UNIT 3

The EXEC Statement.

The EXEC Statement

- Objectives.
- Purpose of the EXEC statement.
- Syntax.
- Parameters.
    a. PGM
    b. PROC
    c. REGION
    d. TIME
    e. PARM
- Examples.

Objectives

- Understand the need for the EXEC statement.
- Learn the most important parameters for the EXEC statement.
- Code different EXEC statement depending on the requirement.

Purpose of the EXEC statement

- The EXEC statement is needed for the following reasons

    a. To specify which programs need to be executed.

    b. To specify which procedures (covered in later units) need to be executed.

    c. To specify the system required parameters for each Step.

The main purpose of a Job is to execute programs. Therefore, any Job will have at least one Step (EXEC statement).

When speaking of the EXEC statements, Step and Exec are very frequently used interchangeably.

All parameters required by a Program are coded along with that step. Parameters may also be coded to override either the default parameter values and/or the parameters given in the JOB statement.

Syntax

The EXEC statement has one positional parameter and other keyword parameters.

Example:

```
//JONNAME  JOB  NOTIFY=‘userid’
//STEPNAME EXEC PGM=Program-name
```

or

```
//JONNAME  JOB  NOTIFY=‘userid’
//STEPNAME EXEC PROC=Proc-name
```

or

```
//JOBNAME  JOB  NOTIFY=‘Userid’
//STEPNAME EXEC Procedure-name
```

The program or procedure parameter must be the first parameter in the EXEC statement. Hence, it is ‘Positional’ in nature.

PGM= or PROC= is a positional parameter even though it is coded in keyword format.

Parameter – PGM

Type: Positional parameter

Purpose: To name the load module program which is to be executed in the EXEC statement

Syntax:

```
 //JOBNAME   JOB  ACCTINFO,'PROGRAMMER NAME',CLASS=A,.
//STEP1      EXEC  PGM=Program-Name,..
```

Example:

```
//ABC    JOB  ACCT123,'PROGRAMMER NAME',CLASS=A,
//STEP1  EXEC PGM=SAMPLE,.
```

The PGM parameter names the load-module you wish to execute in a particular Step. The name of the load-module is a character-string ranging from 1 to 8 characters.

Parameter – PROC

Type: Positional parameter

Purpose: To name the Procedure which is to be executed in the EXEC statement

Syntax:

```
//JOBNAME   JOB  ACCTINFO,'PROGRAMMER NAME',CLASS=A,.
 //STEP1    EXEC  PROC=Proc-Name,..
```

Instead of coding PROC= procedure-name, one can code the procedure-name directly, without the 'PROC=' syntax, since this parameter is positional in nature.

Example:

```
//STEP1  EXEC PROC=USERPROC,…
```

Can also be coded as:

```
//STEP1  EXEC USERPROC,…
```

Parameter – REGION

Type: Keyword parameter

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

Syntax:

```
//JOBNAME  JOB  ACCTINFO,'PROGRAMMER NAME',CLASS=A,.
//STEPNAME EXEC PGM=Prog-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',CLASS=A,..
//STP1   EXEC PGM=MYPGM1,REGION=1024K

//ABC    JOB  ACCT123,'PROGRAMMER NAME',CLASS=A,..
//STEP1  EXEC PGM=MYPGM2,REGION=10M
```

As with the JOB statement, the REGION parameter can also be coded for the EXEC statement. If it is coded in both the statements, the value specified in the EXEC statement overrides that of the JOB statement. However, the value in the EXEC statement cannot be more than that of the JOB statement.

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',CLASS=A,..
//STEPNAME EXEC PGM=PGMNAME,TIME=(mm,ss),.
```

TIME = (minutes, seconds)

Example:

```
//JOBNAME  JOB  TIME=(1,30)
//STP1     EXEC PGM=ABC, TIME=1
```

As with the JOB statement, the TIME parameter can also be coded with the EXEC statement. If it is coded in both the statements, the value specified in the EXEC parameter overrides that of the JOB statement. But, the value specified in the EXEC statement cannot be more than that of the JOB statement.

Parameters – PARM

Type: Keyword parameter

Purpose: To pass data to the program that is being executed in the step.

Syntax:

```
//JOBNAME  JOB ACCTINFO,'PROGRAMMER NAME'
//STEPNAME EXEC PGM=PGMNAME,PARM='Parm'
```

Example:

```
//JOBNAME  JOB  ,,NOTIFY='JSMITH'
//STEP1    EXEC PGM=SAMPLE,PARM='ABCDEFGH'
```

The maximum length of the value specified to be passed to the program should be 100 bytes. But, the system attaches a 2-byte binary value at the beginning of the data-block passed. This filler contains the length of the data passed. Hence, the program should compensate for the additional 2 bytes at the beginning of the data-block into which it receives the value and refer the actual data beginning only from the 3rd byte onwards.

Parameter – PARM (Continued)

Relationship of PARM to Cobol Program

```
//JOB1    JOB  ACA123),'ANDREW', CLASS=A,MSGLASS=A
//JOBLIB  DD   DSN=MAIN006.X100.LOADLIB,DISP=SHR
//STEP1   EXEC PGM=Sample,PARM='PRINT'
//DDIN    DD   DSN=MAIN086.INFILE,DISP=SHR
//DDOUT   DD   SYSOUT=*
```

Take a look at linkage Section of a COBOL source code.

```
LINKAGE SECTION.
01   PARM-FIELD.
       05   PARM-LENGTH      PIC     S9(04)  COMP.
       05   PARM-INDICATOR   PIC     X(05).
PROCEDURE DIVISION   USING PARM-FIELD
A000-CHECK-PARM.
      IF PARM-INDICATOR  = 'PRINT'
              NEXT SENTENCE
      ELSE
             PERFORM 0100-CLOSE-FILES
```

Parm Indicator is equal to the string 'PRINT'.
Parm-length is the length of the parm field.

Examples of EXEC statements

Following are a few sample EXEC statements in a multi-step JOB

```
//JOBNAME JOB  NOTIFY='userid'
//STEP1   EXEC PGM=MYPROG,REGION=4M,
//             TIME=NOLIMIT
//STEP2   EXEC PROC=MYPROC
//STEP3   EXEC PGM=MYPROG,PARM='E001BROWN'
```

Unit 3 Exercises

Complete the following:

1. _____ or _____ is the first parameter to be coded for a EXEC statement.

True or False (Circle One)

2. **(T/F) One can omit giving the program name as well as the procedure name in the EXEC statement.**

3. **(T/F) The TIME parameter in the EXEC statement overrides the JOB statement TIME parameter.**

4. **(T/F) There are no positional parameters in the EXEC statement.**

5. **(T/F) It is mandatory to code the REGION parameter for each step.**

6. **(T/F) The PARM parameter can be coded in the JOB statement.**

7. **(T/F) Coding a step name is mandatory.**

8. **(T/F) A JOB can consist of only one EXEC statement.**

Unit 3 Lab Exercises

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

1. In your PDS called 'USERID.JCL.CNTL', create a new member called JOBTEST2.
2. Copy the JOB card from member JOBTEST1, and change the Jobname to useridB.
3. A. Write an EXEC statement to execute a utility program IEFBR14 (see example below).

   B. Code the TIME parameter of five seconds.

   C. Code region parameter for this step as 4K

   D. Copy the rest of the JCL below to create the rest of the job.

   E. Save the member and submit the job.
4. Check SDSF for all account messages and JES messages.
5. Check the actual execution time in the JES messages.

```
//useridB  JOB account number,'your name',
//              MSGLEVEL=(1,1),
//              CLASS=class,
//              MSGCLASS=msgclass,
//              NOTIFY=userid
//*
//* Substitute valid values for TIME and REGION.
//*
//STEP1    EXEC  PGM=IEFBR14,
//              TIME(mm,ss),
//              REGION=nnnK
//*
//* Substitute 'userid' with your own security userid.
//*
//DD1      DD    DSN=userid.PQR.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
//
```