UNIT 2

The JOB Statement

The JOB Statement

- Objectives
- Purpose of the JOB statement
- Syntax
- Parameters
  - Accounting information and Programmer name
  - CLASS
  - MSGCLASS
  - NOTIFY
  - MSGLEVEL
  - REGION
  - TIME
  - RESTART
  - TYPRUN
- Examples

Objectives

- Understand what the JOB statement accomplishes
- Learn the most important parameters for the JOB statement
- Code different JOB statements depending on the need of the Job

Purpose of the JOB statement

The JOB statement is needed for the following reasons

- To identify the Job and the person who submitted the Job.
- Specify which Class the Job is to belong to.
- Specify how the output should be handled.
- To make an entity or group accountable for the execution of the Job.

The JOB statement should always be the first statement in a Job.

You can code more than one Job in the same member, though it is not usually recommended.

Syntax

The JOB statement accepts both positional and keyword parameters.

Example:

```
//ABC    JOB  'MY ACCOUNT','PROGRAMMER NAME',CLASS=A
```

- You can use the same Jobname for more than one of your Jobs.
- When a job is submitted to the CPU, it is assigned a unique JES number. Thus, if multiple jobs with the same Jobname are submitted to the CPU, they are tracked individually by their JES number.

Parameters – Accounting Information and Programmer Name

Parameter Type: Positional parameters

Accounting information and Programmer‘s name are both positional parameters. Accounting information appears first followed by Programmer name.

Accounting Information: Identifies the account code to be used for the job. This is primarily used for billing purposes.

An installation dependent value ranging from 1 to 143 characters.

A comma will replace absence of the accounting Information field.

Programmer Name: Identifies the person or department who is responsible for the job.

A character string ranging from 1 to 20 characters.

A comma will replace absence of the Programmer Name field.

Appears at the bottom of every page of printed output for the Job.

Examples:

```
//MYJOB    JOB  MTPLACC25,’JOHN SMITH’,…….

//YOURJOB  JOB  ,’SUSIE JOHNSON’,….

//HISJOB   JOB  MTPLAC30,,……..
```

Parameter – CLASS

Type: Keyword parameter

Purpose: To assign a Class to the Job.

Syntax:

```
//JOBNAME   JOB  ACCTINFO,'PROGRAMMER NAME',CLASS=A,..
```

CLASS=Class

$\downarrow$

(1 Character. Installation dependent. ‘A’ thru ‘Z’ and ‘0’ thru ‘9’)

Example:

```
//ABC    JOB  ACCT123,'PROGRAMMER NAME',CLASS=A,…….
```

Job ABC will execute in class ‘A’.

- As we have seen in Unit 1, the Class of a Job determines the input priority (a basis for Initiator selection)
- As most installations do not prefer the users specifying the dispatching priority for their Jobs (a different parameter), the CLASS parameter provides a less explicit way to specify the input priority. For example, the user might select a Class that most Initiators can handle.

Parameter – MSGCLASS

Type: Keyword parameter

Purpose: To specify how the Job’s output is to be handled.

Syntax:

```
//JOBNAME   JOB  ACCTINFO,’PROGRAMMER NAME’,CLASS=A,MSGCLASS=A..
```

MSGCLASS=Message-class

$\downarrow$

(1 character. Installation Dependent. ‘A’ thru ‘Z’ of ‘0’ thru ‘9’)

Example:

```
//ABC    JOB  ACCT123,’PROGRAMMER NAME’,MSGCLASS=A
```

- Using the MSGCLASS parameter, you can tell the system how to handle the output of your Job.
- The valid values that you can specify for this parameter are dependent on your installation.
- In one installation, MSGCLASS=Z may mean that the output has to be immediately routed to a printer. In another installation, it may mean that the output has to be held in the Output Queue for later retrieval.

Parameter – NOTIFY

Type: Keyword parameter

Purpose: To specify which user is to be notified after the Job finishes execution.

Syntax:

```
//JOBNAME    JOB  ACCT123,'PROGRAMMER NAME',NOTIFY=IBMUSER
```

NOTIFY=User-Id

$\downarrow$

(Any valid User ID defined to your security system)

Examples:

```
//ABC    JOB  ACCT123,'PROGRAMMER NAME',NOTIFY=IBMUSER
```

This example will notify the user id MAIN006 when the job completes

```
//ABC    JOB  ACCT123,'PROGRAMMER NAME',NOTIFY=&SYSUID
```

The value '&SYSUID' indicates that the user id, which submitted the job, should be notified when the job completes.

Comment [T1]: Is this here for a reason?

- The NOTIFY parameter can used be when someone needs to be notified when a job completes, along with its maximum return-code (discussed later) or Abend code.
- Once the Job finishes, if the user specified in NOTIFY has logged on, he or she will get the message once he or she presses ENTER or any other Attention Identifier Key. If not logged on, the user will get the message(s) along with the logon prompts during the next login procedure.

Parameter – MSGLEVEL

Type: Keyword parameter accepting 2 positional parameters.

Purpose: To specify how much output has to be generated for this Job by the system.

Syntax:

```
//ABC    JOB  ACCT123,'PROGRAMMER NAME',MSGLEVEL=(a,b)
```

MSGLEVEL=(a,b)

$\downarrow$

a: Specifies how much of the Job has to be printed in the output (job log).
Valid values are 0, 1 and 2
- 0 – Print only the Jobcard (JOB statement of the Job)
- 1 – Print all JCL and JES statements including all statements in procedures, if any
- 2 – Print only submitted JCL and JES statements. Statements in procedures are not printed.

b: Specifies which messages will be printed in the output (job log).
Valid values are 0 and 1
- 0 – Print the entire messages only if the Job abends (abnormal termination).
- 1 – Print all the messages regardless of the outcome of the Job how the job terminates

Examples:

```
//ABC    JOB  ACCT123,'PROGRAMMER NAME',MSGLEVEL=(1,1)

//ABC    JOB  ACCT123,'PROGRAMMER NAME',MSGLEVEL=(,1)

//ABC    JOB  ACCT123,'PROGRAMMER NAME',MSGLEVEL=(2,)
```

Parameter – REGION

Type: Keyword parameter

Purpose: To limit the maximum amount of memory that the Job can utilize.

Syntax:

```
    //JOBNAME   JOB   ACCTINFO,'PROGRAMMER NAME',REGION=nnnnM
```

REGION=nnnnnnnK
or
REGION=nnnnM

n: Numeric value
Valid ranges:
0 thru 2047000 in case of K
0 thru 2047 in case of M
K: Kilobytes   M: Megabytes

Examples:

```
//ABC    JOB  ACCT123,'PROGRAMMER NAME',REGION=1024K

//ABC    JOB  ACCT123,'PROGRAMMER NAME',REGION=1M
```

- REGION need not be explicitly coded unless specified in the documentation for certain utilities.
- If REGION=0K or REGION=0M is coded, the system assumes a default region size.

Parameter – TIME

Type: Keyword parameter

Purpose: The TIME parameter can be used to specify the maximum length of CPU time that a job or job step is to use the processor.

Syntax:

```
//JOBNAME  JOB  ACCTINFO,'PROGRAMMER NAME',TIME(m,ss)
```

TIME = (minutes, seconds)

ON JOB STATEMENT
Indicates the maximum processing time for the JOB.

ON EXEC STATEMENT
Indicates the maximum processing time for the STEP.

Special Values

Any combination of minutes and seconds can be coded. However, there are three values for the parameter that have a special meaning:

TIME=NOLIMIT	means the job or step is not to be timed.
TIME=1440	is equivalent to coding TIME=NOLIMIT, indicating that the job will not be timed. Thus 1440 is the only numerical value of TIME that is not to be interpreted literally.
TIME=MAXIMUM	means the job or step will be allowed to use to 357912 minutes. (This is about 8.25 months of processor time)

- If coded on the JOB statement and any step causes the total time to be exceeded, the job is abnormally terminated. If not coded, the JES installation default is used.

Examples - TIME Parameter

```
//JOBNAME  JOB    TIME=(1,30)
```

TERMINATE JOB IF TOTAL TIME FOR ALL STEPS EXCEEDS 1 MINUTE AND 30 SECONDS

```
//STP1     EXEC   PGM = ABC, TIME=1
```

```
//STP2     EXEC   PGM=BAKER, TIME=(,20)
```

TERMINATE THIS STEP IF TIME EXCEEDS 20 SECONDS

```
//STP3     EXEC   PGM=CHARLTE, TIME=NOLIMIT
```

DO NOT TIME THIS STEP

- The entire job will be limited to a maximum of 1 minute and 30 seconds. The first step will be limited to a maximum of 1 minute.
- Note that the submitter is apparently asking that the last step not be timed. However, coding TIME=NOLIMIT on the last step will NOT accomplish this objective. The TIME=(1,30) on the JOB statement will limit the maximum time for step 3 to 1 minute and 30 seconds.
- If the time consumed by a job (or step) exceeds the value specified for the TIME parameter for that job (or step), the job ABENDS.

Parameters – RESTART

Type: Keyword parameter

Purpose: The RESTART parameter allows you to begin the execution of the Job from a Step other than the first one.

Syntax:

```
    //JOBNAME JOB ACCTINFO,’PROGRAMMER NAME’,RESTART=Stepname
```

Example:

```
//ABC     JOB ACCT123,’PROGRAMMER NAME’,RESTART=STEP2
```

- If you want to skip one or more of the initial steps in the Job before starting execution, use the RESTART parameter.
- For example, if you have run a Job with 2 steps, the first step runs and the second step abends. You make the changes in the second step to set right any error causing statement or parameter. Now, you will require running only the second step. Therefore, you change the JOB statement and add the RESTART parameter to start execution from the second step onwards.

Parameter – TYPRUN

Type: Keyword parameter

Purpose: The TYPRUN parameter controls the type of execution for the Job.

Syntax:

```
//JOBNAME JOB  ACCTINFO,’PROGRAMMER NAME’,TYPRUN=typerun,…
```

TYPRUN=SCAN	Only check the Job for syntax errors. Do not execute the Job. This is a common way to debug errors in your JCL. The output, including error messages, if any, is directly sent to the Output Queue.
TYPRUN=HOLD	Check the Job for syntax errors and, if no errors are present, hold it in the Input Queue. The job will execute when a person (usually an operator) or another program releases the hold.
TYPRUN=COPY	With COPY, no syntax checking or execution takes place. The source content of the Job immediately gets directed to the Output Queue. The JCL is copied as submitted to the sysout class specified in the MSGCLASS parameter (JES 2 ONLY).

Example:

```
//JOB1  JOB  ACCT2254,’MARK JOHN’,TYPRUN=SCAN,…

//JOB2  JOB  ACCT2112,’MARK JOHN’,TYPRUN=HOLD,…

//JOB3  JOB  ACCT634,’MARK JOHN’,TYPRUN=COPY,…
```

Examples of JOB statements

Following are a few sample JOB statements

SAMPLE1

```
//USERJOB  JOB  ACCT54,'JOE SMITH',TYPRUN=SCAN,
//     MSGCLASS=X,CLASS=A,
//     MSGLEVEL=(1,1)
```

SAMPLE2

```
//MAINJOB1  JOB  ,,RESTART=STEP3,
//     REGION=4M
```

SAmPLE3

```
//TECJOB41  JOB  ,'SMITH',
//    TIME=1440,
//    MSGLEVEL=(2,0)
```

- It is a good practice to code individual parameters on separate lines. If one of the parameters needs to be cancelled, all you need to do is comment out the line containing the parameter (by putting an '*' in column 3).

Unit 2 Exercises

Complete the following:

1. The _________________ statement should always be the first statement of the job.

2. What parameter would you add to the JOB card to indicate that the job should be held in the Input Queue when it is submitted to the CPU?

3. What parameter, when added to the JOB card, would indicate that all of the steps of the job must complete within 1 hour?

4. What parameter indicates that the job should execute in any initiator that contains class 'X'?

5. Match the following fields with their corresponding descriptions:

Fields	Description
1. ____CLASS .	A Specifies which user is to be notified upon job completion
2. ____MSGCLASS	B. Indicates that the Job should start at the indicated step (other than the first step)
3. ____NOTIFY	C. Controls the type of execution for the Job
4. ____MSGLEVEL	D. Limits the amount of memory that a Job can utilize
5. ____REGION	E. Assigns a class to a Job
6. ____TIME	F. Indicates the maximum processor time that a Job can use
7. ____RESTART	G. Indicates how much output is to be generated for the Job
8. ____TYPRUN	H. Specifies how the output of a job is to be handled

6. Use the following information and code a job statement for each set of criteria:

a) JOBNAME	-	APJ100
ACCOUNT #	-	345
PGMR NAME	-	WILMA
MESSAGE CLASS	-	E
PRINT ALL JCL STATEMENTS		
PRINT ALL MESSAGES		

____

____

b)

JOBNAME	-	APJ200
ACCOUNT #	-	834
PGMR NAME	-	FRED
MESSAGE CLASS	-	A
PRINT ONLY JOB STATEMENT		
NO MESSAGES		

c)

JOBNAME	-	APJ300
ACCOUNT #	-	235-65
PGMR NAME	-	BARNEY
NOTIFY	-	USERID
MESSAGE CLASS	-	D
PRINT ALL JCL STATEMENTS		

Unit 2 Lab Exercises

Logon to TSO/ISPF and perform the following exercises. Wherever you see "userid" in lower case, substitute your valid security userid.

1. **Create a training library for your userid.**
   - **A. Allocate a PDS called 'userid.JCL.CNTL', where 'userid' is your valid userid.   All the exercises should be carried out in separate members in this library.**
2. **Create a JCL deck with a valid JOB card.**
   - **A. Create a member your library called JOBTEST1.**
   - **B. In member JOBTEST1, type the following JCL, where:**

'account number'	**is a valid account number for your site**
'your name'	**is your actual name**
class	**should be a valid execution class for your site**
msgclass	**is a the 'HOLD' message class for your site, normally used for testing purposes**
userid	**is your userid**
SYSDA	**if SYSDA creates an error, use a UNIT name that fits your site JCL standards**

```
//useridA JOB account number,'your name',
//              MSGLEVEL=(1,1),
//              CLASS=class,
//              MSGCLASS=msgclass,
//              NOTIFY=userid
//*
//* A line that starts with '//*' is a comment line…
//*
//* Substitute your USERID for all occurrence of USERID
//* Substitute any number for your account number.
//* Your name will be the programmer's name
//*
//STEP1         EXEC PGM=IEFBR14
//DD1      DD   DSN=userid.XYZ.CNTL,DISP=(NEW,CATLG,DELETE),
//              UNIT=SYSDA,SPACE=(TRK,(2,1,5),RLSE),
//              DCB=(BLKSIZE=800,LRECL=80,RECFM=FB)
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DUMMY
//
```

C. Type SAVE in the command line to save this member.

D. Type SUBMIT in the command line to submit this job the CPU.

E. Type =3.8 on the command line and view the job held output for this JCL.

F. Review the job for errors. If there are any, return to the JCL and fix them before continuing.

<u>SDSF (System display and search facility):</u>

A. This same JES JOB output can be seen by SDSF (system display and search facility) option.

B. Type TSO SDSF ST on the command line.

C. Type S on the beginning of the line corresponding to your Job=Id.

D. You can also view the status of the JOB by typing a command line command i.e.=S;ST and selecting the specific JOB-ID.

E. You can execute a command line command for displaying the active users on the system by typing =S.DA.

F. You can execute a command line command for displaying the jobs held in the input queue by typing =S.I.

G. You can execute a command line command for displaying the jobs held in the output queue by typing =S.O.