Legac-E Education

Batch Job Design – Single versus Multiple Steps

When constructing the z/OS Job Control Language (JCL) for a Batch Application one of the early design choices is whether to have a single step in a Job, or have multiple steps per Job. z/OS JCL provides two mechanisms for controlling the execution of steps within a Job once it has been scheduled for execution and these are the COND parameter and the newer Conditional JCL; IF/ELSE/ENDIF construct.

The older COND parameter was designed for selective EXCLUSION of Job Steps and is still heavily used in IBM supplied JCL procedures. This concept of exclusion is what makes the logic of the parameter and its operands difficult to comprehend. The newer Conditional JCL is designed to permit INCLUSION and hence is easier to understand as it is similar to other IF/ELSE/ENDIF constructs both in programming languages and the Access Method Services Utility (IDCAMS).

Most installations will have a Job Scheduling package to control workflow within the Production Environment. Besides job submission, such packages check the System Management Facility (SMF) data when a Job terminates. Any ABEND codes or Condition Codes (COND CODE) issued by the Job's steps can be used to determine which Job in a sequence is to be submitted next. It is this capability of the Scheduling Package which begs the question "Should Jobs be single or multi-step?"

There is nothing wrong with having multi-step jobs, indeed IBM supplied language compilation procedures typically work on this premise, and rely on the COND parameter to control the flow. The issue with multi-step jobs occurs if the Job fails physically (ABEND) or logically (COND CODE greater than zero). (The term logical failure is used in respect of COND CODE, as from the Operating Systems perspective the fact that the COND CODE was issued means the Job terminated normally.)

The RESTART parameter can be used on a JOB statement to skip early steps within a multi-step job, but there are a couple of scenarios which need to be taken into account. Temporary data sets are not kept beyond Job end so these will not be available on restart unless the restart begins at the point of their creation rather than their use. Restarting a Job will impact the logic of any conditional processing irrespective of which method is used so this adds a complication. It can be seen that while single step jobs restrict the use of temporary data sets they do overcome the complications of conditional JCL, especially if reliance is made on the capabilities of the Scheduling Package.

Batch Job Design – Single versus Multiple Steps

Does single step Job mean only one EXEC statement within the Job? Possibly not as there may be a case for constructing re-runnable Jobs and if so the Job would most likely contain two EXEC statements, the second for the Application Program and the first for a tidy up utility. The function of the tidy up utility is to ensure that any data sets to be created by the Application Program are deleted before it runs. This action should avoid JCL errors due to “Duplicate Name” conditions. Whilst it is possible to code a third parameter in the DISP field to delete data sets if the program ABENDS, this has two disadvantages; 1) it inhibits investigation within the deleted data set should that be required, 2) it only applies to ABENDS not COND CODE greater than zero. There are two methods for deleting data sets as the first step of a Job, the first being to use IEFBR14 and the second is to use IDCAMS.

Using IEFBR14

This is a dummy program in that the only processing it should do is set the COND CODE to ZERO. It was originally designed to aid the testing of JCL streams. It does not do any file processing and indeed has no files of its own so there is no need for SYSPRINT or SYSOUT DD Statements. The deletion is actually done by the z/OS Allocation/Termination routines when they honour the DISP parameter. This means that all that is required, in addition to an EXEC statement for IEFBR14, is a DD Statement with any valid DDNAME of the user’s choice, a DSN parameter naming the data set to be deleted, a SPACE parameter such as SPACE=(TRK,0), and a DISP parameter such as DISP=(MOD,DELETE). The beauty of the DISP parameter is that if the data set exists the system will locate the data set at allocation time, and delete it on termination; however if the data set does not exist the system will create it as new at allocation and delete it at termination. In both cases there is no danger of a JCL failure because of “Data Set No Found”. The peculiar SPACE parameter is there to satisfy any translation of MOD to NEW if the data set does not exist. (There is no requirement to occupy disk space, merely meet the needs of the disks Volume Table of Contents (VTOC).

Use a separate DD statement for each data set to be deleted.

Example:

```
//         SET  STU=&SYSUID        /* DATA SET HLQ      */
//S0010    EXEC PGM=IEFBR14
//MAGIC    DD   DSN=&STU..SEQ.F300,DISP=(MOD,DELETE),
//              SPACE=(TRK,0)
```

Batch Job Design – Single versus Multiple Steps

Using IDCAMS

This method uses Access Method Services (AMS) and as such there will be one or more DELETE control statements in the SYSIN data set of IDCAMS. This means that besides the EXEC statement for IDCAMS, there will be a SYSPRINT DD statement for messages, and a SYSIN DD statement to provide the control statements. The attraction of IDCAMS is that there is no possibility of the control statement causing a JCL error if the data set to be deleted does not exist. That however is countered by the fact that a COND CODE greater than zero will be issued by each control statement in error. This can be addressed by ensuring that after the last DELETE statements there is a statement which reads like " SET MAXCC=0" which effectively nullifies the control statement errors.

Prior to z/OS Version 2 the biggest disadvantage of this method was that symbolic parameters were not allowed in the input stream, but that restriction has been removed. Additionally it is now possible to have in-stream data within JCL procedures so there is no need for separate control statement members. These two recent features help simplify the construction of re-runnable Jobs without having multiple control statement members to meet different circumstances.

Example

```
//         EXPORT SYMLIST=STU
//         SET  STU=&SYSUID
//S0010    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *,SYMBOLS=JCLONLY
  DEL &STU..KSDS.FILE
  SET MAXCC=0
```

Is there a recommendation?

It was not the purpose of this document to make a recommendation on any of the topics discussed. The intent was to provide information to allow an informed choice. That said it might make some sense to standardize on single step Jobs for Production where the Scheduler is in control, and use multi-step jobs in Development where Conditional JCL can be used to aid single thread dependencies.