

- * Started Task Control (STC) are procedures started from the console which remain in its own address space.
- * All z/OS subsystems and most system software products (e.g., CICS, IMS, DB2, WAS) are started as a STC.
- * STC PROCs, like application PROCs, reside in PROCLIBs (JCL Procedure Libraries) which are defined in the //PROCLIB DD statement of the JES2 startup PROC (SYS1.PROCLIB/JES2).

```

SYS1.PROCLIB(JES2) - 09.44          CSNNMS 98091 0001
***** Top of Data *****
//JES2  PROC DSN=SYS1.PROCLIB      * STANDARD PROCLIB
//        DSN=SYS1.PROCLIB.1BY     * SYSTEM LEVEL  PROCs
//        DSN=SYS1.SYSUPT.PROCLIB  * PROG. PRODUCT PROCs
//        DSN=SYS1.USER.PROCLIB    * USER PROCLIB
//        DSN=SYS1.COMBINED.PROCLIB * USER PROCLIB
//        DSN=SYS1.SPOOL.PROCLIB   * SPOOL PROCs
//        DSN=SYS1.SPOOL.PACK.SERIAL(3300) * SPOOL PACK SERIAL (3300)
//        DSN=SYS1.DEFAULT.NAME.ALTERNATE * DEFAULT NAME ALTERNATE
//        DSN=SYS1.SYNSMMS         * SYNSMMS
//        DSN=SYS1.SYSNAME         * SYSNAME
//        DSN=SYS1.HRS.NJZ/CTS     * HRS_NJZ/CTS
//        DSN=SYS1.DRAIN          * SPARE INITIATORS
//*****
//MODIFIED JES2 START PROCEDURE - WASHINGTON SYSTEMS CENTER
//*****
//PROCLIB EXEC DSN=SYS1.PROCLIB.1BY(1,1) TIME=1440
//*****
//PERFORM=SPHIN DUMMY PERFORM FOR TRACKING
//*****
//STERLIB DD DSN=STERLIB.DISP=SHR (MUST BE AUTHORIZED)
//*****
//DEFAULT PROCEDURE LIBRARIES LIST
*****
  
```

The STC PROCLIBs DDname are identified by the PROCLIB= parameter of the STCCCLASS statement which is defined in the datasets (i.e., JES2 initialization parameter file) that are referenced by the //HASPPARM and //ALTPARM DD statements.

- * When you start a started task, z/OS writes the demand select job's JCL into the STCINRDR data set.
- * TSO logon JCL is written into the TSOINRDR data set. The DD statements for these data sets are defined in the MSTJCLxx member: //STCINRDR DD - Defines the internal reader where started tasks are to be sent.

NOTE: Internal reader for started tasks is STCINRDR. The internal reader facility is a logical device similar to a card reader that allows you to submit jobs to JES. You can also read job streams from tape, disk or any QSAM-supported device through the internal reader to JES by using the procedure named RDR. Using the internal reader facility, you can submit jobs from time-sharing logons, started tasks, or other jobs. The ability to submit jobs from currently running jobs or tasks is especially powerful. This ability gives the programmer the flexibility to have a job that reaches a point successfully to submit another job for execution.

The system programmer defines internal readers used to process all batch jobs inclusive of STCs and TSO requests.

JES2 initialization statements define the internal readers which JES2 allocates during its initialization processing.

The internal readers for batch jobs can be used for STCs and TSO requests.

- * The INTRDR initialization statement defines a logical device for jobs entering the system through the internal reader facility.
- * One of these attributes is the maximum number of jobs that can enter the system concurrently from the internal reader facility.
- Specify the maximum number (up to 32765) through the RDINUM parameter of the INTRDR statement.
- The default number of internal readers assigned by JES2 is four (1 for started tasks, 1 for time sharing users, and 2 for batch jobs).

```

***** INTRDR Defaults *****
//*****
//INTRDR  DEFAULTS Example - JES2 Startup SYS1.PARMLIB(JESPARM)
//*****
INTRDR  AUTH=(JOB=NO,DEVICE=NO,SYSTEM=YES),
        BATCH=YES,           /* ALLOW BATCH JOBS TO USE          INWC */
        HOLD=NO,            /* DON'T HOLD JOBS READ          INWC */
        HONORLIM=NO,        /* DO OUTPUT EXCESSION FOR INTRDR  INWC */
        PRTYINC=0,          /* DON'T PRTY AGE JOBS           INWC */
        PRTYLIM=15,         /* LIMIT JOB PRTY TO 15          INWC */
        RDINUM=50,          /* NUMBER OF INTERNAL RDRS       INWC */
        TRACE=NO            /* ALLOW TRACING                  INWC */
//*****
  
```

Determining whether to use a Started Task

- * When you determine where and when you want a set of JCL to run, you will consider using batch jobs or started tasks.
- Batch jobs are scheduled by a job entry subsystem (JES) and are scheduled to run based on the resources they require and their availability, or based on controls you put on the batch system.
- Controlling where and when a batch job runs is more complex than using a started task.
- A started task is a set of JCL that is run immediately as the result of a START command (see lower right).
- Started tasks are generally used for critical applications.
- The advantages to using started tasks are:

1. You can control where and when your set of JCL is run. For example, you can have the set of JCL started at each IPL of the system.
2. You can specify both static system symbols and JCL symbols in the JCL. Static system symbols and JCL symbols provide additional control over JCL that is used on different systems. For example:
 - > When access to production data sets is controlled to protect critical business data, you can specify symbols that represent test data sets. After testing the data sets, you can change the values of the symbols to represent production data sets without changing the source JCL.
 - > When you need to swap in an older level of a subsystem while diagnosing problems with a newer level, you can change the values of symbols to represent the older subsystem without changing the source JCL.

NOTE: In the past, some users set up batch jobs that controlled their programs. Users allocated a PDS, added JOB JCL to a member of the PDS, and then read the PDS member into an internal reader; these actions initiated a batch job for the started task. While this method afforded some advantages, it did not allow for symbolic support.

- Caveats:**
- * The TYPRUN parameter is not supported for started tasks. If TYPRUN is specified, the job will fail.
 - * The security environment of started tasks is defined using a RACF class, not through the USER, PASSWORD, GROUP, and SECLABEL parameters. If these parameters are specified, the started task will fail.
 - * The CLASS parameter is not supported for started tasks in a JES2 environment.
 - * For started tasks in a JES3 environment, all class related attributes and functions are ignored except device fencing, SPOOL partitioning, and track group allocation.

CheatSheet

#63 z/OS

STARTED TASK CONTROL

```

//IEFPROC EXEC PGM=IEBEDIT
//SYSUT1 DD DDNAME=IEFRDR
//IEFRDR DD DSN=NULLFILE,
//          DISP=OLD
//SYSUT2 DD SYSOUT=(A,INTRDR)
//SPRINT DD SYSOUT=A
//SYSIN DD DUMMY
  
```

Using the RDR Procedure: shows the JCL procedure IBM supplies for you to use the internal reader to read jobs from tape, disk or any QSAM-supported device.

Master JCL

MSTJCL00

IBM supplies default master JCL in the MSTJCL00 load module in SYS1.LINKLIB. Lower right displays MSTJCL00 as it exist before it is assembled and Link-edited into SYS1.LINKLIB,

WLM uses a special subsystem type STC to define Started Tasks which also includes system component address spaces such as the TRACE and PC/AUTH address spaces.

RACF - Even if your installation uses the STARTED class, you must have a started procedures table (ICHRIN03). RACF cannot be initialized if ICHRIN03 is not present. A dummy ICHRIN03 is shipped with and installed by RACF.

IBM recommends using JOBNAME= parameter, as it assigns a "proper" job name that's available to SMF.

See active MSTJCLxx: SYS1.PARMLIB(MSTJCL00)

Setting up started tasks with the Master JCL * The source JCL for a started task can be a job (source JCL that begins with a JOB statement) or a cataloged procedure.

- * If the source JCL for a started task is a job, the member containing the JCL must be part of a data set in the IEFPDSI DD or the IEFJOBS DD concatenation of MSTJCLxx. (If the member is not part of a data set in the IEFPDSI or IEFJOBS concatenation of MSTJCLxx, the procedures that act as source JCL for other started tasks will not be found.)
- * IBM suggests that you define a new data set in MSTJCLxx (pointed to by the IEFJOBS DD statement) that will contain the tailored JCL to support started tasks.

To create source JCL that is a job (with, for example, JOB, JES2 JECL, and OUTPUT statements) for a started task, you must first decide whether these jobs should be mixed with procedures (part of the IEFPDSI concatenation) or placed into a separate data set (part of the IEFJOBS concatenation) containing only jobs.

Using IEFJOBS to define started tasks If MSTJCLxx contains a DD named IEFJOBS, the source JCL for a started task can be placed in one of the data sets within the IEFJOBS concatenation.

- Doing so allows jobs and the procedures they invoke to have the same name.
- These jobs can contain the minimum set of JCL needed to define the job level characteristics (for example, JOB statements, JCLLIB, and JECL), and then either invoke an existing procedure or use the INCLUDE keyword to invoke the desired set of JCL.

- Use a name that allows you to quickly identify the data set. For example, the entry in MSTJCLxx could appear as: //IEFJOBS DD DSN=SYS1.STCJOBS,DISP=SHR

> For this example, you can put the source JCL for the started tasks in the SYS1.STCJOBS data set.

* Consider these reasons for using IEFJOBS to define started tasks:

1. z/OS and other products ship procedures placed in procedure libraries. Your modifications to the members that contain your source JCL for started tasks might be lost if a new version of the procedure is received and placed into the procedure library.
 2. Defining data sets for IEFJOBS allows you to have the member name containing the source JCL invoke a procedure of the same name. For example, member name DUMPCHK in the IEFJOBS data set SYS1.STCJOBS is a job that invokes procedure DUMPCHK in SYS1.PROCLIB. This minimizes the effect on commands that might be issued from a variety of sources, and allows job parameters to be added transparently.
- Note:** Do not attempt to change the IEESYSAS procedure to a started job, or place a member named IEESYSAS in the IEFJOBS data set concatenation. IEESYSAS is reserved for use by the system for starting address spaces.
3. Maintaining procedures in procedure libraries and jobs in an IEFJOBS data set can reduce potential confusion. For example, if you place a job in a procedure library, a user could mistakenly assume that the job is a procedure and invoke it as a procedure within a job; a job invoked as a procedure within a job fails.

Using IEFPDSI to define started tasks * As shipped, MSTJCL00 contains an IEFPDSI DD statement that points to only one procedure library (SYS1.PROCLIB) used by the master subsystem (see upper left).

- You can place individual jobs in any other procedure libraries pointed to by the IEFPDSI DD statement.
- As an alternative to placing individual jobs in any of the other procedure libraries pointed to by the IEFJOBS DD statement you can instead put jobs in the members of SYS1.PROCLIB or other data sets in the IEFPDSI concatenation.
- > In this case, you must make sure that there is not another member of the same name in one of the data sets in the concatenation ahead of the data set containing the member with the source JCL.

Started task processing * When you start a started task, the system determines whether the START command refers to a procedure or a job. (The system validates that the IEFJOBS DD exists within the MSTJCLxx member.)

* If the IEFJOBS DD exists, the system searches the IEFJOBS DD concatenation for the member requested in the START command.

- If there is no member by that name in the IEFJOBS concatenation, or if the IEFJOBS concatenation does not exist, the system searches the IEFPDSI DD for the member requested in the START command.
- If a member is found, the system examines the first record for a valid JOB statement and, if one exists, uses the member as the JCL source for the started task.
- If the member does not have a valid JOB statement in its first record, the system assumes that the source JCL is a procedure and creates JCL to invoke the procedure.
- After JCL source has been created (or found), the system processes the JCL.

Determining the Source JCL for the Started Task * If you decide to use a started task, you must then determine what the source JCL will be and where the JCL will be located.

- The source JCL can be a JOB (located in a member of a data set defined in the IEFJOBS or IEFPDSI concatenation of master JCL) or a procedure (located in a subsystem procedure library, for example, SYS1.PROCLIB).
- In the latter case, the system will process only the JCL associated with the first JOB statement in the procedure; it will bypass the second and subsequent jobs.

Note: In the past, the source JCL for started tasks was always a procedure.

```

S membername[,identifier][,devicetype | ,[/]devnum][,volumeserial]
  [,parameters][,JOBNAME=jobname][,JOBACCT=acct_info]
  [,SUB=subsystemname][,keyword=option[,keyword=option]...]
  
```

Started tasks are initiated by the START command which identifies the member that contains the source JCL for the task.

MSTJCL00 CSECT

```

DC CL80'//MSTJCL00 JOB MSGLEVEL=(1,1),TIME=1440'
DC CL80'// EXEC PGM=IEEMB860'
DC CL80'//STCINRDR DD SYSOUT=(A,INTRDR)'
DC CL80'//TSOINRDR DD SYSOUT=(A,INTRDR)'
DC CL80'//IEFPDSI DD DSN=SYS1.PROCLIB,DISP=SHR'
DC CL80'//SYSUADS DD DSN=SYS1.UADS,DISP=SHR'
DC CL80'/*'
END
  
```

Note: MSTJCL does not contain the START command that starts the primary job Entry (JES) subsystem during master scheduler initialization. You can specify an alternate version of the master JCL if need be to include IEFJOBS.

DD statements needed to define the internal reader data sets for started task control and TSO/E logons. SYS1.UADS, a system data set used in TSO/E logons and terminal communications.