SHARED SPOOL MODS

Installation Documentation

For Jes2 1.7 & 1.8

Installation Instructions

Table of Contents

General Information and Overview

What are the SHARED SPOOL MODS, and what can they do for you?

The SHARED SPOOL MODS are the latest version of what was previously known as the MELLON SHARED SPOOL MODS. We all owe a great deal to Mellon Bank for originating these mods, but in light of the fact that they have been maintained outside of Mellon Bank for a dozen or more years, and that I have completely rewritten them twice in that time, I will be referring to them as the SHARED SPOOL MODS from this time forward.

The SHARED SPOOL MODS have traditionally provided additional job selection criterion that could be used by one job to indicated that it should be selected for execution on a system containing a specific resource, or that it should be selected or not selected based on the execution status or resource requirements of other related jobs. Please note that it is NOT necessary to be running a Multi-Access Spool Configuration, or MAS to take advantage of this package. Beginning with this release of the SHARED SPOOL MODS the control of jobs has been extended to include operational controls that can be used to place limits on the total number of jobs that may run concurrently in each job class on each system within a MAS, regardless of the availability of JES2 or WLM initiators. We use this feature to provide operational controls that allow us to use WLM managed initiators for jobclasses that must be otherwise regulated, such as job classes that require a finite number of tapes. Other system level controls can be used to limit concurrently executing jobs from a singe user, or group of users based on either jobname or userid.

The SHARED SPOOL MODS enhance the job selection routines that JES2 uses when selecting the next job for execution from the input queue by adding new requirements and qualifications to submitted jobs. Throughout this document, whenever we refer to job selection, we mean the process of selecting a job from the input queue for assignment to an initiator and its immediate execution.

In addition to job selection enhancements, this version of the SHARED SPOOL MODS provides extended control features for WLM managed initiators. The additional control allows your operations staff to stop, start, or limit the maximum number of jobs selection for individual job classes, or for all job classes. These same limits are extended to traditional JES2 initiators as well as to WLM managed initiators. You are allowed to limit the number of jobs in concurrent execution based on a userid mask or jobname mask, so that you can prevent a single user, or group of users from monopolizing the available initiators in the system.

The new requirements that can be used to qualify when a job is eligible to be selected to run, or on which system it can run if you are in a MAS (Multi-Access Spool configuration), are expanded by the SHARED SPOOL MODS beyond existing native JES2 mechanisms. The new selection capabilities are listed below.

The SHARED SPOOL MODS have traditionally provided additional job selection criterion that could be used by one job to indicated that it should be selected for execution on a system containing a specific resource, or that it should be selected or not selected based on the execution status or resource requirements of other related jobs. Please note that it is NOT necessary to be running a Multi-Access Spool Configuration, or MAS to take advantage of this package. Beginning with this release of the SHARED SPOOL MODS the control of jobs has been extended to include operational controls that can be used to place limits on the total number of jobs that may run concurrently in each job class on each system within a MAS, regardless of the availability of JES2 or WLM initiators. We use this feature to provide operational controls that allow us to use WLM managed initiators for jobclasses that must be otherwise regulated, such as job classes that require a finite number of tapes. Other system level controls can be used to limit concurrently executing jobs from a singe user, or group of users based on either jobname or userid.

Additionally job selection can be altered in the following ways -

- All jobs can be delayed from the time they are first read until they are eligible for execution by some number of seconds.
- Delay a job by some specified amount of time, up to 100 hours.
- Delay a job until some specific time of day occurs.

System specifications can be set to restrict jobs, even jobs using WLM managed initiators in the following ways -

- A limit may be set for the number of jobs that can run in each job class.
- A limit may be set that limits the number of active jobs based on characters in the jobname.
- A limit may be set that limits the number of active jobs based on characters in the userid associated with each job.

A special option is available to prevent the use of the '$SJ(jobnumber)' to force a job into execution if it is needed. If the $SJ command is allowed, it will also override all of the above options that could prevent a job from moving into execution.

Here are some examples of how these enhancements to job selection can be used.

Assuming you have some number of jobs that can only run where a particular resource exists – a CICS region, a vendor program, or maybe where extra tape drives are attached, and assuming those resources may be available on different systems in the MAS configuration each time your jobs are submitted, you can still ensure your job is only initiated on the correct systems, by using the "/*ROUTE XEQ *scheduling environment name*" JECL statement if those resources are described by a WLM scheduling environment, and the WLM scheduling environment is activated only where the resource exists. The resource described by a WLM scheduling environment does not have to represent real

resources. You could for example have a resource defined to WLM named "BATCHWINDOW" that operations or automation enables, or disables whenever they want, and jobs that are submitted with a "/*ROUTE XEQ BATCHWINDOW" statement would only be eligible for job selection when, and only on the systems where, the BATCHWINOW resource is enabled.

The "/*ROUTE XEQ HERE' statement allows jobs to execute only on the LPAR that they are submitted from, working like a dynamic SYSAFF= parameter. This option is maintained for compatibility reasons only, since the same thing can now be accomplished with standard JCL.

The "/*CNTL *resource_name*,EXC | SHR" statement can be used to create an arbitrary resource name and have any jobs you wish coordinate their availability for execution based on the need for exclusive or shared access of that arbitrary resource name. No previous resource name setup is required to use this feature; only agreement among the participating jobs on the resource name you wish to co-ordinate your job execution on.

One use of this feature is to serialize or single thread a group of jobs, where only one may run at a time. All of the jobs to be serialized would use a "/*CNTL" statement that uses the EXC or exclusive option, and the same resource name. Any resource name will do, it just has to be the same in all participating jobs. As soon as one of the jobs in the group is selected for execution all other jobs with the same "/*CNTL" statement will become ineligible for selection until the first job ends, and so on. Many times a DD statement with a dummy dataset name and a disposition of old is used to accomplish nearly the same thing, but that solution potentially ties up initiators that could be used to process other ready work and actually creates a throughput bottleneck. The use of the "/*CNTL" statements prevents the competing jobs from even being initiated, and so does not tie up, or waste initiators while each job waits for it's turn to run. It also will not produce the annoying messages that jobs are waiting for exclusive use of the dummy dataset name. A more complex example using this feature is particularly valuable to systems programmers using SMP/e, where multiple jobs may run concurrently using the SHR option with a common name as long as each job is not actually performing updates, but a job that performs updates must run alone is coded to use the same resource name but with an EXC or EXCLUSIVE option specified, which ensures it will always run alone. In my shop we use our SMP/e CSI name as a resource name, and SMP/e LIST or CHECK jobs are coded with a SHR option, while APPLY, or RESTORE jobs are coded with an EXC for the same resource name. At the request of 'power' users of this feature, a new option, PRG or PURGE, is now available that will prevent any other jobs regardless of whether or not they have a SHR or EXC specification from running until after the job with the PRG is selected and run. The PURGE option effectively says, "This job must run alone, and no other job with this resource name may start until after the job with the PRG specification has completed, and it must be the NEXT job to run holding this resource name.

The "/*AFTER", "/*BEFORE", and "/*WITH" and "/*WITHOUT" control statements can be used to dictate the order of job selection for a group of jobs in relation to each

other. There are some specific options, other than the most obvious ones, that can be specified for each system that determine rules that must be met before any of the jobs controlled by these statements is satisfied. The processing options that can be specified for these types of statements are covered later in this document.

A job class limit can be specified for each MAS member that limits the number of concurrently executing jobs in any execution class. Different limits can be set for each class, and each can be dynamically changed via operator command. The limits are effective for both traditional JES2 managed job classes as well as WLM managed job classes. This option might be used when putting a single lpar into maintenance mode, it is possible with just a couple of commands to stop all further job selection, except for the jobclass or jobclasses that maintenance jobs will run in, and then to restore the original environment that probably allowed any jobclass to execute. This option can be used when WLM management is in effect for a jobclass, to allow jobs in a single jobclass to run on only a single MAS member. This option can also be used to prevent WLM from starting too many jobs of a particular class, before it has a chance to determine the overall effect of the jobs on the system.

A mask value can be set that is used to select characters from the userid or jobname that is associated with each job, and the number of concurrently executing jobs may be limited for all jobs with a matching resultant value.

Masks are composed of up to 8 characters, with each character being either an asterisk, or a character "U". The characters that form the userid are compared against the mask, and for each position in the mask with a 'U' the corresponding character from the userid is extracted. All of the extracted characters from the userid are then concatenated, to form an intermediate result. All jobs with the same intermediate result are considered to belong to the same group, and it is that group that the limit, if any, is imposed against.

As an example of limiting active jobs based on userids and a mask value, assume your installation has a standard for creating userids that says the userids are formatted with the following scheme;

Positions 1-2 represent location, LL
Positions 3-4 Are an individuals initials, II
Positions 5-6 Are the department, DD
Positions 7-8 Are job function, FF

A userid could then be said to be in the form of C'LLIIDDFF'.

Next if you wished to limit the number of jobs that were owned by users in each department, a mask of ****UU** could be set. With that mask, only positions 5 and 6 of the userid would be taken into account when determining if the maximum number of jobs that are allowed for that group. Likewise, a mask of **UU**** could be set and then the limit would be imposed for jobs that were owned by userids with the same first and last initials, or a mask of UU**UU** could be set to limit the number of active jobs for each

location and department combination. Obviously then a mask of UUUUUUUU would limit the number of jobs for each userid.

The masking feature is used exactly the same way with the JOBNAME as it is with the USERID.

Interaction with WLM Resources and Scheduling Environments

One of the primary job selection criterions available through the SHARED SPOOL MODS is the availability of a particular resource name being available on a particular system. In the past, we maintained these names in a table within the SHARED SPOOL MODS code; and provided JES2 commands that would alter the state of the resources, either on or off. We now use WLM Scheduling Environment names instead. The scheduling environment names are the same ones displayed on the SDSF SE panel. The scheduling environments are on when all of the resource names that make up each environment are also on. WLM resource names are the same names displayed on the SDSF RES panel. The WLM scheduling environment names are what are matched on the "/*ROUTE XEQ scheduling environment name" JECL statements. This function is virtually identical to the new SCHENV parameter on the jobcard. In fact, we substitute the internal value of the SCHENV with what we find in the /*ROUTE XEQ card only if there is no current value set for SCHENV. These mods supported the function long before JES2 adopted the facility, which appears to have been modeled after the SHARED SPOOL MODS, and arose out of a long standing SHARE requirement.

While these mods continue to support the older style /*ROUTE XEQ statements to route jobs for execution based on resource locations, **we strongly suggest that the older /*ROUTE XEQ statements for job selection be replaced by the new IBM supported SCHENV= parameter on the jobcard.**

Compatibility and Support of the SHARED SPOOL MODS

The SHARED SPOOL MODS, as they exist today, do not modify any JES2 source directly, other than the $USERCBS macro that is intended to be modified by the user, and only makes use of fully supported and documented exits and facilities; we see no reason why they may become unsupportable in the future.

History of the SHARED SPOOL MODS, at least as I know it.

These mods were originally designed and maintained by MELLON Bank, and we owe a great deal of gratitude to MELLON for its original design and forethought. I have maintained the SHARED SPOOL MODS personally for over ten years, and they were maintained by others at this company for several years before that time. Around the turn of the millennium, I completely rewrote the mods. The intent of the rewrite was to incorporate the new JES2 functions that make use of WLM resource names and scheduling environments, and to repackage the mods such that they were all contained in standard JES2 table pairs and exits, so that we would make no direct changes to the JES2 source code. That objective was met. The conversion effort for 1.7 was such that we virtually rewrote everything one more time. While in their current state, they are not simple exits, they are all standard documented exits, all using standard, documented interfaces to JES2 and WLM, and are quite manageable. I will refer to these mods simply as the 'SHARED SPOOL MODS' from now on.

JOB Level Method of Invoking the SHARED SPOOL MODS features

The SHARED SPOOL MODS features may be invoked by individual jobs via 8 JECL statements, which are:

"/*CNTL XEQ resource, { EXC | SHR | PRG}"
"/*ROUTE XEQ *scheduling environment name* | HERE "
"/*WITH jobname"
"/*WITHOUT jobname"
"/*AFTER jobname"
"/*BEFORE jobname"
"/*HOLDFOR hh:mm:ss"
"/*HOLDTIL hh:mm:ss"

For a detailed explanation of what each JECL statement does, and its format, please refer to the "SHARED SPOOL MODS for JES2 1.7 Users Guide and Documentation".

In terms of operational support for the SHARED SPOOL MODS, the mods expand the results of the $DJ command to include SHARED SPOOL MODS information for each job that has one or more of the SSM's JECL statements. We also created a new informational message $HASP890 with detailed information about each of the JECL statements that are recognized in a job's JCL. These are both documented below.

SYSTEM Level SHARED SPOOL MODS features

The features that are not associated with specific job selection, i.e. the features not listed in the item directly above this one, are all SYSTEM level options, and can be set either through JES2 PARMS or through the use of new JES2 commands. Examples of the features that can be specified at a system level include;

- Specifying whether or not to write SMF records.
- Specifying the level of SMF recording.
- Specifying the SMF number if SMF recording is active.
- Specifying a maximum number of jobs that can run in each job class.
- Specifying a maximum number of jobs that can be run based on a masked userid.
- Specifying a maximum number of jobs that can be run based on a masked jobname.

It is important to note three things about the system level options, first they are effective for the individual JES2 system, and they are NOT MAS wide in effect. Second, they can be altered through simple JES2 commands as well as fixed JES2 parms. And finally, all of the system level options are optional, none of them are required.

The JES2 $DJ command

The JES2 $DJ command output has been extended to include information about /*CNTL statements. Up to five CNTL names are displayed qualified with an “E” for exclusive, or an “S” for shared. One /*WITH jobname, one /*WITHOUT, one /*BEFORE and one /*AFTER jobname, will each be displayed if those types of statements are present in the job. Examples of the extended displays are given below. Please note that the information is included in either the standard or long versions of the command.

Altered Display Commands –

```
-$DJ(25926)

$HASP890 JOB(T0SM139)
$HASP890 JOB(T0SM139)   STATUS=(AWAITING EXECUTION),CLASS=X,
$HASP890                PRIORITY=6,SYSAFF=(ANY),HOLD=(NONE),
$HASP890                DELAY RSN=HOLDTIL TIMR,AFTER=T0SM150,
$HASP890                BEFORE=T0SM160,WITH=T0SM140,WITHOUT=T0SM138,
$HASP890                HOLDFOR=00:02:00¦ELAPSED,HOLDTIL=10:20:00,
$HASP890                CNTL=(RESNAME1-E,MYSTUFF-S,YOURSTUF-P,
$HASP890                COMMON-S,RESNAME1-E)
```

The **BOLD** text in the display above is all as a result of SHARED SPOOL MODS statements in the JCL. First the ‘DELAY RSN=’ is only displayed for jobs with SHARED SPOOL MODS statements in the JCL, and indicates whether the job has been bypassed for job selection due to a SHARED SPOOL MODS restriction or if it has simply never been selected by JES2 as a candidate for execution. In a future release I intend to make the ‘DELAY RSN’ field available for every job in the system. In the case above the job is held due to the HOLDTIL timer value of 10:20:00. The AFTER=, BEFORE=, WITH=, and WITHOUT= all indicate the jobname associated with each like named control statement. The HOLDFOR= and HOLDTIL= fields indicate the time values specified, and whether or not they have elapsed. In this case the HOLDFOR time has expired, the HOLDTIL time has not. The CNTL= field lists the values specified in up to 5 /*CNTL statements followed by either a -S for shared, -E for exclusive, or -P for purge.

The LONG version of the Display Job command is shown below, it also contains the same SHARED SPOOL MODS information that the short form of the display does.

```
-$DJ(25926),LONG

$HASP890 JOB(T0SM139)
$HASP890 JOB(T0SM139)   STATUS=(AWAITING EXECUTION),CLASS=X,
$HASP890                PRIORITY=6,SYSAFF=(ANY),HOLD=(NONE),
$HASP890                CMDAUTH=(LOCAL),OFFS=(),SECLABEL=,
```

```
$HASP890                USERID=T0SM0,SPOOL=(VOLUMES=(JES2T3),TGS=1,
$HASP890                PERCENT=0.0009),ARM_ELEMENT=NO,CARDS=16,
$HASP890                REBUILD=NO,SRVCLASS=BATTSTMD,SCHENV=TAPE,
$HASP890                SCHENV_AFF=(TSPC,TSPD),CC=(),DELAY=(),
$HASP890                CRTIME=(2007.116,13:42:07),
$HASP890                DELAY RSN=HOLDTIL TIMR,AFTER=T0SM150,
$HASP890                BEFORE=T0SM160,WITH=T0SM140,WITHOUT=T0SM138,
$HASP890                HOLDFOR=00:02:00¦ELAPSED,HOLDTIL=10:20:00,
$HASP890                CNTL=(RESNAME1-E,MYSTUFF-S,YOURSTUF-P,
$HASP890                COMMON-S,RESNAME1-E)
```

$HASP943 messages

In addition informational messages, $HASP493 and $HASP494 are written to the log as jobs with /*CNTL, /*WITH, /*BEFORE, or /*AFTER are read. Examples of the messages follow.

These messages were issued for the job displayed above, as it was submitted. These form one of the audit trails available for used SHARED SPOOL MODS options. The other audit trail option is of course the optional SMF recording.

```
$HASP943 T0SM139  * -- HOLD UNTIL = 10:20:00       --
$HASP943 T0SM139  * --   HOLD FOR = 00:02:00       --
$HASP943 T0SM139  * -- WITH     JOBNAME = T0SM140   --
$HASP943 T0SM139  * -- WITHOUT  JOBNAME = T0SM138   --
$HASP943 T0SM139  * -- CONTROL INFO = RESNAME1,EXC  --
$HASP943 T0SM139  * -- CONTROL INFO = MYSTUFF ,SHR  --
$HASP943 T0SM139  * -- CONTROL INFO = YOURSTUF,PRG  --
$HASP943 T0SM139  * -- CONTROL INFO = COMMON  ,SHR  --
$HASP943 T0SM139  * -- CONTROL INFO = RESNAME1,EXC  --
$HASP943 T0SM139  * -- AFTER    JOBNAME = T0SM150   --
$HASP943 T0SM139  * -- BEFORE   JOBNAME = T0SM160   --
```

New JES2 Commands for the Shared Spool Mods

In support of the new features we now have fir the first time, JES2 parm statements that can be included in your JES2 startup parms. The values set, or defaulted to, in the JES2 parms can be displayed using JES2 commands, and can be modified by using JES2 commands.

The general form of the new display commands are;

$D SSM,option,option,option,…

The $D SSM command, with no options will display all SHARED SPOOL MODS system level options at one time - about 15 lines of console output.

The general form of the new modify commands are;

$T SSM,option,option,option…

The specific options that can be displayed or changed are -
SMFNUM - the SMF number to use when writing SSM SMF records.
SMFOPT - The SMF recording level option
BEAFTER - the specialized options to take when using /*BEFORE and /*AFTER.
BATIME - # of seconds to delay all jobs before job selection, it BEAFTER=DELAY
UIDMAX - the maximum number of concurrently active jobs with matching UIDMASK values.
UIDMASK - the MASK to apply to the submitters, userid before checking the UIDMAX value.
JBNMAX - the maximum number of concurrently active jobs with matching JBNMASK values.
ALLOWS - determines if $SJ commands are allowed or not.
CLASSOPT - determines if CLASSLIM values are effective.
CLASSLIM(jobclass) - sets the limit of concurrently active jobs for each jobclass.

These commands are explained in much greater detail in the SHARED SPOOL MODS OPERATIONS COMMANDS document.

Additional Information

Detailed information about all of the features is available in the ‘SHARED SPOOL MODS User Documentation’.

Installation Procedures

Step 0 - Always take a backup before you start

This is an SMP/e install, so backup accordingly. You will be replacing JES2 macro $USERCBS - it was designed to be updated by the user, all other elements are NEW, we are not going to update, modify, or replace any existing JES2 macros, source, or modules, but you ALWAYS want to be able to get back to where you started if you need to!

Step 1 - Apply the Shared Spool Usermods

The SHARED SPOOL MODS have been repackaged so that they are a single usermod which I have named LSES500. It contains several SRC, JCLIN, and MAC member additions to JES2. The additional members are added directly to the IBM provided JES2 source libraries SHASMAC and SHASSRC, and are packaged, to assemble into the IBM provided link library, SHASLNKE. They do NOT replace any existing macros, or source elements with the exception of the $USERCBS macro which is designed to be updated, these are all new elements and they are detailed at the end of this section. To install the package you simply run a RECEIVE / APPLY CHECK / APPLY sequence of SMPe jobs. Then make the updated SHASLNKE library available to the system as you would normally do after any other JES2 maintenance, update your JES2 initialization parms, and warm start JES2.

The mods will add new source members to DDDEF SHASMAC and SHASSRC and will place new linkedited modules into DDDEF SHASLNKE. The JCLIN also references standard SYSLIB DD datasets;

SYS1.MACLIB
SYS1.MODGEN
SYS1.SHASMAC
SYS1.SHASSRC
SYS1.AHASMAC
SYS1.AHASSRC
SYS1.AMACLIB
SYS1.AMODGEN

The names should be fine unless the distribution of JES2 changes significantly. If it does, it may be necessary to modify the JCLIN before receiving the usermods.

Just to restate this one more time, the macros, source, and modules are all new - except for macros $USERCBS which is intended by IBM to be updated. The source, macros, and load modules will all be placed into standard JES2 libraries. With the exception of

$USERCBS, NO IBM SOURCE, MACROS, OR LOAD MODULES are either modified or replaced. These are all new elements, that will be implemented via standard exits and interfaces.

NOTE * * * This will result in a large number of large assemblies. I had to change the SMP/e Utility entry for ASMA90 (or whatever your ASM utility entry points to) to have a parm value that includes “SIZE(MAX)”, and then also use a REGION of 18M on the JOBCARD and EXEC statement in the APPLY step. Otherwise I got some really odd error messages. You should get a return code of zero for all the assemblies and links.

You may have been better than I, and already had size(max) setup of course.

Step 2 - Update the JES2 parms

After SMP installation, update JES2 parms by adding the following three groups of JES2 parms statements: LOADMOD statements, EXIT statements, and SSM statements.

The LOADMOD statements:

```
LOADMOD(STJTABS)  /*     DYNAMIC TABLE DEFINITIONS FOR JQE EXT. #JES7*/
                      /*                                            */
LOADMOD(STSSMTBS) /*     DYNAMIC TABLE DEFINITIONS FOR SSMT     #JES7*/
                  /*                                                */
LOADMOD(STSCX04A)  /*   CALLS EXIT 54 ROUTINES                      */
LOADMOD(STSCX54A) STORAGE=CSA  /*ROUTE XEQ RESNAME AND VALIDATES     */
                  /*                                                */
LOADMOD(STSCX54B) STORAGE=CSA /*  OTHER SPECIAL JECL STATEMENTS      */
LOADMOD(STSCX05B)  /*   PREVENTS PURGING JOBS BY RANGE              */
                  /*                                                */
LOADMOD(STSCX06A)   /*  TURNS /*ROUTE CNTL,XX INTO SCHENV= VALUES    */
                  /*                                                */
LOADMOD(STSCX19A)  /* INITIALIZATION STMTS (FOR SSM)                */
                  /*                                                */
LOADMOD(STSCX20A)  /*            JCT TO JQE COPY ROUTINE            */
                  /*                                                */
LOADMOD(STSCX24A)  /* POST-INITIALIZATION (FOR SSM)                 */
                  /*
*/LOADMOD(STSCX49A)   /* ACCEPTS OR REJECTS JES2 NEXT CHOICE OF JOBS */
                  /*                  ( THE QGOT ROUTINE )          */
LOADMOD(STSCX50A) STORAGE=CSA /* MOVE JCTX TO THE JQE EXTENSION      */
                  /*                                                */
```

NOTE ** the LOADMOD statements for STJTABS and STSSMTBS, and the EXIT statements for EXIT(19) and EXIT(24) should be physically placed BEFORE the SSM parmlib statements.

The EXIT statements:

```
EXIT(004) ROUTINE=(EXIT04A),STATUS=ENABLED
```

```
              /* A = CALLS EXIT54 ROUTINES                              */
EXIT(054) ROUTINE=(EXIT54A,EXIT54B),STATUS=ENABLED
              /* A = GETS ROUTE XEQ INFO                                */
              /* B = GETS "BEFORE/AFTER/INFORM/HOLDFOR/TIL ETC. "       */
EXIT(006) ROUTINE=(EXIT06A),STATUS=ENABLED
                      /* A = SETS SCHENV BASED ON ROUTE XEQ CARDS       */
EXIT(019) ROUTINE=(EXIT19A),STATUS=ENABLED
EXIT(024) ROUTINE=(EXIT24A),STATUS=ENABLED
                      /* 19A = BUILDS TEMP SSMT NAME/TOKEN PAIR + CB    */
                      /* 24A = BUILDS PERM SSMT N/T PAIR                */
EXIT(020) ROUTINE=(EXIT20A),STATUS=ENABLED
EXIT(050) ROUTINE=(EXIT50A),STATUS=ENABLED
                      /* COPIES JCT INFO INTO THE JQE EXTENSION         */
EXIT(049) ROUTINE=EXIT49A,STATUS=ENABLED
                      /* IMPLEMENT BEFORE AFTER WITH CNTL STATEMENTS    */
EXIT(054) ROUTINE=(EXIT54A,EXIT54B),STATUS=ENABLED
EXIT(100) ROUTINE=EXIT100A,STATUS=ENABLED
                      /* JES2X100=STSC  FCB SETUP ROUTINES              */
```

NOTE ** the LOADMOD statements for STJTABS and STSSMTBS, and the EXIT statements for EXIT(19) and EXIT(24) MUST physically be placed BEFORE the SSM parmlib statements.

The SSM statements: (this is an example only - set the parms the way you want them).

```
SSM SMFOPT=NONE,         /* SMF recording level */
    SMFNUM=216,          /* SMF number used to write smf records */
    BEAFTER=PREMOD,      /* BEFORE/AFTER processing options  */
    BATIME=3,            /* Delay time, if BEAFTER=DELAY is selected*/
    UIDMAX=256,          /* 256 jobs per 5 position uid       */
    JBNMAX=0,            /* 0 = no max based on JBNMASK value */
    UIDMASK=UUUUU***,    /* Mask used with UIDMAX to limit jobs by UID*/
    JBNMASK=********,    /* Mask used with JBNMAX to limit jobs by JBN*/
    ALLOWS=OFF,          /* ALLOW or DISALLOW $SJ commands to be used*/
    CLASSOPT=ON,         /*enforce or don't enforce limits by jobclass*/
    CLASSLIM(A-Z,0-9)=234  /*limit for each class - if classopt=on*/
```

NOTE ** the LOADMOD statements for STJTABS and STSSMTBS, and the EXIT statements for EXIT(19) and EXIT(24) MUST physically be placed BEFORE the SSM parmlib statements.

The format of the SSM statement

SSM SMFNUM=###,SMFOPT={ACTION | INPUT | ALL | NONE},
BEAFTER={PREMOD | DELAY }, BATIME=tt,UIDMAX=###,

UIDMASK=mmmmmmmm, JBNMASK=mmmmmmmm, ALLOWS={ ON | OFF }, CLASSOPT={ ON | OFF }, CLASSLIM(class specification)=###

SSM Statement Options -

SMFNUM=0

SMFNUM specifies the number of the smf record that the SHARED SPOOL MODS with write it's SMF data to, if SMFOPT is not set to NONE. Use a number between 200 and 255 that is not being used by any other products in your installation. At SunTrust we have smf number 216 reserved for this purpose. The default of zero specifies that no smf records will be written.

SMFOPT=

SMFOPT= specifies the level of SMF recording, specify either ALL for all SMF record types, INPUT for a record of all SHARED SPOOL MODS input statements, ACTION for actions taken by the SHARED SPOOL MODS, and NONE if you do not want any SMF records written.

BEAFTER={ PREMOD | DELAY }

BEAFTER specifies how the BEFORE and AFTER statements are to be processed, PREMOD specifies that they should be handled as they have historically been handled. DELAY specifies that all jobs should wait on the input queue for a length of time specified in the BATIME operand. Delay can be used to correct some unintended job sequencing that can occur when multiple jobs are submitted simultaneously and they appear to get to the input queue "out of order". - PLEASE see the end of the document for "A few final notes for /*BEFORE and /*AFTER" for a more in depth discussion of the issues around the BEAFTER and BATIME options.

BATIME=###

BATIME is used to determine how many seconds a job must wait on the input queue before becoming eligible for execution if the BEAFTER= option is set to DELAY.

UIDMASK=8 characters each is either an '*' or a 'U'

This specifies the Userid Mask. It is used in conjunction with the UIDMAX value. The USERID owning each active job (or about to be selected for execution job) is examined one character at a time and compares it to the UIDMASK, if the corresponding position in the UIDMASK is a 'U" the character from the UserID is extracted, if the character is an '*' the position is ignored. Once the end of the UserID field is reached, all the selected characters are concatenated to form an intermediate UIDMASK value. The UIDMAX value is used as a maximum count for all jobs that have a matching UIDMASK VALUE.

Ex. UIDMASK=UU*UU***
UIDMAX=2

Given the following USERIDS associated with the following jobs that are active:

JOBNAME1 has a USERID of ABCD1234 - masked value = ABD1
JOBNAME2 has a USERID of ABBD1999 - masked value = ABD1
JOBNAME3 has a USERID of CBAD2000 - masked value = CBD2
JOBNAME4 has a USERID of CBXD2050 - masked value = CBD2
JOBNAME5 has a USERID of CBXD3050 - masked value = CBD3

A new job with a userid value of ABDD1000 - masked value = ABD1, would not be allowed to start since it would become the 3rd (1 more than the limit) job with the same masked value.

A new job with a userid value of CBBD3978 - masked value = CBD3, would be allowed to start since it would only bring the total for that masked value to 2 active jobs (JOBNAME5 + the new job with a userid of CBBD3978).

A new job with a userid value of CBXD4050 - masked value = CBD4, would be allowed to start since it would only bring the total for that masked value to 1 active job with that masked value.

Note - changing the UIDMASK and UIDMAX value to lower values will not affect jobs that have already been selected for execution. They can only affect the decision to allow or reject future jobs as they move from the input to execution queues.

JBNMAX=####

This is the maximum number of jobs to allow to concurrently execute with the same jobname masked value on this JES2 member. The default value is zero and indicates that this test should not be done when JES2 selects a potential job for execution.

JBNMASK=a mask of 8 characters each either an '*' or a 'U'

This specifies the Jobname Mask. It is used in conjunction with the JBNMAX value. The JOBNAME of each active job (or about to be selected for execution job) is examined one character at a time and compared to the JBNMASK. If the corresponding position in the JBNMASK is a 'U" the character from the JOBNAME is extracted; if the character is an '*' the position is ignored. Once the end of the JOBNAME field is reached, all the selected characters are concatenated to form an intermediate JBNMASK value. The JBNMAX value is used as a maximum count for all jobs that have a matching JBNMASK VALUE.

Example - JBNMASK=UU***U**
Given the following active jobnames, and a JBNMASK=U***U** value, and a JBNMAX=2 setting;

JOBNAME1 masked value = JOE
JOB0029 masked value = JO2
JOBX masked value = JO
TSNAME1 masked value = TSE
TSBNAME masked value = TSM
TSXXXM2 masked value = TSM
JOB002X77 masked value = JO2

A newly selected job with a jobname of JOB992 would have a JBNMASK value of JO2, and would not be allowed to execute yet because it would exceed the limit of 2-(JOB002X77 and JOB0029) are already executing.

A newly selected job with a jobname of JOB993 would have a JBNMASK value of JO3 and with no matching jobname masks would be allowed to execute (the count for JBNMASK JO# would then become 1).

A newly selected job with jobname TSODEEP would have a JBNMASK value of TSE, and since there is only one other job with a matching mask value (TSNAME1), it would be allowed to execute. Then the limit would be met for that JBNMASK value.

ALLOWS={ ON | OFF }

ALLOWS determines whether or not the $SJ command is allowed when using WLM managed initiators. The default OFF prevents the use of $SJ commands from being used. Using a $SJ command will override ALL SHARED SPOOL MODS controls and allow the job to run immediately.

CLASSOPT={ ON | OFF }

CLASSOPT determines whether or not the classlim values that limit the number of active jobs on this system, in each class are enforced or not. ON means that the classlim value for each class is being enforced. OFF means that the classlim value for each class is NOT being enforced. Note - setting a low limit will not stop, or cancel any jobs, it will just prevent any new jobs from starting until the total number of jobs for each class is within the limit specified in the CLASSLIM statement for each class.

CLASSLIM(class specification)=###

CLASSLIM specifies the maximum number of jobs for each class that will be allowed to start. Valid CLASSLIM class specifications are;

A single character.
A range of characters ie. A-L or A-Z or 0-9
A masked value ie. * (meaning all classes)
A combination of the above separated by commas, ie. CLASSLIM(A-G,J,K,0-9)

Make Updated Modules available to the JES2 address space

Make the updated JES2 load modules available. That may require a refresh of LLA or copying a maintenance pack to production. All modules created by the JES2 usermods are link-edited into DDDEF SHASLNKE, so you may copy the individual modules, or recopy the entire JES2 library. The final step is to shutdown and restart JES2. This can be accomplished via 'rolling warm starts". NOTE - it is not possible to update all of these exits simply by using $REPEXIT or $ADDEXIT commands. Some must be active while the JES2 parms are still being read, others create dynamic table extensions and only take effect at JES2 initialization. Generally speaking, you may use $REPEXIT with exits other than exit 19 and 24, or the STJTABS and STSSMTBS modules. It is not necessary to shutdown all JES2 tasks in the MAS at the same time.

Perform any final testing - the JES2 mods are now installed!

Let me know where you are - for updates new releases and a coffee mug!

Drop me a line at SGMcCOLLEY@ALLTEL.NET and I will be happy to add you to my list of people to notify incase we find a bug, have a bug fix, release the next version and so on. I also have a limited number of 'SHARED SPOOL MODS' coffee mugs, they are free, just let me know where to send you one.

If you do by chance find a problem, please let us, know. While I can NOT offer to fix anything, and we never guarantee anything, I will do what I can. We run these mods as well and do not want bugs floating around to bite anyone, especially us. Same contact address.

How the exits work, and what each does –

Module STJTABS, plus macro definitions

This first mod is a Shared Spool Mod. The first LOADMOD statement that is added to your JES2 parms for these mods is for module STJTABS. STJTABS creates a dynamic $SCANTAB entry for the DISPLAY command, and dynamic $BERTTAB entry to define the JQE extensions, and a dynamic $PCETAB to create a special PCE to be used with a TQE chain used and maintained by the SHARED SPOOL MODS. There is one entry associated with this module - STPCENT - it is used in the definition of the PCE and is not referenced by any EXIT statements. STPCENT is called when one of our special PCEs is dispatched. Our special PCEs are dispatched at JES2 initialization, where they perform local initialization functions and wait to be dispatched. After initialization the special PCEs are dispatched only when a timer queue element that has been $WAITED with a $STIMER macro expires. The function of the code for the special PCE is solely to clear the chain of TQEs of posted elements, and to issue a $POST XEQ to cause JES2 to look for work to move from the input queue to execution. This Usermod also establishes the needed macros $STQNAME and $STJCTX, BOOLEAN, $SSMTB, $STTQE, $STTQEXW, STPCSMFD and updates the $USERCBS macro. If you already have changes other than these mods that affect the $USERCBS macro, you need to merge the previous changes with these.

Module STSCX50A, JES2 end of input user exit

The module handles special processing for the special case of a /*ROUTE XEQ HERE statement, setting the sysaff if needed. The real work of this exit is to call common module STSX2050, which is also called by exit20. STSX2050 copies information from the JCT to the JQE extension for later use in job selection.

Module STSCX04A and STSCX54A, exit04A and exit54a

This is part of the SHARED SPOOL MODS. Exits 04 and 54 perform the same ‘JCL statement scan’ function, but in different environments. Exit04A simply sets up a user environment, and calls exit54A to do the needed work. Exit54a validates the /*ROUTE XEQ RESNAME statement. If the RESNAME is a valid JES2 route value, we leave it alone and let JES2 handle it. If the RESNAME is not a valid route value, it is saved in the JCT until the rest of the job’s JCL is processed. It is finally used or discarded by exit6. The exit also performs specialized processing for the special case of /*ROUTE XEQ HERE.

Module STSCX54B, exit 54B

This is part of the SHARED SPOOL MODS; it is also used as an exit54. Note, in the exit statements that you add to JES2 parms, EXIT(54) calls two modules; 54a and 54b. This is the second one 54b. It parses and validates the /*BEFORE, AFTER, WITH WITHOUT, HOLDFOR, HOLDTIL and CNTL statements, and then saves the information in a JCT extension for the job. This exit makes extensive use of the new $STMTTAB facility to scan the JCL statements.

Module STSCX06A, exit 06

This is part of the SHARED SPOOL MODS; it is used as an exit 06. Module STSCX06A turns /*ROUTE XEQ schenvname into SCHENV= values. If a valid SCHENV environment has not been set with a SCHENV statement, or possibly be some other means, and if we found a /*ROUTE XEQ statement with a valid SCHENV name that also would not have been a valid destination (as in a valid /*ROUTE XEQ statement as intended by JES2 specs), then we use that value to set the SCHENV value in this exit.

Module STSC2050

This module is called by both exit 20 and exit 50. Exits 20 and 50 perform similar functions, but in different environments. The function of module STSC2050 is to move JCT extentions information into a BERT JQE extension at end of input time.

Module STSCX20A, exit 20

This is part of the SHARED SPOOL MODS support; and is used as an exit 20. This exit sets up the proper environment and then calls module STSX2050 which in turn copies JCT info into the JQE extension before the JCT is lost, when the last of the input JCL has been read. The information is actually written into a JQE extension, not the JQE itself. The extension is known as a BERT or Block Extension Reuse Table, and it was defined in LSES500. The BERT is incorporated into JES when the $USERCBS macro was updated with the $STQNAME macro that defines the extension. The $USERCBS macro update forces a reassembly of all JES2 modules that might reference the extension.

This exit also checks to see if a /*ROUTE XEQ HERE statement was in the job stream, and if one was present, and a sysaff was not already set via the jobcard or some other means, the sysaff is set to match the system name that the job was read from.

Module STSCX49A, exit 49

This is part of the SHARED SPOOL MODS; and is used as an exit 49. This module implements the before, after, with, without, holdtil, holdfor and cntl statements by rejecting or allowing JES2's suggested 'next' job in the job selection exit, exit49. Exit 49 is commonly referred to as the QGOT exit; it is called after JES2 has gone through its normal process of selecting the next job on the input queue that is ready for execution, but before the job actually starts running. Based on the /*BEFORE, AFTER, CNTL, HOLDFOR, HOLDTIL, WITHOUT, or WITH statements, this exit makes a final decision to allow the job to run, or to ask JES2 to locate another candidate job.

Modules STSCX19A, STSCX24A, and STSSMTBS

Module STSSMTBS, is a dynamic $SCANTAB that is used to process SSM parm statements in the JES2 parms, and as referenced in $T commands. $SCANTAB entries are extensively used to parse the SSM parms and $T SSM commands.

Module STSCX19A, which is used called for exit 19, is used to create a control block used by the SHARED SPOOL MODS in ECSA which is referenced by a Name/Token pair. The STSCX19A checks to see if there is a "left-over" copy of the control block for this JES2 member, and if one exists, deal with it appropriately. Since we made no provision for deleting the control block at JES2 termination, this check must be done to clean up the control block from a previous run, and possibly from an incomplete JES2 startup attempt. Finally a new control block is allocated, initialized with default values, and pointed to with a temporary Name/Token pair.

Module STSCX24A is used as exit 24. It checks for appropriate use of related SSM statement values. It also deletes a temporary SSMT name/token pair and replaces it with a permanent pair with global scope whose name is dynamically determined, and a name/token pair with a fixed name and a local scope.

SMF RECORD LAYOUTS

The SMF records produced by the SHARED SPOOL MODS are subtyped records all with the same SMF record number as specified in the SSM,SMFNUM=### statement. A different subtype is specified for each type of record.

```
** * ------------------------------------------------------------ * **
** * -- THIS GROUP OF DS'S IS INTEDED TO BE USED AS PART OF AN  -- * **
** * -- EXISTING DSECT, TO DESCRIBE AN SMF BUFFER FOR RECORD    -- * **
** * -- TYPE 216 (D8) TECH SUBTYPED SMF RECORDS.                -- * **
** * ------------------------------------------------------------ * **
SMFXLEN  DS    XL2               LRECL INCLUDING RDW
SMFXSEG  DS    XL2               SEGMENT - ALWAYS ZEROS
SMFXFLG  DS    XL1               B'0101 1110' INDICATES SUBTYPES
SMFXRTY  DS    XL1               SMF RECORD TYPE = 216 = X'D8'
SMFXTME  DS    XL4               TIME SINCE MIDNIGHT IN 1/100TH SEC.
*  TOD, USING FORMAT FROM TIME MACRO WITH BIN. INTVL
SMFXDTE  DS    PL4               X'01YYDDDF'
*  DATE IN PACKED DECIMAL FORM: 01YYDDDF
SMFXSID  DS    XL4               SYSID FROM ( SID )
SMFXSSI  DS    XL4               SUBSYS ID  (SSID = TECH) OR BLANKS
SMFXSTY  DS    XL2               RECORD SUBTYPE X'01'-X'FF'
*                  ****  PROGRAM EXECUTION TRACKING SUBTYPES  ****
*                          X'01' = TECH PGM EXECUTION
*                          X'02' = TECH PGM EXECUTION DUP LIB.
*                          X'03' = TECH PGM EXECUTION DUP LIB.
*                          X'04' = TECH PGM EXECUTION DUP LIB.
*                          X'05' = TECH PGM EXECUTION DUP LIB.
*                  ****  S.S.M. = SHARED SPOOL MODS SUBTYPES  ****
*                          X'40' = SSM REJECTION INFORMATION
*                          X'41' = SSM JOB PASSED SELECTION
*              FUTURE      X'42' = SSM OPERATOR ACTIONS ($T CMDS)
*                          X'43' = $SJ - ALLOWED OR REJECTED
*                          X'44' = SSM JECL CARD ACCEPTED
*              FUTURE      X'45' = SSM JECL CARD REJCTED JCL ERROR
*                          X'46' = SSM JOB SELECTION REDRIVEN
*              FUTURE      X'47' = JES2 SSM PARM ACCEPTED
*              FUTURE      X'48' = RESERVED FOR SSM
*              FUTURE      X'49' = RESERVED FOR SSM
SMFXNUMT DS    XL2               NUMBER OF TRIPLETS (SUBTYPES 1-5= 2)
*
SMFXRESV DS    XL2               LENGTH OF SELF-DEFINING SECTION
*** SELF-DEFINING SECTION ***
* - FIRST TRIPLET - PRODUCT SECTION
OFFPRD01 DS    XL4               OFFSET FROM RDW TO PROD. SECTION
LENPRD01 DS    XL2               LENGTH OF PRODUCT SECTION
NUMPRD01 DS    XL2               NUMBER OF PRODUCT SECTIONS
*- SECOND TRIPLET - SUBTYPED DATA SECTION
OFFTEC01 DS    XL4               OFFSET FROM RDW TO SUBTYPED DATA
```

```
LENTEC01 DS    XL2               LENGTH OF SUBTYPED SECTION
NUMTEC01 DS    XL2               NUMBER OF SUBTYPED SECTIONS
*
SDSEND   EQU   *                 END OF SELF DEFINING SECTION
SMFD8SSD EQU   SDSEND-OFFPRD01   EQU'D LEN OF SELF DEFINING ssm SECT.
SMFD8TSD EQU   SDSEND-OFFPRD01   EQU'D LEN OF SELF DEFINING tech SECT.
*
* THE PRODUCT SECTION(S) FOR JES2 SSM GOES HERE
*
         ORG   SDSEND            ORG TO END OF SELF DEFINING SECITON
*
PRDOFS   EQU   *-SMFXLEN         OFFSET TO PRODUCT SECTION
SMFD8STY DS    XL2               SUBTYPE - REPEATED - JUST IN CASE
SMFD8SVR DS    XL4               SAME AS UBRVRM AND UJCXVRM
SMFD8SID DS    XL16              C'SHARED SPOOL MOD'
PRDLENS  EQU   *-SMFD8STY        LENGTH OF THE SSM PRODUCT SECTION
*
PRD8JZZ  EQU   *                 END OF SSM PRODUCT SECTION
*
* THE SUBTYPED SHARED SPOOL MODS DATA GOES IN HERE
*
STDOFFS  EQU   *-SMFXLEN         OFFSET TO SUBTYPED SSM DATA
*
SMFD8S40 DS    XL2               SUBTYPE - X'0040' SSM REJECTION INFO
SMFD80JI DS    XL4               JOBID
SMFD80JN DS    CL8               JOBNAME
SMFD80SI DS    XL4               NODE ID REJECT TOOK PLACE ON
SMFD80TE DS    XL8               NODE NAME REJECT TOOK PLACE ON
SMFD80GN DS    XL8               NODE NAME REJECT TOOK PLACE ON
SMFD80ME DS    CL12              REJECTION REASON
SMFD80XT DS    CL8               STCK FORMAT DATE AND TIME
SMFLNS40 EQU   *-SMFD8S40        LENGTH OF SUBTYPED DATA
SMFXLS40 EQU   *-SMFXLEN         LENGTH OF THE ENTIRE RECORD
*
         ORG   PRD8JZZ           ORG TO END OF SSM PRODUCT SECTION
*
SMFD8S41 DS    XL2               SUBTYPE - X'0041' SSM JOB SELECTED
SMFD81IN DS    XL2               INPUT NODE ID       JQEINPND
SMFD81XN DS    XL2               EXECUTION NODE ID   JQEXEQND
SMFD81CD DS    CL1               JQE CREATION TIME    - JQXCRTME
SMFD81JC DS    CL1               JOB CLASS   JQEJCLAS
SMFD81JI DS    XL4               JOBID
SMFD81JN DS    CL8               JOBNAME JQEJNAME
SMFD81RI DS    CL8               USERID OF JOB OWNER    - JQEUSRID
SMFD81SL DS    CL8               SECURITY LABEL OF JOB   - JQESECLB
SMFD81XT DS    CL8               STCK FORMAT DATE AND TIME - THIS REC
SMFD81SE DS    CL16              SCHEDULING ENVIRONMENT NAME -JQASCHE
SMFD81TE DS    XL8               NODE NAME ACCEPTED ON
SMFD81GN DS    CL8               XCF GROUP NAME ACCEPTED ON
SMFLNS41 EQU   *-SMFD8S41        LENGTH OF SUBTYPED DATA
SMFXLS41 EQU   *-SMFXLEN         LENGTH OF THE ENTIRE RECORD
*
         ORG   PRD8JZZ           ORG TO END OF SSM PRODUCT SECTION
```

```
*
SMFD8S42 DS    XL2               SUBTYPE - X'0042' SSM OPER CMDS
SMFD82XT DS    CL8               STCK FORMAT DATE AND TIME - THIS REC
* THE ONLY THINGS AN OPERATOR CAN CHANGE ARE IN THE ECSA AREA -
* HERE IS A BEFORE AND AFTER COPY OF THE ECSA AREA
SMFD82NN DS    XL8               NODE NAME ACCEPTED ON
SMFD82NX DS    XL8               NODE ID COMMAND ENTERED ON
SMFD82CM DS    CL140             THE COMMAND ITSELF (IF WE CAN GET IT)
SMFD82CB DS    XL(SSMTBLEN)      THE ECSA AREA ITSELF (BEFORE)
SMFD82CA DS    XL(SSMTBLEN)      THE ECSA AREA ITSELF (AFTER)
SMFLNS42 EQU   *-SMFD8S42        LENGTH OF SUBTYPED DATA
SMFXLS42 EQU   *-SMFXLEN         LENGTH OF THE ENTIRE RECORD
*
         ORG   PRD8JZZ           ORG TO END OF SSM PRODUCT SECTION
*
SMFD8S43 DS    XL2               SUBTYPE - X'0043' $SJ ALLOWED OR NOT
SMFD83DT DS    CL8               STCK FORMAT DATE AND TIME - THIS REC
SMFD83NN DS    CL1               A=$SJ IS ALLOWED ;;;; X=$SJ REJECTED
SMFLNS43 EQU   *-SMFD8S43        LENGTH OF SUBTYPED DATA
SMFXLS43 EQU   *-SMFXLEN         LENGTH OF THE ENTIRE RECORD
*
         ORG   PRD8JZZ           ORG TO END OF SSM PRODUCT SECTION
*
SMFD8S44 DS    XL2               SUBTYPE - X'0044' JECL CARD ACCPETED
SMFD84IN DS    XL2               INPUT NODE ID       JQEINPND
SMFD84CD DS    CL1               JQE CREATION TIME    - JQXCRTME
SMFD84JC DS    CL1               JOB CLASS   JQEJCLAS
SMFD84JI DS    XL4               JOBID
SMFD84JN DS    CL8               JOBNAME JQEJNAME
SMFD84RI DS    CL8               USERID OF JOB OWNER    - JQEUSRID
SMFD84XT DS    CL8               STCK FORMAT DATE AND TIME - THIS REC
SMFD84SE DS    CL16              SCHEDULING ENVIRONMENT NAME -JQASCHE
SMFD84MG DS    CL60              DETAILED INFO FOR SMF
SMFD84JA DS    CL(UJCXSLN1)      THE STQNAME (JCT EXTENSION) AFTER
SMFLNS44 EQU   *-SMFD8S44        LENGTH OF SUBTYPED DATA
SMFXLS44 EQU   *-SMFXLEN         LENGTH OF THE ENTIRE RECORD
*
         ORG   PRD8JZZ           ORG TO END OF SSM PRODUCT SECTION
*
SMFD8S45 DS    XL2               SUBTYPE - X'0045' JECL CARD REJECTED
SMFD85IN DS    XL2               INPUT NODE ID       JQEINPND
SMFD85CD DS    CL1               JQE CREATION TIME    - JQXCRTME
SMFD85JC DS    CL1               JOB CLASS   JQEJCLAS
SMFD85JI DS    XL4               JOBID
SMFD85JN DS    CL8               JOBNAME JQEJNAME
SMFD85RI DS    CL8               USERID OF JOB OWNER    - JQEUSRID
SMFD85SL DS    CL8               SECURITY LABEL OF JOB   - JQESECLB
SMFD85XT DS    CL8               STCK FORMAT DATE AND TIME - THIS REC
SMFD85SE DS    CL16              SCHEDULING ENVIRONMENT NAME -JQASCHE
SMFD85TE DS    XL8               NODE NAME ACCEPTED ON
SMFD85MG DS    CL140             JECL CARD IMAGE PROCESSED
SMFLNS45 EQU   *-SMFD8S45        LENGTH OF SUBTYPED DATA
SMFXLS45 EQU   *-SMFXLEN         LENGTH OF THE ENTIRE RECORD
```

```
*
         ORG   PRD8JZZ            ORG TO END OF SSM PRODUCT SECTION
*
SMFD8S46 DS    XL2                SUBTYPE - X'0046' QSEL IS REDRIVEN
SMFD86IN DS    XL2                NODEID SOMEWHERE IN $HCT OR $HCCT
SMFD86XT DS    CL8                STCK FORMAT DATE AND TIME - THIS REC
SMFLNS46 EQU   *-SMFD8S46          LENGTH OF SUBTYPED DATA
SMFXLS46 EQU   *-SMFXLEN           LENGTH OF THE ENTIRE RECORD
*
         ORG   PRD8JZZ            ORG TO END OF SSM PRODUCT SECTION
*
SMFD8S47 DS    XL2                SUBTYPE - X'0047' SSM PARM ACCEPTED
SMFD87IN DS    XL2                NODEID SOMEWHERE IN $HCT OR $HCCT
SMFD87XT DS    CL8                STCK FORMAT DATE AND TIME - THIS REC
SMFD87PM DS    CL256              PARM VALUE ACCEPTED
SMFD87EC DS    XL(SSMTBLEN)       THE ECSA AREA ITSELF
SMFLNS47 EQU   *-SMFD8S47          LENGTH OF SUBTYPED DATA
SMFXLS47 EQU   *-SMFXLEN           LENGTH OF THE ENTIRE RECORD
*
         ORG   PRD8JZZ            ORG TO END OF SSM PRODUCT SECTION
*
SMFD8S48 DS    XL2                SUBTYPE - X'0047' SSM PARM ACCEPTED
SMFD8801 DS    XL1                SOME DATA TO RECORD(UNUSED FOR NOW)
SMFD8802 DS    XL1                  MORE DATA TO RECORD
SMFLNS48 EQU   *-SMFD8S48          LENGTH OF SUBTYPED DATA
SMFXLS48 EQU   *-SMFXLEN           LENGTH OF THE ENTIRE RECORD
*
         ORG   PRD8JZZ            ORG TO END OF SSM PRODUCT SECTION
*
SMFD8S49 DS    XL2                SUBTYPE - X'0047' SSM PARM ACCEPTED
SMFD8901 DS    XL1                SOME DATA TO RECORD(UNUSED FOR NOW)
SMFD8902 DS    XL1                  MORE DATA TO RECORD
SMFLNS49 EQU   *-SMFD8S49          LENGTH OF SUBTYPED DATA
SMFXLS49 EQU   *-SMFXLEN           LENGTH OF THE ENTIRE RECORD
*
*   end of record layout *
```

A few final notes for /*BEFORE and /*AFTER

A few final notes concerning the relationship between /*BEFORE, and /*AFTER. There is a peculiar type of problem that has been around since the first design of the Mellon Mods, and I only bring it up here to clarify the potential problem, and offer possible solutions for the problem.

Many people try to use these statements, and stack two or more jobs in the same PDS member and submit them all at the same time with one SUBMIT command. This usually works as expected, but sometimes JES2 does not 'seem' to PROCESS the jobs in the order they appear in the submitted member. Actually JES2 always processes in the order they are presented to JES2 in, however these exits do not 'see' the jobs for the purposes of job selection until after they have completed the conversion process and you could have several different converter tasks running under JES2 which could easily lead to one job with a very few DD statements finishing the conversion process and being placed on the next queue long before a job with many different DD statements that was submitted just before the smaller job. This can result in a job with a /*AFTER statement for a prior job you think JES2 has already seen and processed because of the sequence the jobs are in when first submitted, being processed and initiated before JES2 ever finishes reading the job that is the object of the /*AFTER statement. This problem seldom crops up, but can be VERY confusing and difficult to explain when it does. This is also the way the mods were originally designed, and have been working for many, many years.

This problem can be avoided by making sure that jobs with /*BEFORE and /*AFTER requirements are submitted separately from each other and in an appropriate sequence, however that may not be practical in all situations.

In an effort to mitigate the potential problem described in the scenario above we have introduced the BEAFTER parm that can be set to the value of PREMOD or DELAY. If the parm is set to PREMOD, or allowed to default, the processing will continue as it always has in the past. If however DELAY is set, then the value in parm BATIME will be used as a delay time for all jobs. By delaying all jobs a few seconds after they are first available for execution, we are hoping that any 'slow' jobs, ones that may take a long time in the conversion queue, will have a chance to catch up, and be available for consideration when a job with a /*BEFORE or /*AFTER card is considered for possible execution. There is a trade off for this type of processing however, it obviously introduces a small delay for all jobs before execution, even though only a small number of jobs will potentially benefit from the delay. There may of course be other reasons why a shop may want to delay jobs before allowing them to become available for execution, I just can't think of any.

There are three other methods that I have come across that can also be used to alleviate this special situation that can come up in regard to /*BEFORE and /*AFTER processing. First you may opt to have only a single conversion task (see JES2 parm PCEDEF CNVTNUM) although this is NOT recommended, it will force jobs to convert one at a

time, and the ‘out of order’ condition can not occur. Next you may add one of the new /*WAITFOR 00:00:04 statements to all jobs associated with /*BEFORE /*AFTER control statements. This has the same effect as specifying BATIME=4 and BEAFTER=DELAY, except that it only affects the jobs that actually might be affected. Finally we could code a POSITIVE recognition that the before or after jobname condition is being met. This is considerably more complex than it first sounds, and introduces yet more potential problems, such as; how far back do we reference job completions to see if a /*AFTER jobname, has been satisfied? It is however something that I intend to do in the future.

In the meantime, I would strongly suggest that you either use the BEAFTER option of DELAY or add /*HOLDFOR 00:00:04 cards to the jobs using /*BEFORE and /*AFTER statements.

```
Here is a full example of this type of problem - two jobs are in the
same member to be ‘sub’ed at the same time, although a scheduling
package that submits tow jobs at very nearly the same time can have the
same effect.

MEMBERA Contains.
      //FIRSTJOB  JOB  (123,abc),CLASS=X
      /*BEFORE  LASTJOB
      //STEP001    EXEC  PGM=ANYPGM
      //DD001      DD   DSN=A.B.C.D,DISP=SHR
      //DD002      DD   DSN=A.B.C.E,DISP=SHR
      //…… (another 100 dd statements go in here)
      //LASTJOB  JOB  (123,abc),class=x
      /*AFTER FISTJOB
      //STEPONLY  EXEC PGM=IEFBR14

MEMBERA is submitted -

  Now the intent is clearly that LASTJOB should run after FIRSTJOB, but
if they are both submitted at the same time, each job could be assigned
a different converter processor.  Since LASTJOB has no DD statements
and only one JOB and one EXEC statement, it will complete conversion
and move to the XEQ queue long before FIRSTJOB is converted.  Then if
while FIRSTJOB is still in conversion processing, LASTJOB is selected
by JES2 as a potential job to execute, the SHARED SPOOL MODS will check
to see if FIRSTJOB is either in the same queue waiting to execute or is
already executing, but since it has not even finished conversion it
will not be found, and LASTJOB will be allowed to run (with the exits
making the assumption that FIRSTJOB must have already run).  Later,
maybe only a few milliseconds later, FIRSTJOB may finish conversion and
be selected for possible execution.  The exits will check to see if
jobname LASTJOB is in the input queue, will not find it there, and will
allow the job to execute.

  Now the intent is clearly that LASTJOB should run after FIRSTJOB, but
if they are both submitted at the same time, each job could be assigned
```

a different converter processor. Since LASTJOB has no DD statements and only one JOB and one EXEC statement, it will complete conversion and move to the XEQ queue long before FIRSTJOB is converted. Then if while FIRSTJOB is still in conversion processing, LASTJOB is selected by JES2 as a potential job to execute, the SHARED SPOOL MODS will check to see if FIRSTJOB is either in the same queue waiting to execute or is already executing, but since it has not even finished conversion it will not be found, and LASTJOB will be allowed to run (with the exits making the assumption that FIRSTJOB must have already run). Later, maybe only a few milliseconds later, FIRSTJOB may finish conversion and be selected for possible execution. The exits will check to see if jobname LASTJOB is in the input queue, will not find it there, and will allow the job to execute.