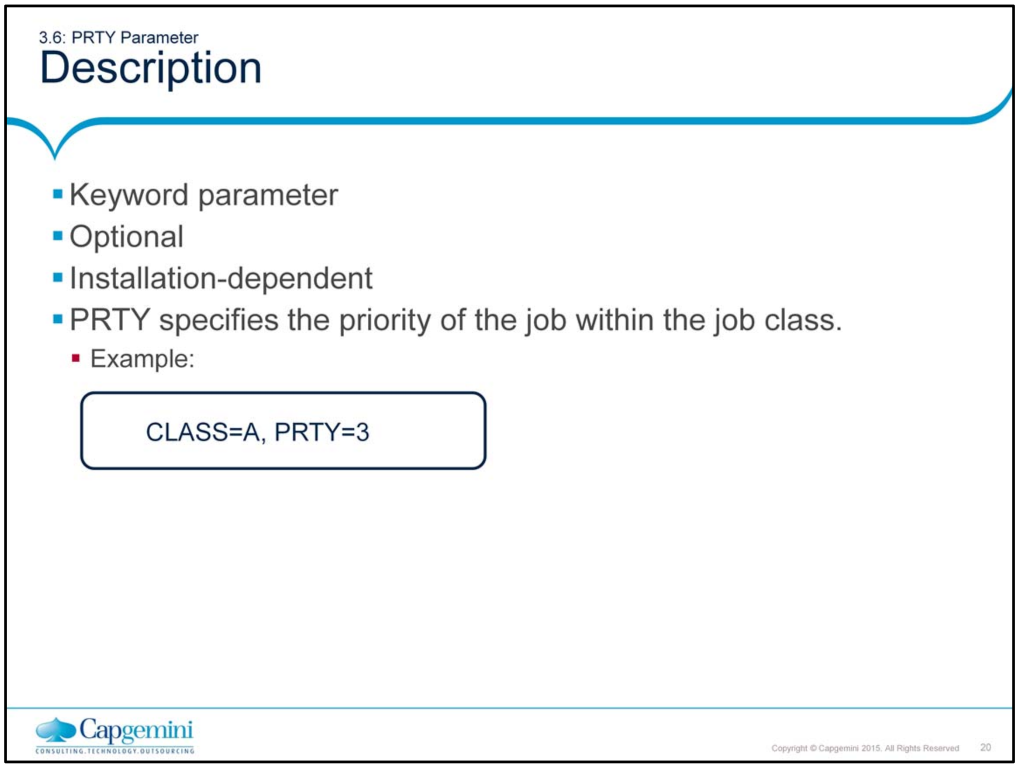
For example, suppose the default CLASS is A:

Job statement:

```
//DA0001TA JOB  LA2719,CG,MSGCLASS=A,MSGLEVEL=(1,1)
```

is equivalent to:

```
//DA0001TA   JOB
//        LA2719,CG,MSGCLASS=A,MSGLEVEL=(1,1),CLASS=A
```



3.6: PRTY Parameter

## Description

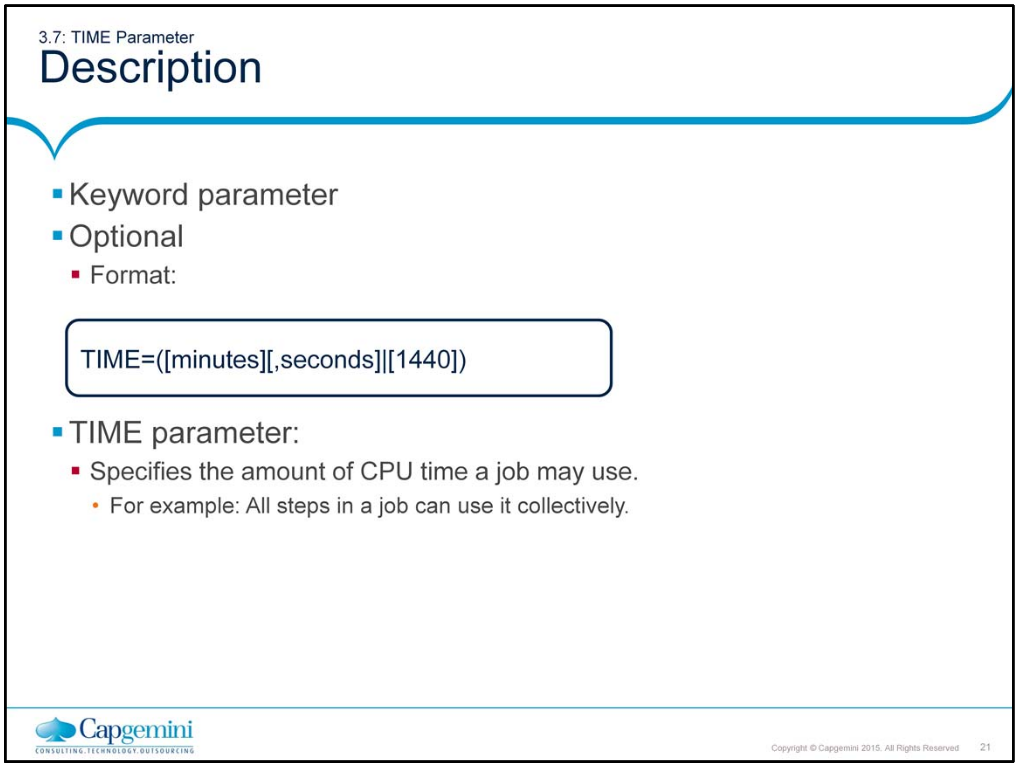
- Keyword parameter
- Optional
- Installation-dependent
- PRTY specifies the priority of the job within the job class.
  - Example:

```
CLASS=A, PRTY=3
```

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

This parameter determines the scheduling priority of a job in relation to other jobs in the job input queue of the same class.
The PRTY parameter is used to define the job's input class selection priority. The higher the number, the better (greater) the priority. The PRTY parameter simply controls the job's position in the input queue. It has no affect on the job's performance. Jobs with higher priorities will be selected before job's will lower priority. A job's priority does not affect its performance. Once the job is selected for execution, the priority function is finished. Two jobs having same job class and same priority will be executed in sequence. It is meaningless to compare the PRTY parameter of two jobs
belonging to different classes.
Format :
PRTY=priority-no.

For JES2, it is a no. between 0 & 15
For JES3, it is a no. between 0 & 14
0 - lowest priority
15 - highest priority for JES2)


3.7: TIME Parameter

## Description

- Keyword parameter
- Optional
  - Format:

```
TIME=([minutes][,seconds]|[1440])
```

- TIME parameter:
  - Specifies the amount of CPU time a job may use.
    - For example: All steps in a job can use it collectively.

This parameter specifies the total amount of CPU time that all steps in a job can use collectively.

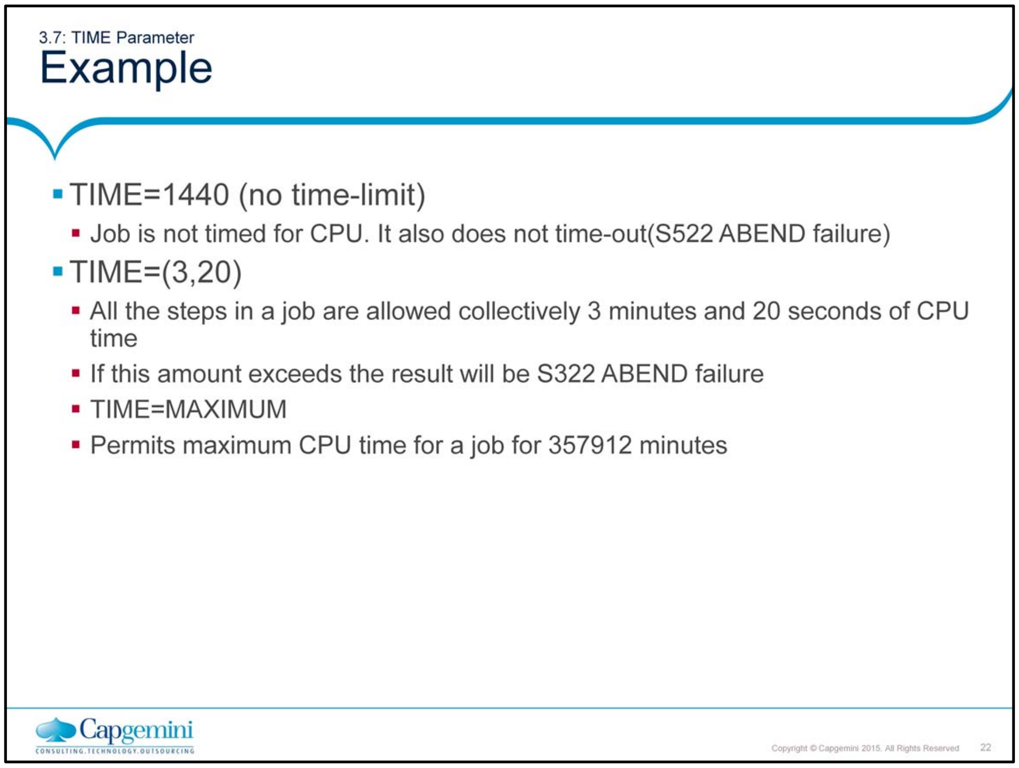
minutes - a number from 1 to 357912 (248.55 days)
seconds - a number from 1 to 59

The job is not timed for CPU. Note that TIME=1440 is rarely used, and most installations disallow its use in a testing environment. TIME=1440 should be used by an on-line system like CICS OR ADS/O.

When the TIME parameter is omitted, an installation-defined default is used. This default is usually very high and unlikely to cause an S322 ABEND failure unless the program goes into an endless loop.

If the TIME parameter is also coded in the JOB statement and exec statement within job , both will be in effect and either can cause a S322 ABEND failure. It is not advisable to use them both.

CPU time is the amount of time that the computer devoted to the job after it was selected for processing. It is not the amount of time it was in the machine.


3.7: TIME Parameter

## Example

- TIME=1440 (no time-limit)
  - Job is not timed for CPU. It also does not time-out(S522 ABEND failure)
- TIME=(3,20)
  - All the steps in a job are allowed collectively 3 minutes and 20 seconds of CPU time
  - If this amount exceeds the result will be S322 ABEND failure
  - TIME=MAXIMUM
  - Permits maximum CPU time for a job for 357912 minutes

Reasons to code TIME

While testing a new program which may go into a loop, unless it is “timed out” automatically (system code 522)

Special jobs which need to by-pass any Time limit barrier and which must not be “timed out” too.

TIME parameter puts an upper limit on the amount of CPU time that a job may use.

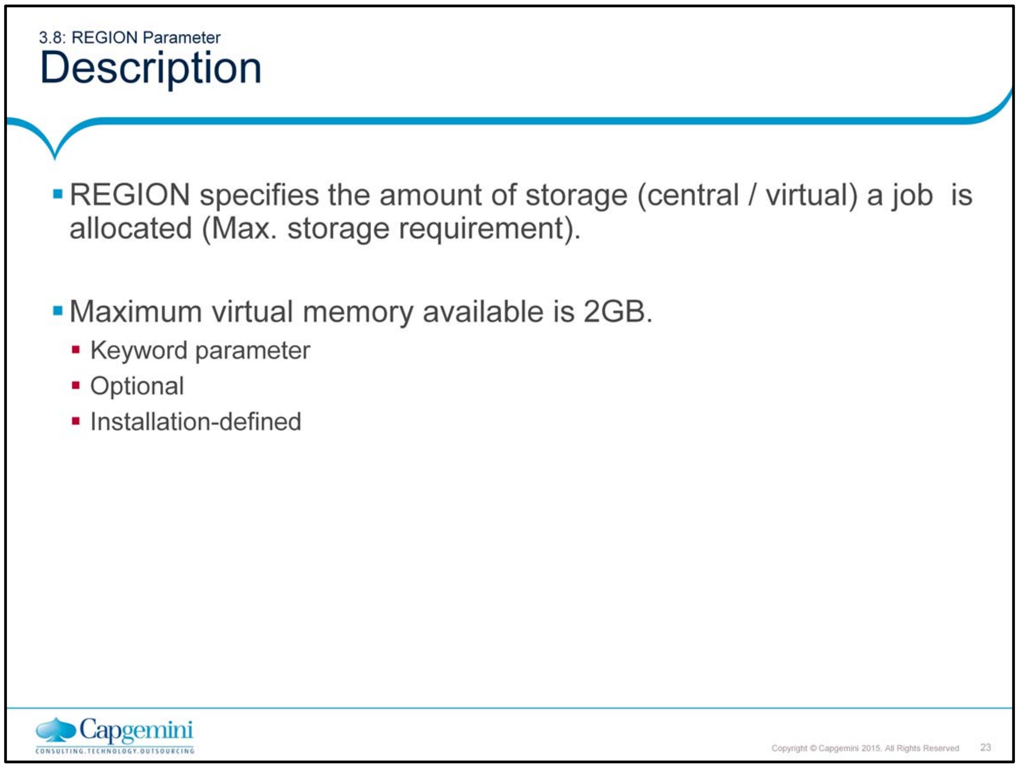
Example: TIME=(3,20). All the steps in the job are allowed collectively 3 minutes and 20 seconds of CPU time. If this amount is exceeded, the result will be a S322 ABEND failure.

If the TIME parameter is coded using only minutes, seconds defaults to zero. For example, TIME=5 is the same as TIME=(5,0).

If the TIME Parameter is coded using only seconds, minutes defaults to zero. For example, TIME=(,6) is the same as TIME=(0,6).

The TIME parameter is intended almost exclusively for a testing environment and should be coded to preempt the program going into CPU loop.

The TIME parameter can also be supplied by the CLASS parameter. When the TIME parameter is omitted and the CLASS parameter does not supply it, the job will not be timed for CPU time. However each step will be individually timed (TIME parameter at EXEC statement or its installation-defined default), unless it contains TIME=1440.



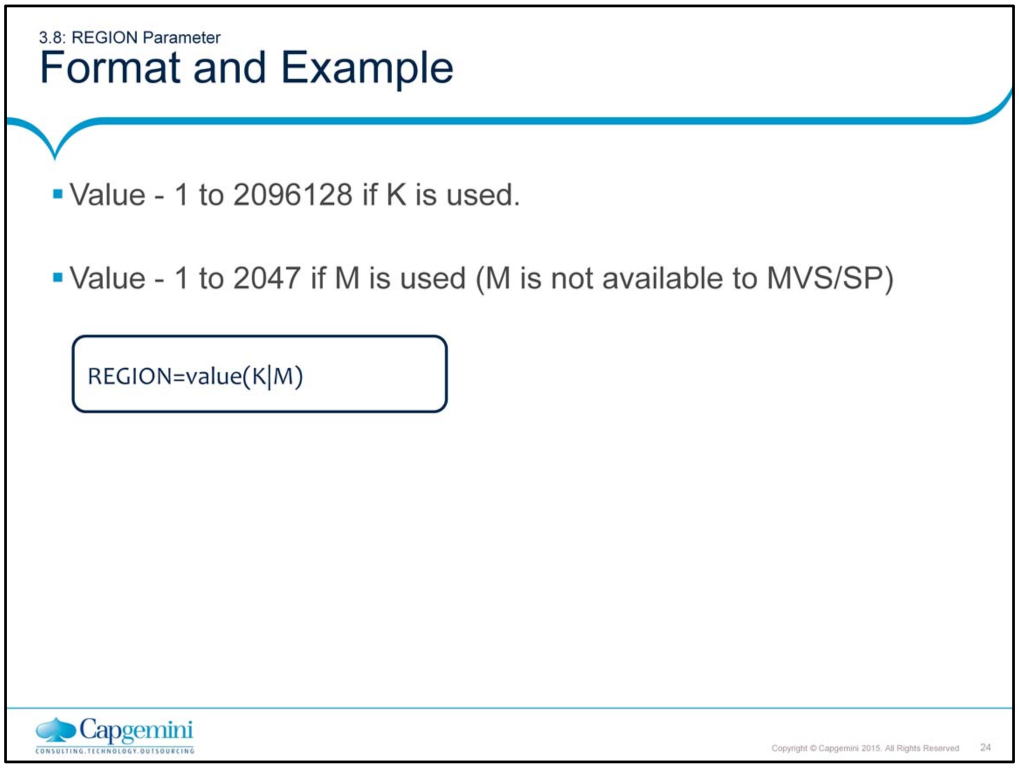
3.8: REGION Parameter

## Description

- REGION specifies the amount of storage (central / virtual) a job is allocated (Max. storage requirement).
- Maximum virtual memory available is 2GB.
  - Keyword parameter
  - Optional
  - Installation-defined

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

This parameter specifies the limit of available storage for each of the steps in the job within the job’s address space. i.e., the amount of storage the job is allocated. In other words, it specifies the amount of storage needed by the step (within the job) with the highest storage requirements.



3.8: REGION Parameter

## Format and Example

- Value - 1 to 2096128 if K is used.
- Value - 1 to 2047 if M is used (M is not available to MVS/SP)

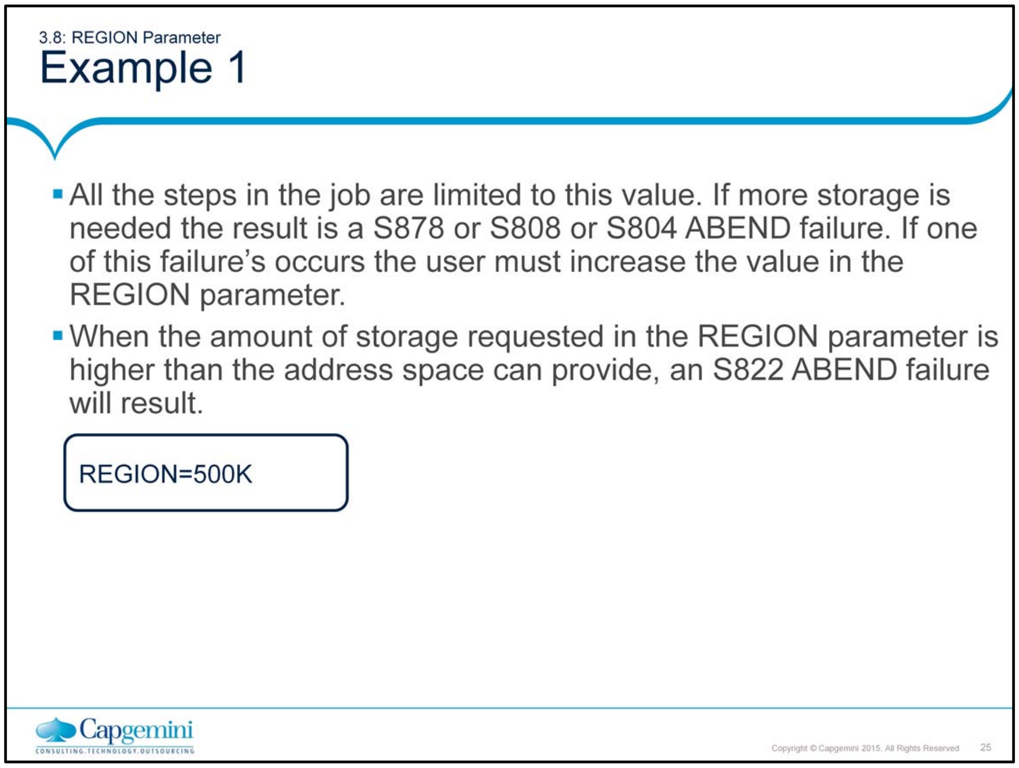
```
REGION=value(K|M)
```

24

General Syntax

```
   REGION=value{K|M}
keyword parameter
```

value – 1 to 2096128 if K (1024 bytes) is used. It should be an even number. If an odd number is used it will be rounded off to the next higher even number.
value – 1 to 2047 if M (1024K or 1048576 bytes) is used. M is not available to MVS/SP, only to MVS/XA and MVS/ESA.
When a job is selected by an initiator for execution, it is given an address space of 16 MB (minus what MVS/SP uses). In case of MVS/XA, job is given an address space of 2GB. And all of it is available to the job's steps. However a step normally requires only a small fraction of this huge storage, below the 16M line. An ordinary COBOL or any other language program seldom needs more than 1000K. This is normally what the value in the REGION parameter represents in the installations. Few jobs like CICS, IMS, DB2 need storage above the 16M line. An ordinary batch job seldom has such high requirements and, as a result confined to storage below the 16M line. Storage availability below this line varies in different installations, but is generally around 8MB in MVS/SP and around 9 MB in MVS/XA. Storage above the 16M line can be acquired by coding a value higher than 16M. However, it may be restricted by the installation to only those jobs that need it.



3.8: REGION Parameter

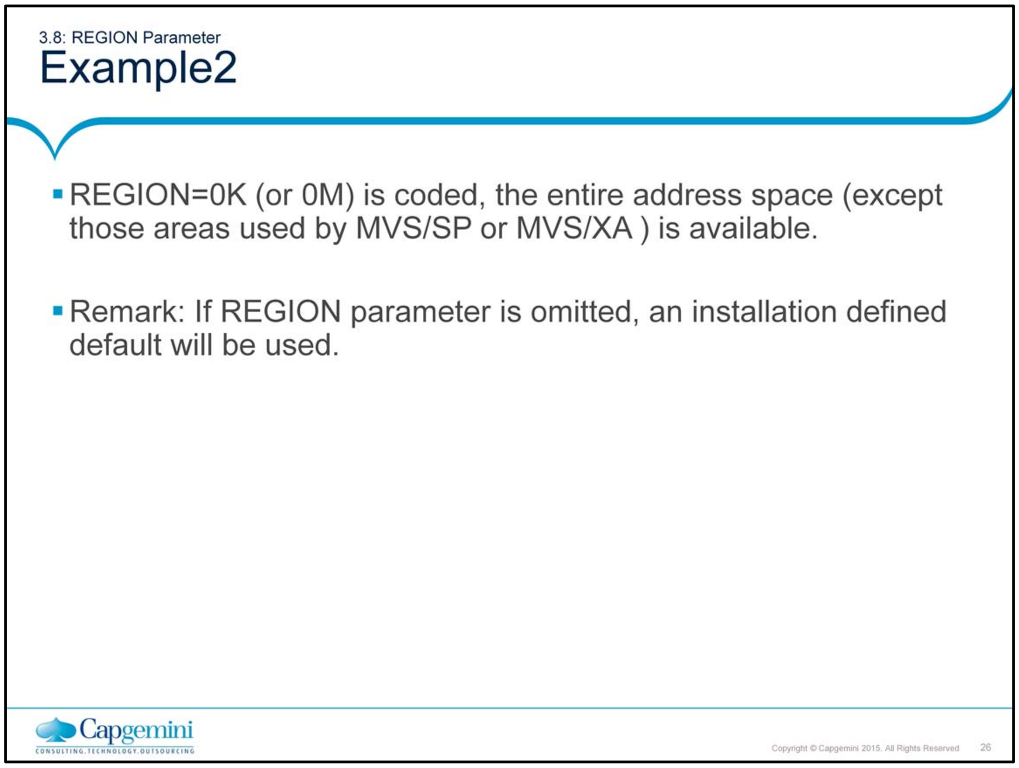
## Example 1

- All the steps in the job are limited to this value. If more storage is needed the result is a S878 or S808 or S804 ABEND failure. If one of this failure’s occurs the user must increase the value in the REGION parameter.
- When the amount of storage requested in the REGION parameter is higher than the address space can provide, an S822 ABEND failure will result.

```
REGION=500K
```

Examples:
Example 1:
Assume REGION=1000K were coded in the JOB statement. All the steps in the job are limited to this value. If more storage is needed, the usual result is S878 or S80A or S804 ABEND failure. If one of these failures occurs, the user must increase the value in the REGION parameter.

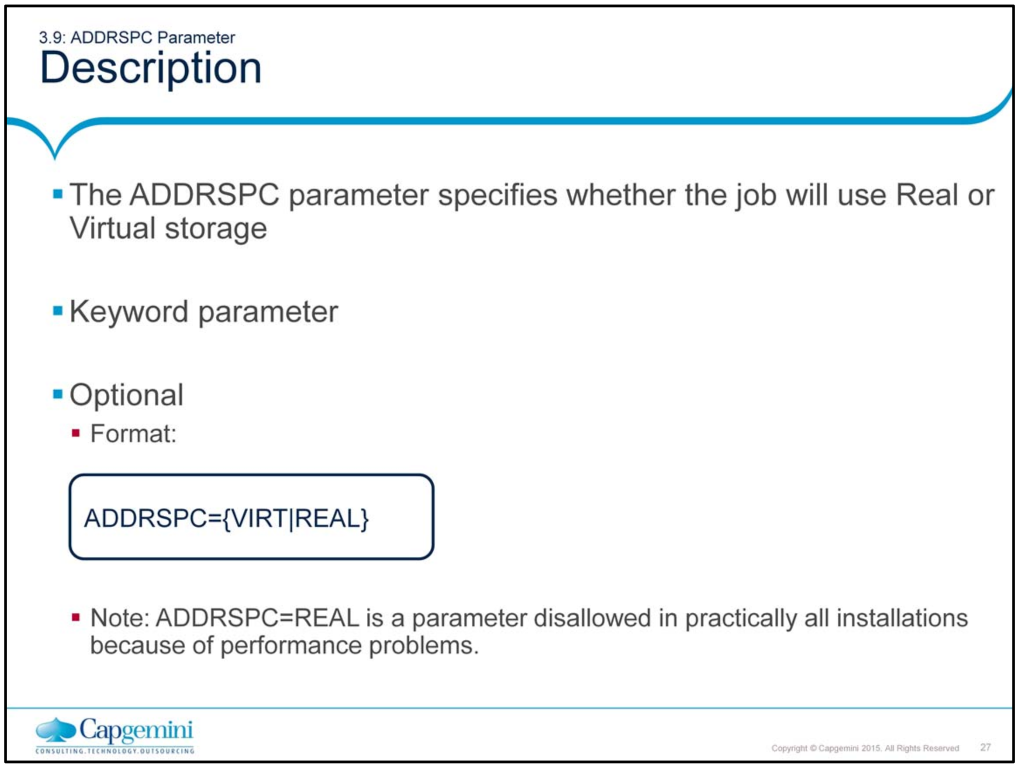
Example 2:
REGION=10M
When the amount of storage requested in the REGION parameter is higher than the address space can provide, an S822 ABEND failure occurs.



3.8: REGION Parameter

## Example2

- REGION=0K (or 0M) is coded, the entire address space (except those areas used by MVS/SP or MVS/XA ) is available.
- Remark: If REGION parameter is omitted, an installation defined default will be used.



3.9: ADDRSPC Parameter

## Description

- The ADDRSPC parameter specifies whether the job will use Real or Virtual storage
- Keyword parameter
- Optional
  - Format:

```
ADDRSPC={VIRT|REAL}
```

  - Note: ADDRSPC=REAL is a parameter disallowed in practically all installations because of performance problems.

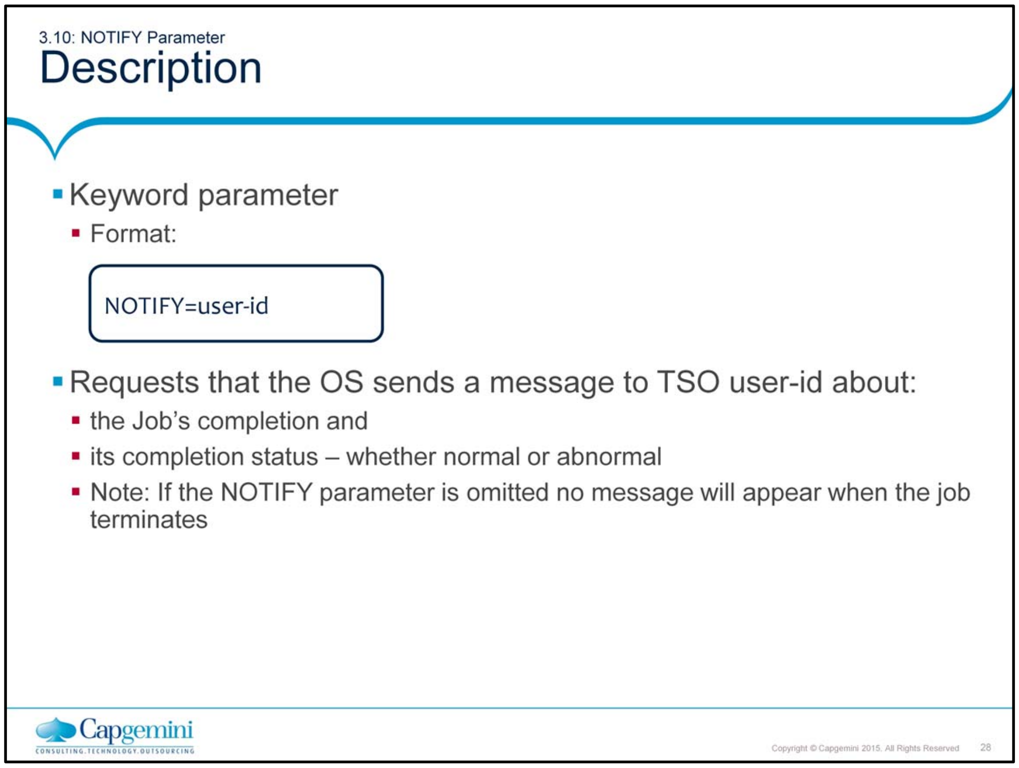
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

General Syntax:

```
ADDRSPC={VIRT|REAL}
```

ADDRSPC=REAL the allocation is done in REAL storage and the program is not page-able
ADDRSPC=VIRT the allocation is done in VIRTUAL storage and the program is page-able

Remark: This is the rarely used parameter because of the default. Note that ADDRSPC=REAL is a parameter that is disallowed in practically all installation because it can cause serious performance problems for other jobs.


3.10: NOTIFY Parameter

## Description

- Keyword parameter
  - Format:

```
NOTIFY=user-id
```

- Requests that the OS sends a message to TSO user-id about:
  - the Job's completion and
  - its completion status – whether normal or abnormal
  - Note: If the NOTIFY parameter is omitted no message will appear when the job terminates

General Syntax

```
NOTIFY=userid                keyword parameter
```

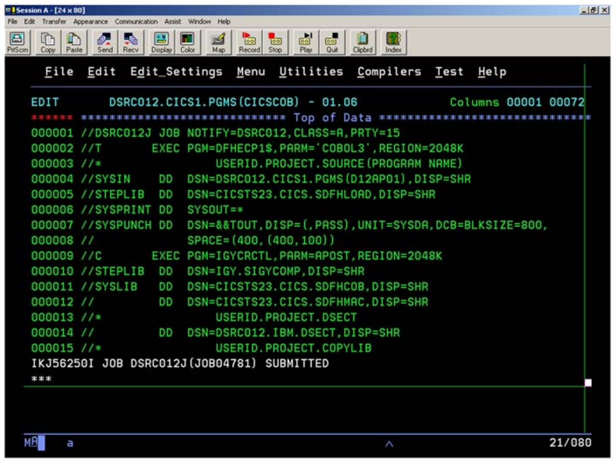
userid – A name from 1 to 7 characters, identifying a valid TSO user.
Example: NOTIFY=DA0001T

If coded, a message will appear on the user's TSO terminal indicating if the job abended or got a JCL error. If the job terminates while the user was logged off, the message will appear when the user logs on. If the NOTIFY parameter is omitted, no message will appear when the job terminates.

3.10: NOTIFY Parameter

## The JOB Statement – NOTIFY

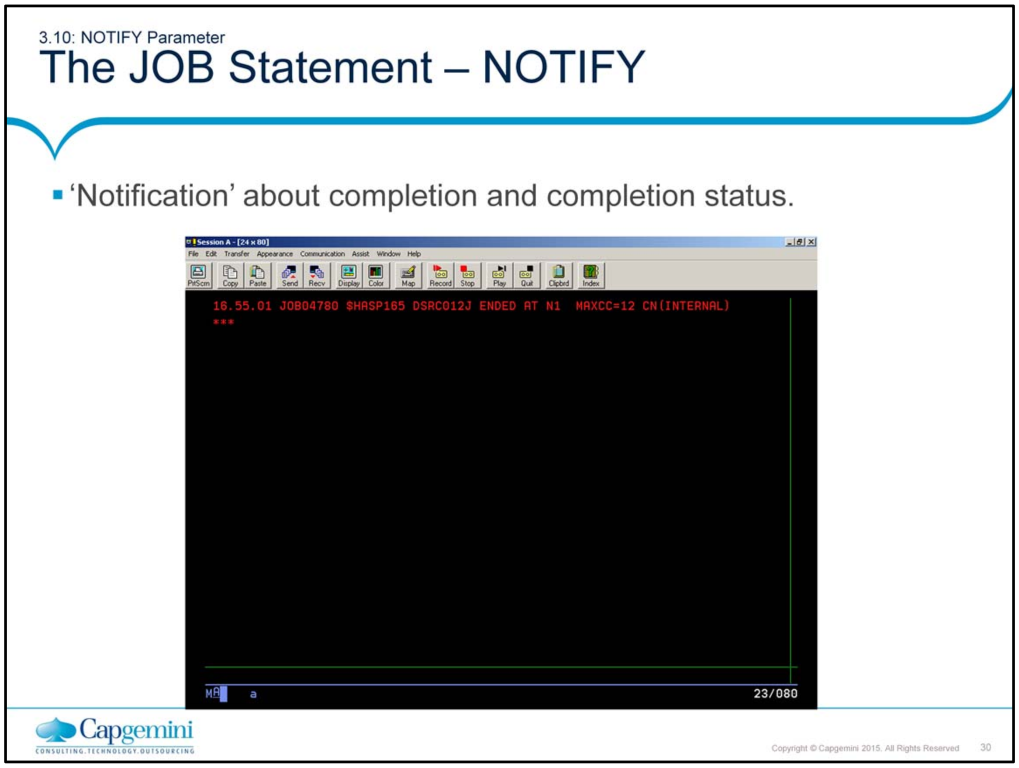
- The OS giving your job a number (JobID) when submitted.


Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
PrtScrn Copy Paste Send Recv Display Color Map Record Stop Play Quit Clipbrd Index

```
   File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       DSRC012.CICS1.PGMS(CICSCOB) - 01.06            Columns 00001 00072
****** **************************** Top of Data *****************************
//DSRC012J JOB NOTIFY=DSRC012,CLASS=A,PRTY=15
//T        EXEC PGM=DFHECP1$,PARM='COBOL3',REGION=2048K
//*                 USERID.PROJECT.SOURCE(PROGRAM NAME)
//SYSIN     DD  DSN=DSRC012.CICS1.PGMS(D12AP01),DISP=SHR
//STEPLIB   DD  DSN=CICSTS23.CICS.SDFHLOAD,DISP=SHR
//SYSPRINT  DD  SYSOUT=*
//SYSPUNCH  DD  DSN=&&TOUT,DISP=(,PASS),UNIT=SYSDA,DCB=BLKSIZE=800,
//              SPACE=(400,(400,100))
//C        EXEC PGM=IGYCRCTL,PARM=APOST,REGION=2048K
//STEPLIB   DD  DSN=IGY.SIGYCOMP,DISP=SHR
//SYSLIB    DD  DSN=CICSTS23.CICS.SDFHCOB,DISP=SHR
//          DD  DSN=CICSTS23.CICS.SDFHMAC,DISP=SHR
//*                 USERID.PROJECT.DSECT
//          DD  DSN=DSRC012.IBM.DSECT,DISP=SHR
//*                 USERID.PROJECT.COPYLIB
IKJ56250I JOB DSRC012J(JOB04781) SUBMITTED
***

MA    a                                          ^                    21/080
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

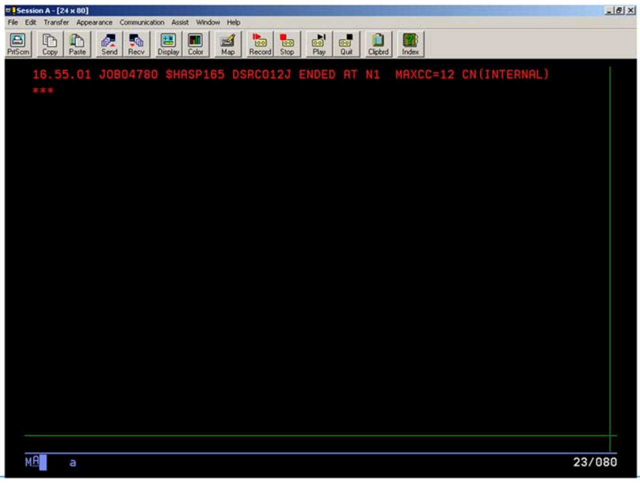
29



3.10: NOTIFY Parameter

## The JOB Statement – NOTIFY

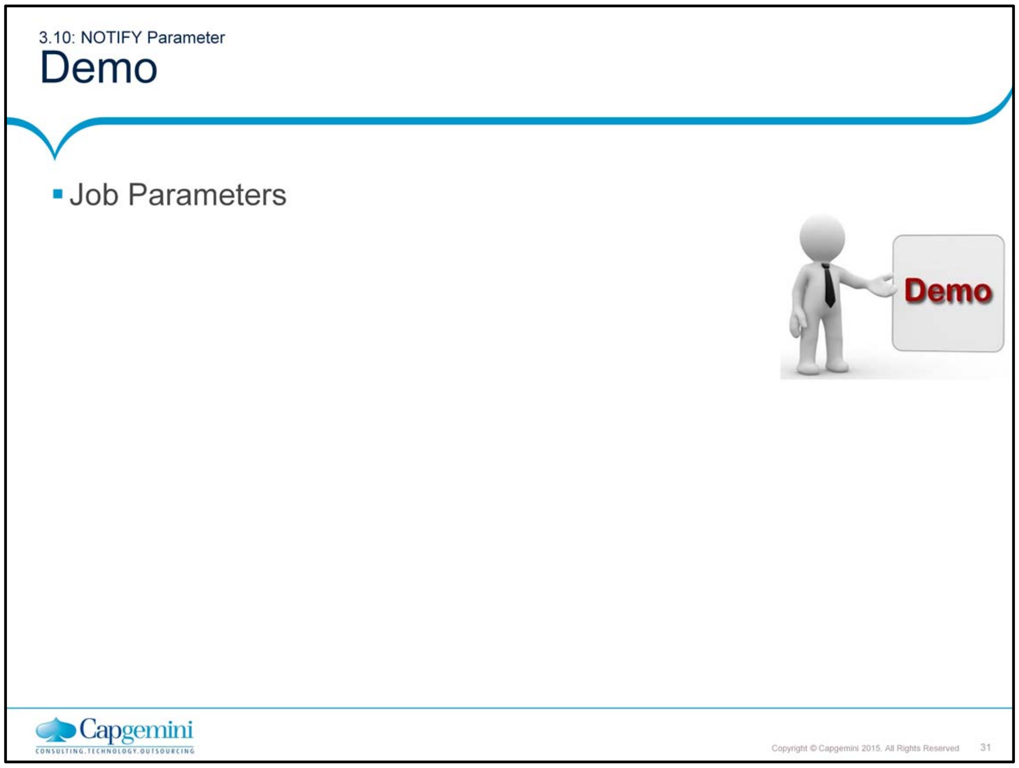
- 'Notification' about completion and completion status.



Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
PrtScrn Copy Paste Send Recv Display Color Map Record Stop Play Quit Clipbrd Index

```
16.55.01 JOB04780 $HASP165 DSRC012J ENDED AT N1  MAXCC=12 CN(INTERNAL)
***
```

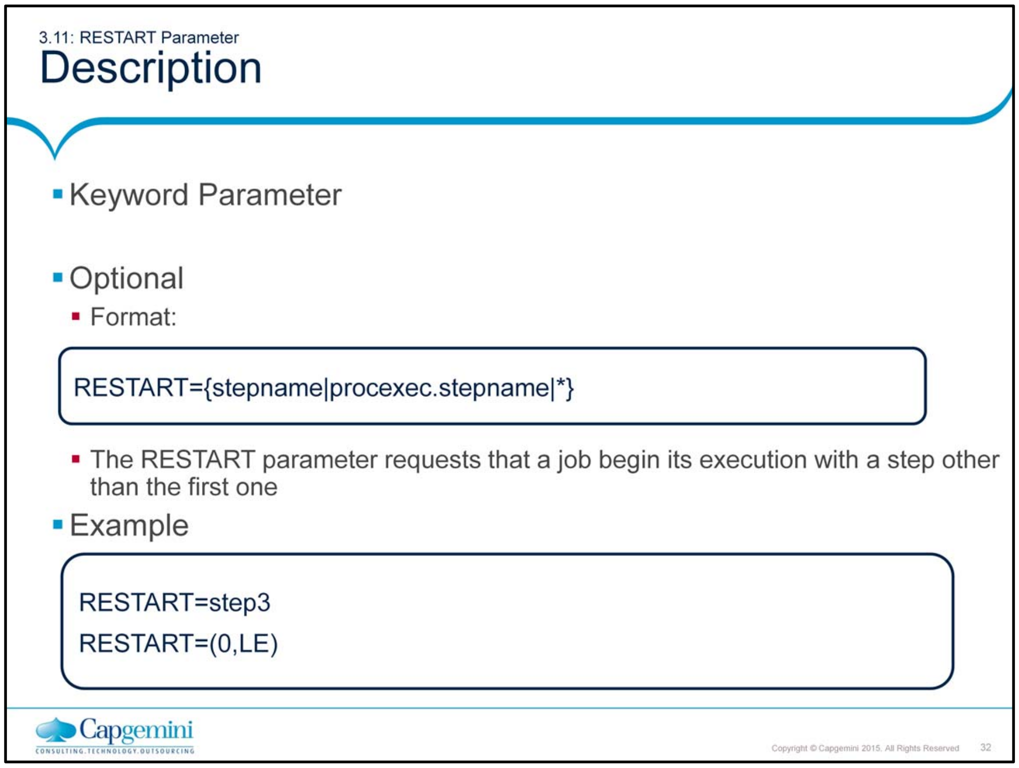
```
MA     a                                                     23/080
```



3.10: NOTIFY Parameter

## Demo

- Job Parameters


3.11: RESTART Parameter

## Description

- Keyword Parameter
- Optional
  - Format:

```
RESTART={stepname|procexec.stepname|*}
```

  - The RESTART parameter requests that a job begin its execution with a step other than the first one
- Example

```
RESTART=step3
RESTART=(0,LE)
```

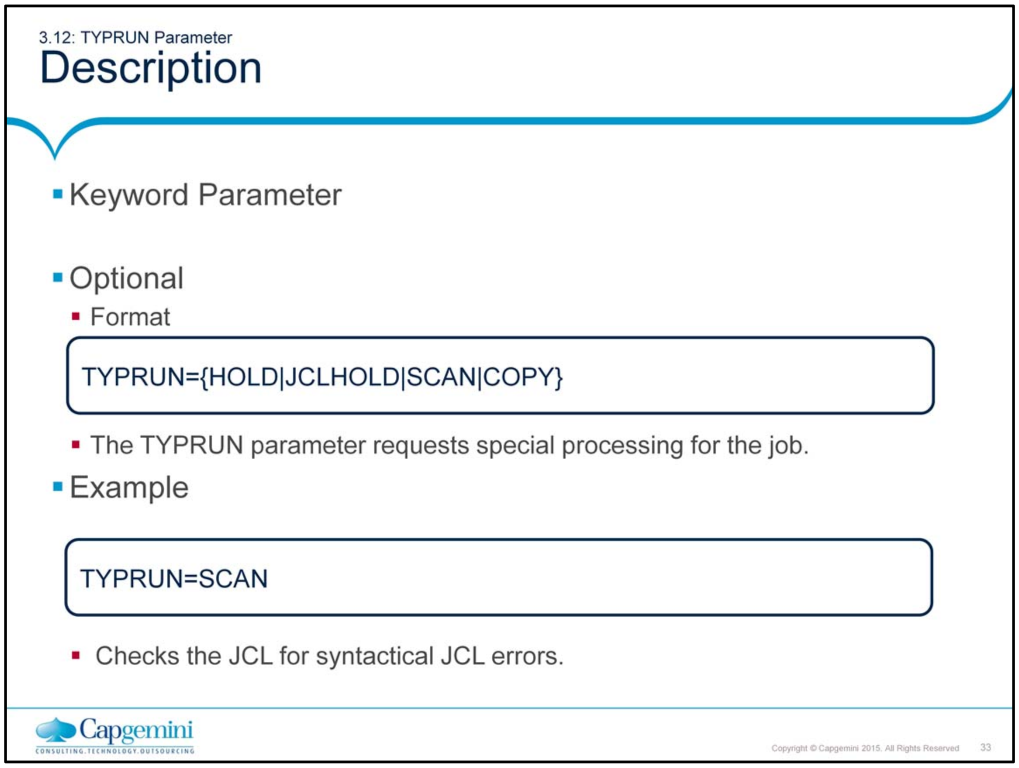
General Syntax:

```
RESTART={stepname|procexec.stepname| *}     keyword parameter
```

Stepname – The name of the step where execution is to begin.
procexec.stepname – The name of the EXEC statement invoking a procedure and the name of the step within the procedure where execution is to begin.

- Indicates that execution of the job is to begin with the first step and is the default.

Things to avoid:
  Duplicate names for EXEC statements invoking procedure
   If RESTART=procsexec.stepname is used, the first procexec found is used.
  Duplicate stepnames within procedure.
   If  RESTART=procsexec.stepname is used, the first stepname within procexec found is used.
  Duplicate stepnames.
   If RESTART=stepname is used, the first stepname  found is used.
  EXEC statements (invoking procedures or any step) without names.
   No restart is possible.



3.12: TYPRUN Parameter

## Description

- Keyword Parameter
- Optional
  - Format

```
TYPRUN={HOLD|JCLHOLD|SCAN|COPY}
```

  - The TYPRUN parameter requests special processing for the job.
- Example

```
TYPRUN=SCAN
```

  - Checks the JCL for syntactical JCL errors.

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

General Syntax

```
TYPRUN={HOLD|JCLHOLD|SCAN|COPY}          keyword parameter
```

HOLD - Job is held held (and not executed temporarily) until the operator uses a command to release. A job is held in the input queue only if syntactically correct.

JCLHOLD (JES2 only) – Job is held (and not executed) until the operator uses a command to release it. Note the job is held in queue even if it is syntactically incorrect. This option is rarely used.

SCAN – Job is scanned for all syntactical JCL errors but will not execute.

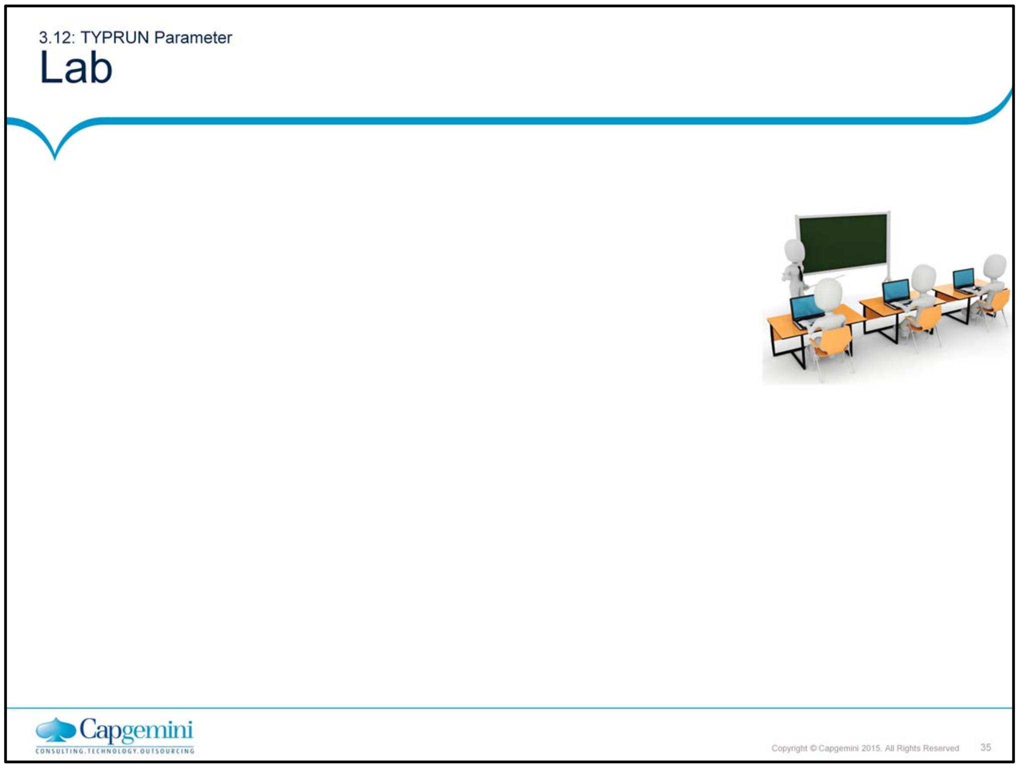
COPY (JES2 only)- Job will be printed. No execution and no syntax checking takes place. This option is also rarely used.

The following JOB statement illustrates the use of the parameters relevant to the JOB statement:

```
//DA0001T JOB LA719,PAI,MSGCLASS=A,MSGLEVEL=(1,1),PRTY=5,
//CLASS=B,REGION=0M,TIME=(0,1),NOTIFY=DA0001T
```

Example: If the compile, link and run steps are given in one JCL and subsequently the execution has to begin from the run step we can give:

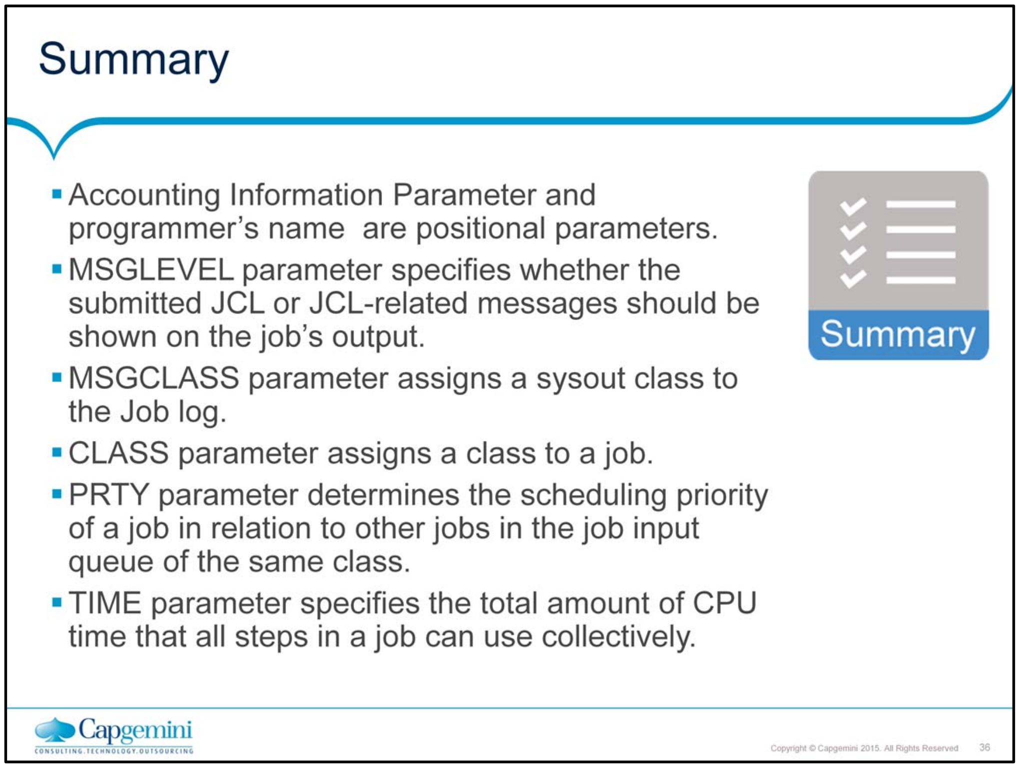
```
//DA0001TC  JOB  LA2719,'SHEELA',NOTIFY=DA0001T,
//     MSGCLASS=X,TIME=(0,1),RESTART=COBRUN
//*************************************************************
//* STEP TO COMPILE A PROGRAM
//* COMPILER PROGRAM NAME - IKFCBL00
//* LIBRARY NAME - SYS1.COBCOMP
//* SYSLIN - OUTPUT FILE NAME
//* SYSIN  - INPUT FILE NAME (I.E. COBOL PROGRAM NAME)
//* SYSUT1,2,3, -  TEMPORARY FILES REQUIRED BY COBOL
COMPILER
//*************************************************************
//COB       EXEC PGM=IKFCBL00,REGION=1024K,
// PARM='NOTRUNC,NODYNAM,LIB,SIZE=4096K,BUF=116K
,APOST,NORES
//SYSLIB      DD  DSN=SYS1.COBCOMP,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLIN      DD DSN=&&TEMP,DISP=(NEW,PASS),
//                  UNIT=SYSALLDA,SPACE=(TRK,(40,40))
//SYSUT1  DD  UNIT=SYSALLDA,SPACE=(TRK,(6,1))
//SYSUT2  DD  UNIT=SYSALLDA,SPACE=(CYL,(6,1))
//SYSUT3  DD  UNIT=SYSALLDA,SPACE=(CYL,(6,1))
//SYSUT4  DD  UNIT=SYSALLDA,SPACE=(CYL,(6,1))
//SYSIN   DD  DSN=DA0001T.SHEELA.COBOL(PRG1),DISP=SHR
//*****************************************************
//* STEP TO LINK THE COBOL PROGRAM
//* LINKER PROGRAM NAME - IEWL
//* LIBRARY NAME - SYS1.COBLIB
//* SYSLMOD - OUTPUT DATASET NAME
//* SYSLIN - INPUT DATASET NAME
//*****************************************************
//LKED  EXEC  PGM=IEWL,PARM='LIST,XREF,LET,MAP',
//  REGION=4096K,COND=(0,LT,COB)
//SYSLIN  DD  DSN=&&TEMP,DISP=(OLD,DELETE)
//SYSLIB  DD  DSN=SYS1.COBLIB,DISP=SHR
//SYSLMOD DD  DSN=DA0001T.SHEELA.LOADLIB(PRG1),
//  DISP=SHR,UNIT=SYSALLDA
//SYSUT1  DD  UNIT=SYSALLDA,SPACE=(1024,(50,20))
//SYSPRINT DD SYSOUT=*
//*
//*STEP TO RUN COMPILED COBOL PROGRAM
//COBRUN  EXEC PGM=PRG1
//STEPLIB DD DSN=DA0001T.SHEELA.LOADLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//INF1    DD DSN=DA0001T.EMPDATA,DISP=SHR
//OTF1   DD DSN=DA0001T.L3,DISP=(NEW,CATLG,DELETE),
//       UNIT=SYSDA,SPACE=(TRK,(1,1)),
//       DCB=(LRECL=80,RECFM=FB,BLKSIZE=800,DSORG=PS)
//SYSOUT DD SYSOUT=*
```



3.12: TYPRUN Parameter

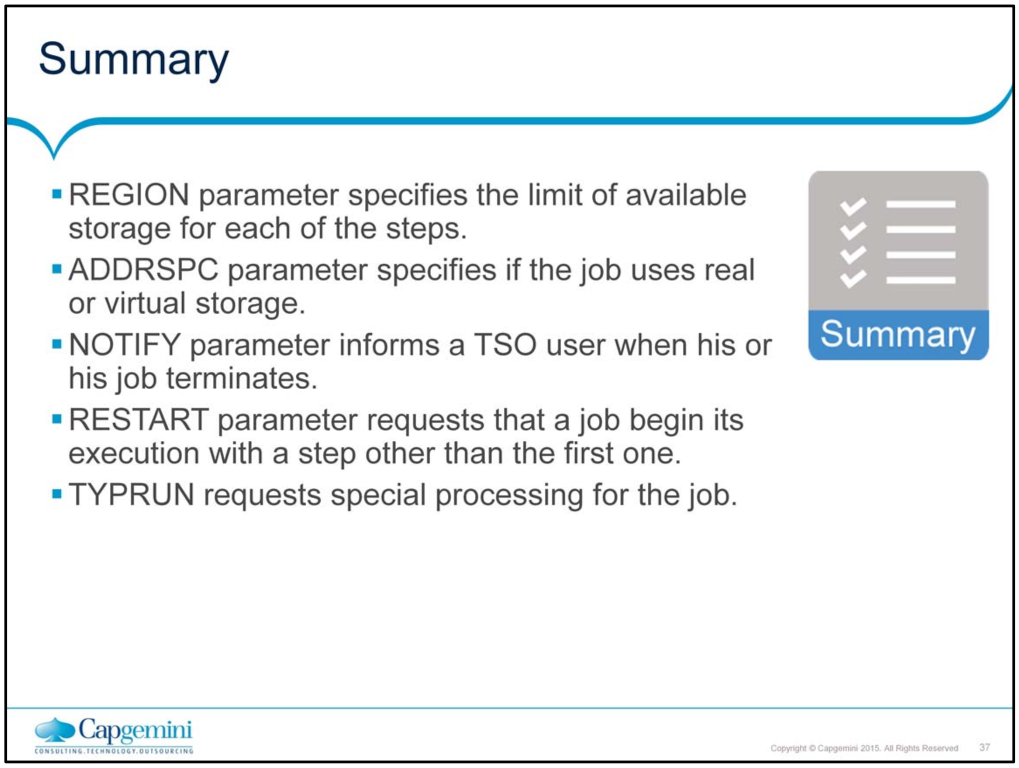
## Lab

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING



## Summary

- Accounting Information Parameter and programmer's name  are positional parameters.
- MSGLEVEL parameter specifies whether the submitted JCL or JCL-related messages should be shown on the job's output.
- MSGCLASS parameter assigns a sysout class to the Job log.
- CLASS parameter assigns a class to a job.
- PRTY parameter determines the scheduling priority of a job in relation to other jobs in the job input queue of the same class.
- TIME parameter specifies the total amount of CPU time that all steps in a job can use collectively.



## Summary

- REGION parameter specifies the limit of available storage for each of the steps.
- ADDRSPC parameter specifies if the job uses real or virtual storage.
- NOTIFY parameter informs a TSO user when his or his job terminates.
- RESTART parameter requests that a job begin its execution with a step other than the first one.
- TYPRUN requests special processing for the job.