JCL - JOB STATEMENT

JOB Statement is the first control statement in a JCL. This gives the identity of the job to the Operating System *OS*, in the spool and in the scheduler. The parameters in the JOB statement help the Operating Systems in allocating the right scheduler, required CPU time and issuing notifications to the user.

Syntax

Following is the basic syntax of a JCL JOB statement:

```
//Job-name JOB Positional-param, Keyword-param
```

Description

Let us see the description of the terms used in above JOB statement syntax.

Job-name

This gives an id to the job while submitting it to the OS. It is can be length of 1 to 8 with alphanumeric characters and starts just after //.

JOB

This is the keyword to identify it as a JOB statement.

Positional-param

There are positional parameters, which can be of two types:

Positional Parameter	Description
Account information	This refers to the person or group to which the CPU time is owed. It is set as per the rules of the company owning the mainframes. If it is specified as ∗, then it takes the id of the user, who has currently logged into the Mainframe Terminal.
Programmer name	This identifies the person or group, who is in charge of the JCL. This is not a mandatory parameter and can be replaced by a comma.

Keyword-param

Following are the various keyword parameters, which can be used in JOB statement. You can use one or more parameters based on requirements and they are separated by comma:

Keyword Parameter	Description
CLASS	Based on the time duration and the number of resources required by the job, companies assign different job classes. These can be visualized as individual schedulers used by the OS to receive the jobs. Placing the jobs in the right scheduler will aid in easy execution of the jobs. Some companies have different classes for jobs in test and production environment. Valid values for CLASS parameter are A to Z characters and 0 to 9

numeric *oflength*1. Following is the syntax:

CLASS=0 to 9 | A to Z

PRTY

To specify the priority of the job within a job class. If this parameter is not specified, then the job is added to the end of the queue in the specified CLASS. Following is the syntax:

PRTY=N

Where N is a number in between 0 to 15 and higher the number, higher is the priority.

NOTIFY

The system sends the success or failure message *MaximumConditionCode* to the user specified in this parameter. Following is the syntax:

NOTIFY="userid | &SYSUID"

Here system sends the message to the user "userid" but if we use NOTIFY = &SYSUID, then the message is sent to the user submitting the JCL.

MSGCLASS

To specify the output destination for the system and Job messages when the job is complete. Following is the syntax:

MSGCLASS=CLASS

Valid values of CLASS can be from "A" to "Z" and "0" to "9". MSGCLASS = Y can be set as a class to send the job log to the JMR *JOBLOGManagementandRetrieval*: *arepositorywithinmainframestostorethejobstatistics*.

MSGLEVEL

Specifies the type of messages to be written to the output destination specified in the MSGCLASS. Following is the syntax:

MSGLEVEL=(*ST, MSG*)

ST = Type of statements written to output log

- When *ST* = 0, Job statements only.
- When *ST* = 1, JCL along with symbolic parameters expanded.
- When *ST* = 2, Input JCL only.

MSG = Type of messages written to output log.

- When *MSG* = 0, Allocation and Termination messages written upon abnormal job completion.
- When *MSG* = 1, Allocation and Termination messages written irrespective of the nature of job completion.

TYPRUN

Specifies a special processing for the job. Following is the syntax:

TYPRUN = SCAN | HOLD

Where SCAN and HOLD has the following description

- TYPRUN = SCAN checks the syntax errors of the JCL without

executing it.

- TYPRUN = HOLD puts the job on HOLD in the job queue.To release the job, "A" can be typed against the job in the SPOOL, which will bring the job to execution.

TIME

Specifies the time span to be used by the processor to execute the job. Following is the syntax:

TIME=*mm, ss* or TIME=ss

Where mm = minutes and ss = seconds

This parameter can be useful while testing a newly coded program. In order to ensure that the program does not run for long because of looping errors, a time parameter can be coded so that the program abends when the specified CPU time is reached.

REGION

Specifies the address space required to run a job step within the job. Following is the syntax:

REGION=nK | nM

Here, *region* can be specified as nK or nM where n is a number, K is kilobyte and M is Megabyte.

When REGION = 0K or 0M, largest address space is provided for execution.In critical applications, coding of 0K or 0M is prohibited to avoid wasting the address space.

Example

```
//URMISAMP JOB (*),"tutpoint",CLASS=6,PRTY=10,NOTIFY=&SYSUID,
//   MSGCLASS=X,MSGLEVEL=(1,1),TYPRUN=SCAN,
//   TIME=(3,0),REGION=10K
```

Here, JOB statement is getting extended beyond the 70th position in a line,so we continue in the next line which should start with "//" followed by one or more spaces.

Miscellaneous Parameters

There are few other parameters, which can be used with JOB Statement but they are not frequently used:

ADDRSPC	Type of storage used: Virtual or Real
BYTES	Size of data to be written to output log and the action to be taken when the size is exceeded.
LINES	Maximum number of lines to be printed to output log.
PAGES	Maximum number of pages to be printed to output log.
USER	User id used to submit the job
PASSWORD	Password of the user-id specified in the USER parameter.
COND and RESTART	These are used in conditional job step processing and are explained in detail while discussing conditional Processing.