214

MVS update

July 2004

In this issue

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Nicole Thomas
E-mail: nicole@xephon.com

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs $505.00 in the USA and Canada; £340.00 in the UK; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 2000 issue, are available separately to subscribers for £29.00 ($43.50) each including postage.

MVS Update on-line

Code from *MVS Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/mvs; you will need to supply a word from the printed issue.

Contributions

When Xephon is given copyright, articles published in *MVS Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

Dynamic and very easy allocation

The availability of BPXWDYN has been documented starting with z/OS 1.4. This module permits access to SVC 99 (dynamic allocation services) as well as SVC 109 (dynamic output) and has simple and versatile rules.

This function is available with all the invocation methods (batch, TSO, program/call, etc) and especially with every programming language of Language Environment.

The required syntax is in text format and is very similar to the parameters required by the TSO command Allocate.

The commands are:

- Alloc – dynamic allocation.
- Free – dynamic unallocation of DDname.
- Concat – concatenate DDname.
- Outdes – create an output descriptor.

Complete details can be found in:

- z/OS V1R4.0 *Using REXX and z/OS Unix System Services*.
- *z/OS TSO/E Command Reference*.

CALL EXAMPLES

Call examples are shown below:

- REXX:

```
/*   rexx  */
Call BPXWDYN (" Alloc dd(test) da(mytest.mvs.dataset) shr ")
```

- Assembler:

```
    LOAD  EP=BPXWDYN              Call BPXWDYN
    LTR   R15,RØ
```

```
         BZ    ROUT_ELOAD              Go Error_Routine

         OI    BPXPARM,X'80'
         LA    R1,BPXPARM              Load parmlist
         BALR  R14,R15
         LTR   R15,R15
         BNZ   ROUT_EALLC              Go Error_Routine

BPXPARM  DC    AL4(LENGBPX)
LENGBPX  DC    AL2(LENGTXT)
TEXTBPX  DC    C'ALLOC FI(DD1)  DSN(FILE.TEXT.BPX) NEW CATALOG '
         DC    C'UNIT(SYSALLDA)  CYL  SPACE(1,1)  '
         DC    C'DSORG(PS) RECFM(F,B) LRECL(80)  '
LENGTXT  EQU   *-TEXTBPX
```

- PL/I:

```
DCL     PLIRETV BUILTIN;
DCL     BPXWDYN EXTERNAL ENTRY OPTIONS(ASM INTER RETCODE);
DCL     STR_BPX CHAR(080) VAR
        INIT('ALLOC FI(DD1) DA(FILE.TEXT.BPX) SHR');
FETCH BPXWDYN;
CALL     BPXWDYN(STR_BPX);
```

APPLICATION EXAMPLE

It's clear how this Unix System Service becomes particularly useful in programming languages such as COBOL, which traditionally don't have the tools to dynamically allocate a file.

Normally it was necessary to write an Assembler routine using the difficult syntax of SVC 99, and then use that within the main program. Now, with BPXWDYN, all those home-made routines are no longer necessary because the technique is a standard part of the base software.

My company has a wealth of COBOL applications, so we often need to work with scores of different files that have similar DCBs in the main program but have a single 'select DDname'.

Until recently this was all accomplished through a rather complex Assembler routine that modified the DDname in the TIOT area of the 'Select/cobol' each time in order to build the parameters for the dynalloc macro. Moreover, this technique was further limited by the need to have the linkage editor attribute set at AC=1.

With BPXWDYN, everything is now much simpler and self-explanatory.

Source JCL:

```
//your_jobcard .......................
//STEP1     EXEC PGM=MYPGM
//STEPLIB   DD DISP=SHR,DSN=your_steplib
//SYSIN     DD   *
DSNAME_FILE1
DSNAME_FILE2
DSNAME_FILE3
DSNAME_FILE4
.....................................
$$$$$$$$$$$
/*
```

COBOL source:

```
       IDENTIFICATION  DIVISION.
       PROGRAM-ID.     MYPGM.
      ** ------------------------------------------------ **
       ENVIRONMENT DIVISION.
      ** ------------------------------------------------ **
       CONFIGURATION   SECTION.
       INPUT-OUTPUT    SECTION.
       FILE-CONTROL.
           SELECT DDNAMEØ1 ASSIGN TO UT-S-DDNAMEØ1.
      ** ------------------------------------------------ **
       DATA DIVISION.
      ** ------------------------------------------------ **
       FILE SECTION.
       FD   DDNAMEØ1
            RECORDING MODE IS F
            BLOCK  CONTAINS Ø RECORDS.
       Ø1 REC-DDNAMEØ1               PIC  X(8Ø).
      ** ------------------------------------------------ **
       WORKING-STORAGE  SECTION.
      ** ------------------------------------------------ **
       Ø1  W-REC-DDNAMEØ1            PIC X(8Ø).
       Ø1  W-DYNFREE                 PIC X(2Ø)
                     VALUE " FREE DD(DDNAMEØ1)".
       Ø1  W-DYNALLC.
           Ø3 FILLER                 PIC X(27)
                      VALUE " ALLOC DD(DDNAMEØ1) SHR DA(".
           Ø3 W-DYNALLC-DSN          PIC X(46).
      ** ------------------------------------------------ **
       PROCEDURE DIVISION.
       EXECØØ.
            ACCEPT  W-DYNALLC-DSN FROM SYSIN.
```

```
  INSPECT W-DYNALLC-DSN REPLACING FIRST "  " BY ") ".
  IF W-DYNALLC-DSN(1:2) IS = "$$"
     GOBACK.
   CALL "BPXWDYN" USING W-DYNFREE.
    IF RETURN-CODE IS NOT = Ø
     DISPLAY "RC/FREE   > " RETURN-CODE
      GOBACK.
   CALL "BPXWDYN" USING W-DYNALLC.
    IF RETURN-CODE IS NOT = Ø
     DISPLAY "RC/ALLOC  > " RETURN-CODE
      GOBACK.
  OPEN     INPUT DDNAMEØ1.
  READ     DDNAMEØ1 INTO W-REC-DDNAMEØ1
     AT END GOBACK.
  DISPLAY  " DATASET NAME          : " W-DYNALLC-DSN.
  DISPLAY  " FIRST RECORD/DDNAMEØ1 : " W-REC-DDNAMEØ1.
.............................
.............................
.............................

  CLOSE    DDNAMEØ1.
  GO TO    EXECØØ.
```


Back-up and recovery

Implementing the right back-up and recovery solution is essential to any successful client/server storage management strategy. But if your organization is like most, the disparate combination of operating systems, applications, and file systems that house your vital data are difficult to manage and growing almost out of control.

By providing a selective recovery capability for the hotsite as well as your local site, ABARS greatly enhances file recovery on demand.

The ABARS monitoring facility provides a simple-to-use ISPF interface to monitor the status of all aggregate back-ups that run within the enterprise. Online capabilities link with back-up indexes to present all the back-up information.

Threshold monitoring will quickly identify any aggregate back-up that runs for too long or is backing up too much data. Other utilities will assist personnel with the task of evenly splitting aggregates into more-easily controlled subsets.

The workshop environment, comprising MVS/ESA guests under VM, provides hands-on experience using the Aggregate Backup and Recovery Support (ABARS) of the Hierarchical Storage Manager (DFSMShsm). The examples illustrate how to identify and back up a group of related datasets (an aggregate) at a back-up site and then recover them at a recovery site.

Centralized monitoring links the back-up history to existing inclusion and instruction datasets. The ability to issue aggregate verification and submission is also provided.

ABARS definitions are shown below:

1a A started task:

```
//*************************************************************
//*      DFHSM SECONDARY ADDRESS SPACE START PROCEDURE        *
//*************************************************************
//DFHSMABR PROC
//DFHSMABR EXEC PGM=ARCWCTL,REGION=7500K
//SYSUDUMP DD SYSOUT=A
//MSYSIN   DD DUMMY
//MSYSOUT  DD SYSOUT=A
```

1b Update HSM:

```
SETSYS   ABARSACTLOGTYPE(SYSOUT(H))
SETSYS   ABARSACTLOGMSGLVL(FULL)
SETSYS   ABARSBUFFERS(4)
SETSYS   ABARSDELETEACTIVITY(Y)
SETSYS   ABARSOPTIMIZE(3)
SETSYS   ABARSPROCNAME(HSMABARS)
SETSYS   ABARSTAPE(STACK)
SETSYS   ABARSUNITNAME(3490)
SETSYS   ABARSVOLCOUNT(NONE)
SETSYS   ARECOVERUNITNAME(3490)
SETSYS   ARECOVERML2UNIT(3490)

SETSYS   MAXABARSADDRESSSPACE(1)
```

1c RACF commands:

```
RDEFINE TAPEVOL HSMABR
RDEFINE FACILITY STGADMIN.ARC.ABACKUP
PERMIT STGADMIN.ARC.ABACKUP CLASS(FACILITY) ID(SYSØ1) ACCESS(ALTER)
PERMIT STGADMIN.ARC.ABACKUP CLASS(FACILITY) ID(EXPØ1) ACCESS(READ)
PERMIT STGADMIN.ARC.ABACKUP CLASS(FACILITY) ID(PROØ1) ACCESS(READ)
RDEFINE FACILITY STGADMIN.ARC.ARECOVER
PERMIT STGADMIN.ARC.ARECOVER CLASS(FACILITY) ID(SYS01) ACCESS(ALTER)
PERMIT STGADMIN.ARC.ARECOVER CLASS(FACILITY) ID(EXP01) ACCESS(READ)
PERMIT STGADMIN.ARC.ARECOVER CLASS(FACILITY) ID(PRO01) ACCESS(READ)
SETR CLASSACT(FACILITY)  REFRESH
```

2a Using ABARS to verify:

```
//*******************************************************************
//** AGGREGATE GROUP * TESTØØØ1 *                                   **
//** SELECTION DATA SETS in HLQ.ABARS.CNTL(TESTØØØ1)                **
//*******************************************************************
//** ABACKUP THE FILES VERIFY                                       **
//*******************************************************************
//DELETE   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DELETE 'HLQ.ABARS.TESTØØØ1.OUTPUT' PURGE
 DELETE 'HLQ.ABARS.TESTØØØ1.LIST'   PURGE
 IF MAXCC <= 8 THEN SET MAXCC=Ø
/*
//ABARS    EXEC PGM=IKJEFTØ1
//*-------------------------
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 HSEND WAIT ABACKUP TESTØØØ1                        +
            VERIFY                                  +
            OPTIMIZE(3)                             +
            FODS('HLQ.ABARS.TESTØØØ1.OUTPUT')
 HSEND WAIT LIST AGGREGATE(TESTØØØ1) ODS('HLQ.ABARS.TESTØØØ1.LIST')
/*
//
            EXECUTE                                 +
```

2b Saving:

```
//*******************************************************************
//** AGGREGATE GROUP * TESTØØØ1 *                                   **
//*******************************************************************
//** ABACKUP  EXECUTE (AFter VERIFY)                                **
//*******************************************************************
//DELETE   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DELETE 'HLQ.ABARS.TESTØØØ1.OUTPUT' PURGE
```

```
 DELETE 'HLQ.ABARS.TEST0001.LIST'   PURGE
 IF MAXCC <= 8 THEN SET MAXCC=0
/*
//ABARS    EXEC PGM=IKJEFT01
//*-------------------------
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 HSEND WAIT ABACKUP TEST0001                          +
            EXECUTE                                   +
            OPTIMIZE(3)                               +
            FODS('HLQ.ABARS.TEST0001.OUTPUT')
 HSEND WAIT LIST AGGREGATE(TEST0001) ODS('HLQ.ABARS.TEST0001.LIST')
/*
//
```

3 Restore:

```
//*******************************************************************
//** AGGREGATE GROUP * TEST0001 *                                   **
//** SELECTION DATASETS IN HLQ.ABARS.CNTL(TEST0001)                 **
//*******************************************************************
//** ARECOVER - PREPARE - TO PASS FIRST IF AN EXTERNAL SITE -       **
//**                    -                        SEE DSN AND VOLSER **
//*******************************************************************
//ABARS    EXEC PGM=IKJEFT01
//*-------------------------
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 HSEND WAIT ARECOVER DSNAME(HSM.ABARS.LOCAL.TEST0001.C.C01V0001) +
                     PREPARE                       +
                     VOLUMES(LN0921)               +
                     UNIT(3490)
/*
//
```

Claude Dunand
Systems Programmer (France)

Migrated dataset reporting utility

DFHSM is widely used in mainframe shops for DASD space management, and unused datasets are migrated to cartridge tapes according to the DASD space management policy.

Information related to the datasets migrated is maintained in MCDS datasets by DFHSM. This utility reports the datasets that were migrated before a specified date. It could be useful to identify datasets that were migrated a long time ago and may not be required any more. Based on the report, steps could be taken to remove such obsolete datasets and thereby some cartridge tapes could be freed up for reuse.

Input to the utility is Type M (Migrated dataset information) records obtained from the MCDS dataset using the DCOLLECT utility. The JCL is shown below:

```
//STEP1      EXEC  PGM=IDCAMS
//SYSPRINT   DD    SYSOUT=*
//DCOUT      DD    DSN=DATASET.TYPEM.OUTPUT,DISP=OLD
//MCDS       DD    DSN=DATASET.MCDS,DISP=SHR
//SYSIN      DD    *
          DCOLLECT -
                OUTFILE(DCOUT) -
                MIGRATEDATA -

/* END OF DCOLLECT COMMAND
```

UTILITY CODE

```
/*------------------------REXX---------------------------------------*/
/*    DFHSM - MIGRATED DATASET REPORTING UTILITY                     */
/*                                                                   */
/*    THIS UTILITY REPORTS THE DATASET THAT WERE MIGRATED BEFORE A   */
/*    SPECIFIED DATE. IT COULD BE USEFUL TO IDENTIFY DATASETS THAT   */
/*    WERE MIGRATED LONG AGO AND MAY NOT BE REQUIRED ANY MORE.       */
/*    BASED ON THE REPORT, STEPS MAY BE TAKEN TO REMOVE SUCH         */
/*    OBSOLETE DATASETS.                                             */
/*                                                                   */
/*    INPUT: 1.DATASET.TYPEM.OUTPUT (DSORG=PS,RECFM=VB,LRECL=32756) */
/*             TYPE M (MIGRATED DATASET INFORMATION) RECORD DCOLLECT*/
/*             OUTPUT                                                */
/*           2.DATECHECK                                             */
/*             SPECIFY THE DATACHECK, BASED ON THIS, THE UTILITY     */
/*             FIND A DATASET THAT GOT MIGRATED BEFORE THIS DATE.    */
/*             FORMAT OF DATECHECK IS YYYYDDD.                       */
/*                                                                   */
/*    OUTPUT:  DATASET.REPORT (DSORG=PS,RECFM=FB,LRECL=256)          */
/*             REPORT OF DATASET MIGRATION BEFORE THE SPECIFIED DATE*/
/*             WITH DETAILS LIKE MIGRATION DATE, EXPIRY DATE,        */
/*             ML2/ML1, MANAGEMENT CLASS, DATASET SIZE.              */
/*                                                                   */
```

```
/*********************************************************************/
/*********************************************************************/
/* SPECIFY THE DATACHECK.                                            */
/*********************************************************************/
 DATECHECK=YYYYDDD
/*********************************************************************/
/* OPEN THE INPUT FILE CONTAINING DCOLLECT M RECORDS                 */
/*********************************************************************/
 MIGDATE = 0
 INNAME = "DATASET.TYPEM.OUTPUT"
 "ALLOC DD(INFILE) DA('"INNAME"')"
 IF RC¬= 0 THEN DO
    SAY  'ALLOCATION OF ('INNAME') FAILED'
    EXIT 8
    END
 "EXECIO 5000 DISKR INFILE (STEM RECORD."
/*********************************************************************/
/* OPEN THE FILE WHERE REPORT IS TO BE WRITTEN                       */
/*********************************************************************/
 "ALLOC DSNAME('DATASET.REPORT') DDNAME(OUTFILE)"
 IF RC¬= 0 THEN DO
    SAY  'ALLOCATION OF ('OUTNAME') FAILED'
    EXIT 12
    END
 OUT_LINE = "            MIGRATED DATASET REPORT           "
 PUSH OUT_LINE
 "EXECIO 1 DISKW OUTFILE"
 OUT_LINE = " "
 PUSH OUT_LINE
 "EXECIO 1 DISKW OUTFILE"
   CALL HEADER
/*********************************************************************/
/* PROCESS RECORD UNTIL END-OF-FILE                                  */
/*********************************************************************/
 DO WHILE RECORD.0 > 0
    DO J = 1 TO RECORD.0
            CALL EXTRACT
            CALL ML1CHECK
            PARSE VAR EXDATE  1 MIGDATE 8 .
            PARSE VAR EXLRFDT 1 REFDATE 8 .
            PARSE VAR EXEXPDT 1 EXPDATE 8 .
            IF DATECHECK > MIGDATE
            THEN CALL RECORD
            ELSE ITERATE
      END
      "EXECIO 5000 DISKR INFILE (STEM RECORD."
 END
 CALL RECORD
 "EXECIO 0 DISKR INFILE (FINIS"
 "EXECIO 0 DISKW OUTFILE (FINIS"
```

```
 "FREE DDNAME(INFILE)"
 "FREE DDNAME(OUTFILE)"
 EXIT
/*************************************************************/
/*         R O U T I N E S                                   */
/*************************************************************/
/*************************************************************/
/* PUT OUT A HEADER LINE                                     */
/*************************************************************/
 HEADER:
   OUT_LINE = LEFT("-",45,"-"),
      RIGHT("-",20,"-"),
      RIGHT("-",24,"-"),
      RIGHT("-",17,"-"),
      RIGHT("-",15,"-"),
      RIGHT("-",11,"-"),
      RIGHT("-",11,"-")
   PUSH OUT_LINE
   "EXECIO 1 DISKW OUTFILE"
   OUT_LINE = LEFT("  DATASET NAME  ",45),
      LEFT("  MANAGEMENT CLASS  ",20),
      LEFT("  LAST REFERENCED DATE  ",24),
      LEFT("  MIGRATED DATE  ",17),
      LEFT("  EXPIRY DATE  ",15),
      LEFT("  SIZE(KB)  ",11),
      LEFT("  ML1/ML2  ",11)
   PUSH OUT_LINE
   "EXECIO 1 DISKW OUTFILE"
   OUT_LINE = LEFT("-",45,"-"),
      RIGHT("-",20,"-"),
      RIGHT("-",24,"-"),
      RIGHT("-",17,"-"),
      RIGHT("-",15,"-"),
      RIGHT("-",11,"-"),
      RIGHT("-",11,"-")
   PUSH OUT_LINE
   "EXECIO 1 DISKW OUTFILE"
 RETURN
/*************************************************************/
/* WRITE A RECORD                                            */
/*************************************************************/
 RECORD:
   OUT_LINE = LEFT(EXDSNAM,45),
      LEFT(EXMGTCL,20),
      RIGHT(REFDATE,24),
      RIGHT(MIGDATE,17),
      RIGHT(EXPDATE,15),
      RIGHT(EXDSIZE,11),
      RIGHT(MIGLEVEL,11)
   PUSH OUT_LINE
```

```
   "EXECIO 1 DISKW OUTFILE"
 RETURN
/*************************************************************/
/* PARSE TYPE M RECOTDS                                      */
/*************************************************************/
 EXTRACT:
   PARSE VAR RECORD.J 1 . 25 EXDSNAM 69 . 70 EXDEVCL 71 . ,
      73 EXDSIZE 77 . 81 EXDATE 85 . 151 EXMGTCL 181 . ,
      205 EXEXPDT 209 . ,
      217 EXLRFDT 221 .
/* CONVERSION         */
   EXEXPDT = C2X(EXEXPDT)
   EXDATE = C2X(EXDATE)
   EXDSIZE = C2D(EXDSIZE)
   EXLRFDT = C2X(EXLRFDT)
 RETURN
/*************************************************************/
/* ML1 CHECK                                                 */
/*************************************************************/
ML1CHECK:
IF EXDEVCL=D
   THEN MIGLEVEL = "ML1"
ELSE MIGLEVEL = "ML2"
RETURN
```

Arun Kumaar R
MVS Systems Programmer (India)

Multi-threading in COBOL

Unlike Java and C++, until recently you could not run your COBOL programs in more than one thread. If you tried to invoke the COBOL program from an application server in the second thread, it used to crash with a run-time error, 'COBOL found in multiple threads'. This heavily limited the possibility of using COBOL subroutines as building blocks for Web applications.

With Enterprise COBOL, IBM provides toleration-level support of POSIX threads and asynchronous signals. This article aims to explain the level of multi-threading support provided by COBOL and the associated features.

THREAD COMPILER OPTION

The THREAD compiler option enables a COBOL program to be multi-threaded (ie it can be called in more than one thread in a single process). This allows the program to run in multiple threads under batch, TSO, IMS, or Unix environments.

To support multi-threading, the program must be thread-safe, allowing multiple copies of it to run in the same run-unit. With the THREAD option, the storage and control blocks get appropriately allocated at invocation, rather than per program. Also, additional serialization logic is generated automatically (which in turn can degrade performance).

Programs compiled with compilers pre-dating Enterprise COBOL are treated as compiled with NOTHREAD.

Note that a program that has been compiled with the THREAD option can:

- Run in CICS/IMS environments.
- Run in AMODE 24.
- Be used in a non-threaded application.
- Call programs that are not enabled for multi-threading (provided the application doesn't have multiple threads).

The following are the prerequisites to be met for running COBOL programs in a multi-threaded environment:

- All the programs within the run-unit must be compiled with the THREAD option.
- Programs must be compiled and link edited with the RENT option.
- Programs must be RECURSIVE.

RECURSIVE PROGRAM

A recursive call is where a called program can directly or indirectly execute its caller. For example, program X calls program

Y, program Y calls program Z, and program Z then calls program X (the original caller).

To make a recursive call, you must code the RECURSIVE clause (IBM extension) on the PROGRAM-ID paragraph of the recursively called program. If the optional RECURSIVE clause is specified, the program can be re-entered recursively while a previous invocation is still active.

LOCAL STORAGE AND WORKING STORAGE

A multi-threaded program needs to be recursive and the persistence of the data for each call depends on whether it is in local storage or working storage.

Multiple threads that run simultaneously share a single copy of the WORKING-STORAGE data (statically allocated and initialized on first entry to a program, and available in the last-used state for the recursive invocations).

A separate copy of LOCAL-STORAGE data is allocated and made available for each call of a program (or invocation of a method); it gets released on returning from the program. If the VALUE clause is specified, the data is re-initialized to be the same for every invocation.

THREAD MANAGEMENT

Currently COBOL doesn't manage the threads; rather, it expects the application server or the calling program (in Java, C/C++, PL/I) to manage them. The threaded application must run within a single Language Environment enclave (created using the CEEPIPI routine).

FILE ACCESS IN MULTI-THREADED PROGRAMS

Multi-threaded COBOL programs can have file operations on QSAM, VSAM, and sequential files. Automatic serialization happens using the implicit lock on the file definition, during the

execution of OPEN, CLOSE, READ, WRITE, REWRITE, START, and DELETE statements.

All threads of execution share the storage associated with the file definition (FD). It is important to note that the FD records and the records with the SAME RECORD AREA clause have the data available in the last-used state.

Serialization is not automatic between uses of these statements – and to avoid coding your own serialization logic (using POSIX APIs), IBM suggests the following:

- For input, the recommended usage pattern is OPEN, READ, process the record, CLOSE.
- For output, the recommended usage pattern is OPEN, construct the output, WRITE, CLOSE.
- Also, define the data items that are associated with the file (such as file-status data items and key arguments) in the LOCAL-STORAGE SECTION.

Sharing in a multi-threaded environment:

- For programs compiled with the THREAD option, the special registers (like ADDRESS-OF, RETURN-CODE, SORT-CONTROL, TALLY, XML-CODE, and XML-EVENT) are allocated (and reset to the initial value) on a per-invocation basis.
- All programs and all threads in an application share a single copy of UPSI switches. If you modify switches in a threaded application, you must code appropriate serialization logic.
- Indexes are normally allocated in static memory associated with the program and are in the last-used state when a program is re-entered. However, if compiled with the THREAD option, the indexes are allocated on a per-invocation basis and must be SET on every entry.
- If you compile your program with the THREAD compiler option, data that is defined in the LINKAGE SECTION is not accessible on subsequent invocations of the program. The

address of the record in the Linkage section must be re-established for that execution instance.

Ending a program in a multi-threaded environment:

- When you use GOBACK from the first program in a thread, the thread is terminated. If that thread is the initial thread in an enclave, the entire enclave is terminated.
- With the EXIT program, the thread is not terminated unless the program is the first (oldest) one in the thread.
- With STOP RUN, the entire LE enclave is terminated, including all threads executing within the enclave.

Other factors to consider when multi-thread enabling COBOL programs:

- In a multi-threaded environment, a program cannot CANCEL a program that is active on any thread. If you try to cancel an active program, a severity-3 Language Environment condition occurs.
- A program executing on multiple threads can execute the same or different XML statements simultaneously.
- In a threaded application, the COBOL program can be interrupted by asynchronous signals, which the program should be able to tolerate. Alternatively, using C/C++ functions, the interrupts can be disabled by setting the signal mask appropriately.
- The RANDOM function can be used in threaded programs. For an initial seed, a single sequence of pseudo-random numbers is returned, regardless of the thread that is running when RANDOM is invoked.

Constraints related to multi-threaded COBOL programs:

- You cannot run multi-threaded applications in the CICS environment (though you can run programs compiled with the THREAD option).
- If the COBOL program has been compiled with the THREAD

option with AMODE 24, then it can only be part of a non-threaded application.

- Nested programs are not supported for programs compiled with the THREAD option.
- Segmentation is not supported for programs compiled with the THREAD option.
- You cannot use the RERUN clause in I/O control in programs compiled with the THREAD option
- Priority-numbers are not valid for programs compiled with the THREAD option.
- You cannot specify ALTER or altered GO TO statements in programs compiled with the THREAD option.
- The MERGE statement is not supported for programs compiled with the THREAD compiler option.
- The SORT statement is not supported for programs compiled with the THREAD option.
- Do not use the STOP literal statement in programs compiled with the THREAD compiler option.
- Debugging sections are not permitted in programs compiled with the THREAD compiler option.
- Environments created by IGZERRE or ILBOSTP0 or by using the RTEREUS run-time options do not support the THREAD option.
- Do not use IGZBRDGE, the macro for converting static calls to dynamic calls, because it is not supported.
- Do not use the modules IGZETUN (for storage tuning) or IGZEOPT (for run-time options) because these CSECTs are ignored.

CONCLUSION

Currently what IBM provides with Enterprise COBOL is basic

support for invoking COBOL programs in a multi-threaded environment. As can be seen, there are quite a few restrictions and behavioural differences when the THREAD option is used. Enabling existing COBOL programs to be multi-threaded can range from just recompiling (best) to modifying program logic that requires thorough testing. It is expected that multi-threading would be handled more naturally in COBOL in future versions.

Sasirekha Cota
Tata Consultancy Services (India)

REXX tool for viewing/copying a VSAM KSDS

Small mainframe shops, primarily used for internal training, and small to medium businesses often find it difficult to bear the extra overhead of tools like File-Aid. While File-Aid is a multi-utility tool, one of its major features is the handling of VSAM files. In the absence of File-Aid, the only other way to view/copy a VSAM KSDS is by using IDCAMS.

This tool presents an alternative way to view/copy a VSAM KSDS. It's a simple REXX routine that can directly display records of a VSAM KSDS on the terminal (record length truncated at 80 bytes) or copy the entire record to a sequential file for easy viewing. For convenience in installation, the entire logic has been incorporated into a single routine. The user just has to copy the routine into a library allocated to STSPROC/SYSEXEC and run the VSAMUTIL routine in TSO. For the same reason, no ISPF commands/utilities/panels have been used. The menu-driven interface is user-friendly and intuitive.

Notes:

1 This tool is intended to be run online in TSO. If the user intends to run this routine in batch using the TSO Terminal Monitor program (IKJEFT01), some customization of the code is required.

2 The sequential file created for the VSAM copy and the work file created (in case the user wants to see the records on the terminal) follow a specific naming convention. The name will be displayed on the terminal before the creation of the files and the user will have the option of terminating the routine then.

Reference: *OS/390 V2R10.0 TSO/E REXX*.

VSAMUTIL

```
/**** rexx *****/
/*********************************************************************/
/**** This is the main driver program which throws the opening screen */
/*********************************************************************/
address tso "tsoclear"
say '          Welcome to the VSAM Utility '
say '          =========================== '
do forever
   say '  '
   say ' 1. VSAM File copy to a seq file'
   say ' 2. VSAM File copy on to the screen'
   say ' 3. Exit.'
   say ' '
   say 'Enter your choice:'
   pull choice
   choice = strip(choice)
   select
      when choice = '3' then
         call exit_para
      when choice = '1' then
         call filecopy_para
      when choice = '2' then
         call scrncopy_para
      otherwise
         call wrgch_para
   end
end
/******* This para is called when the user enters a wrong choice ****/
wrgch_para:
ADDRESS TSO "TSOCLEAR"
say ' '
say 'Invalid choice..Please re-enter. Enter option 3 to exit.'
return
/**** This para is called when the user wants to copy the VSAM recs **/
/**   on to a file  *****/
filecopy_para:
```

```
say ' '
say 'This option will copy the VSAM file on to a seq file......'
say 'It will create/replace the seq file - userid.vsamutil.seq1'
say ' '
say 'Do you want to continue with this option (Y/N):'
pull filecopy_opt
if filecopy_opt = 'Y' then nop
else
   do
     if filecopy_opt = 'N' then
        do
          ADDRESS TSO "TSOCLEAR"
          say 'Returning to the main menu...'
          return
        end
     else
        do
          ADDRESS TSO "TSOCLEAR"
          say 'Wrong Option entered..'
          say 'Returning to the main menu...'
          return
        end
   end
say 'Enter the VSAM FILE Cluster(within quotes):'
pull vsamfle1
x1 = outtrap(vsamf.)
ADDRESS TSO "LISTCAT ENT("vsamfle1")"
x1 = outtrap(off)
flag_vsam1 = Ø
do y1 = 1 to vsamf.Ø
   vsamrec1 = strip(vsamf.y1)
   Z1 = WORD(vsamrec1,1)
   if z1 = 'CLUSTER' then
      flag_vsam1 = 1
   else nop
end
if flag_vsam1 = Ø then
   do
     say 'Invalid VSAM dataset entered..'
     say 'Returning to the main menu..'
     ADDRESS TSO "TSOCLEAR"
     return
   end
say 'Creating the seq file....'
vsamfile = vsamfle1
call filestat_para vsamfile
lrecl1 = seqlrecl
trace o
seq_file = userid() || '.vsamutil.seq1'
seq_file_sysdsn = "'" || seq_file || "'"
```

```
if SYSDSN(seq_file_sysdsn) = 'OK'
then
   ADDRESS TSO "DELETE '"seq_file"'"
else nop
ADDRESS TSO "ALLOC DATASET('"seq_file"') NEW CYLINDER SPACE(1Ø,1Ø)
             DSORG(PS) RECFM(F,B) LRECL("lrecl1")"
if rc = Ø
then
   say 'Seq file created successfully...'
else
   do
     say 'Error in seq file creation....'
     say 'Returning to the main menu....'
     return
   end
say 'Allocating Input and Output files....'
ADDRESS TSO "ALLOC DA("vsamfle1") FI(IDSN) SHR"
if rc <> Ø
then
   do
     say 'Input VSAM File allocation unsuccessful...'
     say 'Returning to the main menu..'
     return
   end
ADDRESS TSO "ALLOC DA('"seq_file"') FI(ODSN) SHR"
if rc <> Ø
then
   do
     say 'Output seq File allocation unsuccessful...'
     say 'Returning to the main menu..'
     return
   end
say 'File Allocation successful...'
say ' '
say 'Is there any specific count of records that needs to be copied..'
say 'Leave it as spaces if you want the whole file to be copied :'
pull copy_count1
say 'Copying the VSAM file...'
if copy_count1 = ' '
then
   do
     ADDRESS TSO "REPRO INFILE(IDSN) OUTFILE(ODSN)"
   end
else
   do
     option_repro = "COUNT("copy_count1")"
     ADDRESS TSO "REPRO INFILE(IDSN) OUTFILE(ODSN)" option_repro
   end
if rc = Ø
then
```

```
   do
     say 'Repro successful....'
     ADDRESS TSO "FREE FI(IDSN)"
     ADDRESS TSO "FREE FI(ODSN)"
     return
   end
else
   do
     say 'Repro not successful....'
     say 'Returning to the main menu...'
     return
   end
return
/*** This para is called when the user wants to see the VSAM recs **/
/*** on the screen *****/
scrncopy_para:
say ' '
say 'This option will display the VSAM recs on to the terminal..'
say 'This will create/replace a work file - userid.vsamutil.seq1.work..'
say 'It will truncate all the records to 80 bytes..'
say 'If you want to see the complete records use option 1..'
say ' '
say 'Do you want to continue with this option (Y/N):'
pull scrncopy_opt
if scrncopy_opt = 'Y' then nop
else
   do
     if scrncopy_opt = 'N' then
        do
          ADDRESS TSO "TSOCLEAR"
          say 'Returning to the main menu...'
          return
        end
     else
        do
          ADDRESS TSO "TSOCLEAR"
          say 'Wrong Option entered..'
          say 'Returning to the main menu...'
          return
        end
   end
say 'Enter the VSAM FILE Cluster(within quotes):'
pull vsamscrn
xS = outtrap(vsamfs.)
ADDRESS TSO "LISTCAT ENT("vsamscrn")"
xS = outtrap(off)
flag_vsams = 0
do ys = 1 to vsamfs.0
   vsasrec = strip(vsamfs.ys)
   Zs = WORD(vsasrec,1)
```

```
   if zs = 'CLUSTER' then
      flag_vsams = 1
   else nop
end
if flag_vsams = Ø then
   do
     say 'Invalid VSAM dataset entered..'
     say 'Returning to the main menu..'
     ADDRESS TSO "TSOCLEAR"
     return
   end
say 'Creating temporary work files...'
vsamfile = vsamscrn
call filestat_para vsamfile
lrecls = seqlrecl
seq_file_s = userid() || '.vsamutil.seq1.work'
seq_file_sysdsn_s = "'" || seq_file_s || "'"
if SYSDSN(seq_file_sysdsn_s) = 'OK'
then
   ADDRESS TSO "DELETE '"seq_file_s"'"
else nop
ADDRESS TSO "ALLOC DATASET('"seq_file_s"') NEW CYLINDER SPACE(1Ø,1Ø)
             DSORG(PS) RECFM(F,B) LRECL("lrecls")"
if rc = Ø
then
   say 'Work file created successfully...'
else
   do
     say 'Error in work file creation....'
     say 'Returning to the main menu....'
     return
   end
say 'Allocating Input and Output files....'
ADDRESS TSO "ALLOC DA("vsamscrn") FI(IDSN) SHR"
if rc <> Ø
then
   do
     say 'Input VSAM File allocation unsuccessful...'
     say 'Returning to the main menu..'
     return
   end
ADDRESS TSO "ALLOC DA('"seq_file_s"') FI(ODSN) SHR"
if rc <> Ø
then
   do
     say 'Output work File allocation unsuccessful...'
     say 'Returning to the main menu..'
     return
   end
say 'Work Files Allocation successful...'
```

```
say ' '
say 'Is there any specific count of records that needs to be displayed.'
say 'Leave it as spaces if you want the whole file to be displayed:'
pull copy_count_s
say 'Populating work files...'
if copy_count_s = ' '
then
   do
     ADDRESS TSO "REPRO INFILE(IDSN) OUTFILE(ODSN)"
   end
else
   do
     option_repro = "COUNT("copy_count_s")"
     ADDRESS TSO "REPRO INFILE(IDSN) OUTFILE(ODSN)" option_repro
   end
if rc = Ø
then
   do
     say 'Work File Population successful....'
     ADDRESS TSO "FREE FI(IDSN)"
     ADDRESS TSO "FREE FI(ODSN)"
   end
else
   do
     say 'Work File population not successful....'
     say 'Returning to the main menu...'
     return
   end
say 'Displaying the VSAM Data......'
say ' '
ADDRESS TSO "ALLOC DA('"seq_file_s"') FI(WKDD) SHR"
ADDRESS TSO "EXECIO * DISKR WKDD (STEM DISPL."
ADDRESS TSO "FREE FI(WKDD)"
do displcnt = 1 to displ.Ø
   final_displ_rec = substr(displ.displcnt,1,79)
   say final_displ_rec
end
say ' '
say 'Display Complete...'
return
/**** This para is called to find the LRECL of the VSAM file ***/
filestat_para:
arg vsamfile
x2 = outtrap(vsamf2.)
ADDRESS TSO "LISTCAT ALL ENT("vsamfile")"
x2 = outtrap(off)
do y2 = 1 to vsamf2.Ø
   vsamrec = STRIP(vsamf2.y2)
   vsamrec = TRANSLATE(vsamrec,' ','-')
   z2 = WORDPOS('MAXLRECL',vsamrec)
```

```
   if z2 = 0 then nop
   else
      do
        z2 = z2 + 1
        seqlrecl = WORD(vsamrec,z2)
        seqlrecl = strip(seqlrecl)
        return seqlrecl
      end
end
return
/******* This is the common exit para *****/
exit_para:
say ' Exiting out of the application...'
exit
```

Manas Biswal
Associate
Cognizant Technology Solutions (USA)

Monitoring USS performance from z/OS – an introduction: part 2

This month we conclude the article looking at USS performance and publish the code.

INTERPROCESS COMMUNICATION REPORT

As it is set by USS design, the processes on the same system can communicate using three different types of IPC resource – message queues, shared memory segment, and semaphores. The BPXPRM*xx* parameters corresponding to the defined maximum values as reported under the heading *PARM VALUE* for message id, semaphor id, shared memory id, and shared memory pages, are IPCMSGNIDS, IPCSEMNIDS, IPCSHMNIDS, and IPCSHMSPAGES. These parameters do affect the processor storage requirements and one should be careful when setting up the values. The setting of these parameters really depends on the applications, and a suggestion for setting up these values is: start with the defaults, look frequently at the reports, and change the values accordingly. Applications will fail if they do not have these resources. Remember that all these structures (queues, semaphores, and shared memory pages) need to be mapped in shared storage pages. Also note that when a process goes away, the IPC resource may stay until all processes connected to the IPC resource go away or until manually cleaned.

MEMORY MAP REPORT (OR CONTROLLING THE ESQA)

The BPXPRM*xx* parameters corresponding to the defined maximum values as reported under the heading *PARM VALUE* for memory map, storage pages, and shared storage pages, are MAXMMAPAREA and MAXSHAREPAGES. MAXSHAREPAGES is the maximum number of pages for shared storage used by the following USS services: fork(), when fork is using 'copy on write'

mode; ptrace(), debugger support (process trace); mmap(), memory mapped files; shmat(), shared memory attach; and dllload(), when it is loading a user-shared library program.

MAXMMAPAREA is the maximum number of pages used for a memory map. Having these values set too low can limit the number of shared storage pages used by Unix work; having them set too high could cause shortages of ESQA. Normally these are Fixed ESQA pages, which implies that real storage as well as virtual storage is allocated. As a general rule, you have to allow enough shared storage to improve USS performance; at the same time you have to be sure that this does not negatively impact your system. On z/OS, shared memory is as efficient as any other type of memory access. When you use it, you need to be aware of its impact on the Extended System Queue Area (ESQA) storage requirements.

ESQA storage is in the common area and page fixed, which causes it to consume real memory. A number of z/OS Unix System Services use base z/OS functions that consume ESQA storage. Installations having constraints on virtual storage or main memory can control the amount of ESQA storage consumed. Ensuring the appropriate size of ESQA and extended Common Service Area (CSA) storage is critical to the long-term operation of the system.

SHARED LIBRARY REGION AND QUEUED SIGNALS REPORT

In the latest release of z/OS, the recording for SMF 74 subtype 3 has been enhanced to provide additional information about limits for shared library regions and queued signals. The enhancements allow you to estimate the size of the shared library region (similar to LPA). If this region is too small, programs are added to each address space's private area instead of being loaded only once into common storage. Queued signals represent pending completions of asynchronous I/O requests. The new option that USS introduces protects the kernel from overcommitting storage. Once the limit is reached, additional asynchronous I/Os are rejected.

CONCLUSION

It should be noted that this performance report writer is not completely comprehensive – nevertheless it is an open-ended program that allows a user to drill down where necessary by modifying and customizing reports generated so as to meet an installation's needs or requirements.

On the other hand, it can be used as the starting point to build a set of USS performance yardsticks.

USSDRIVE JOB

```
//DEL       EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=X
//SYSIN     DD *
    DELETE hql.U92.OUT
    DELETE hql.SMFCOPY.OUT
    DELETE hql.U9211.TEST
    DELETE hql.U304.TEST
    DELETE hql.U74.TEST
    DELETE hql.MAT30A.OUT
    DELETE hql.MAT92A.OUT
    SET MAXCC=0
/*
//SMFEXTR  EXEC PGM=IFASMFDP,REGION=5M
//INDA1 DD DSN=your.smf.dataset,DISP=SHR
//OUTDA DD DSN=hql.SMFCOPY.OUT,DISP=(NEW,CATLG),
//         UNIT=SYSDA,SPACE=(CYL,(50,20),RLSE),
//         DCB=(syshlq.SMFDUMPx)
//SYSPRINT DD SYSOUT=X
//SYSIN DD *
       DATE(2003251,2003252)
       INDD(INDA1,OPTIONS(DUMP))
       OUTDD(OUTDA,TYPE(30,74,92))
/*
//COPYSMF  EXEC PGM=ICETOOL,REGION=0M
//SORTWK01 DD   DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(TRK,(200,200))
//SORTWK02 DD   DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(TRK,(200,200))
//SORTWK03 DD   DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(TRK,(200,200))
//TOOLMSG  DD SYSOUT=*
//DFSMSG   DD SYSOUT=*
//RAWSMF   DD DSN=hql.SMFCOPY.OUT,DISP=SHR
//SMF304   DD DSN=hql.U304.TEST,
//         SPACE=(CYL,(1)),UNIT=SYSDA,DISP=(NEW,KEEP),
//         DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760)
//SMF74    DD DSN=hql.U74.TEST,
//         SPACE=(CYL,(1)),UNIT=SYSDA,DISP=(NEW,KEEP),
```

```
//          DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760)
//SMF9211   DD DSN=hql.U9211.TEST,
//          SPACE=(CYL,(15)),UNIT=SYSDA,DISP=(NEW,KEEP),
//          DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760)
//TOOLIN    DD *
  COPY FROM(RAWSMF) TO(SMF304)  USING(SMF3)
  COPY FROM(RAWSMF) TO(SMF74)   USING(SMF7)
  COPY FROM(RAWSMF) TO(SMF9211) USING(SMF9)
//SMF3CNTL DD *
* Get SMF 30.4 with valid USS data
  OPTION SPANINC=RC4,VLSHRT
  INCLUDE COND=(6,1,BI,EQ,30,AND,23,2,BI,EQ,4,AND,
                129,4,BI,NE,0,AND,
                135,4,BI,NE,0)
  SORT FIELDS=(11,4,CH,A,7,4,CH,A),EQUALS
//SMF7CNTL DD *
* Get SMF 74.3 with valid Kernel data;
* Also get SMF 74.6 with valid HFS data.
* Sort by subtype and date/time
  OPTION SPANINC=RC4,VLSHRT
  INCLUDE COND=(6,1,BI,EQ,74,AND,23,2,BI,EQ,3,
                AND,41,2,BI,GT,0,AND,43,2,BI,GT,0,
                OR,
               (6,1,BI,EQ,74,AND,23,2,BI,EQ,6,
                AND,51,2,BI,GT,0,AND,59,2,BI,GT,0))
  SORT FIELDS=(6,1,BI,A,23,2,BI,A,11,4,CH,A,7,4,CH,A)
//SMF9CNTL DD *
* Get SMF 92.11 with valid Close file data
  OPTION SPANINC=RC4,VLSHRT
  INCLUDE COND=(6,1,BI,EQ,92,AND,23,2,BI,EQ,11)
  SORT FIELDS=(85,4,CH,A,7,4,CH,A,11,4,CH,A),EQUALS
/*
//REXX13    EXEC PGM=IKJEFT01,REGION=0M
//SYSEXEC   DD DISP=SHR,DSN=your.rexx.lib
//SMF30     DD DISP=SHR,DSN=hql.U304.TEST
//SMF74     DD DISP=SHR,DSN=hql.U74.TEST
//SMF92     DD DISP=SHR,DSN=hql.U9211.TEST
//SYSTSPRT DD SYSOUT=*
//SYSTSIN   DD *
prof nopref
%USSASID
%USSFILE
%RMFUSS
/*
//MTOOL     EXEC PGM=ICETOOL
//TOOLMSG   DD SYSOUT=*
//DFSMSG    DD SYSOUT=*
//SMF9211   DD DSN=hql.R92FI.REP,DISP=(SHR,DELETE)
//SMF92     DD DSN=hql.USSFILE.REP,DISP=MOD
//MAT9211   DD DSN=hql.MAT92.OUT,DISP=(SHR,DELETE)
```

```
//MAT92    DD DSN=hql.MAT92A.OUT,
//         DISP=(NEW,CATLG),LIKE=hql.MAT92.OUT
//MAT3Ø4   DD DSN=hql.MAT3Ø.OUT,DISP=(SHR,DELETE)
//MAT3Ø    DD DSN=hql.MAT3ØA.OUT,
//         DISP=(NEW,CATLG),LIKE=hql.MAT3Ø.OUT
//TOOLIN   DD *
  SORT FROM(SMF9211) TO(SMF92) USING(SMFT)
  SORT FROM(MAT9211) TO(MAT92) USING(SMF9)
  SORT FROM(MAT3Ø4)  TO(MAT3Ø) USING(SMF3)
//SMFTCNTL DD *
  OPTION TRUNC=RCØ
  SORT FIELDS=(242,5,CH,A,13,8,CH,A)
//SMF9CNTL DD *
  SORT FIELDS=(1,4Ø,CH,A)
//SMF3CNTL DD *
  SORT FIELDS=(1,4Ø,CH,A)
/*
//REXX4    EXEC PGM=IKJEFTØ1,REGION=ØM
//SYSEXEC  DD DISP=SHR,DSN=your.rexx.lib
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
prof nopref
%MATCH
/*
```

USSASID EXEC

```
/* REXX EXEC (USSASID) to read and format SMF 3Ø.4 records        */
/*-----------------------------------------------------------------*/
/* Part 1: Handle file allocation & dataset existence and         */
/*         print report header and labels                         */
/*-----------------------------------------------------------------*/
Address TSO
  userid=SYSVAR(SYSUID)
  r3Øio  =userid||'.ioact.rep'          /* I/O Activity report     */
  r3Øss  =userid||'.summs.rep'          /* Summary Report          */
  r3Øcp  =userid||'.cpu.rep'            /* Processor usage Report  */
  r3Øom  =userid||'.proces.rep'         /* USS Process Report      */
  r3Øpr  =userid||'.perfom.rep'         /* Performance Report      */
  r3Øpp  =userid||'.storage.rep'        /* Storage & Paging Report */
  r3Øuu  =userid||'.mat3Ø.out'          /* Match/Merge file        */
  x = MSG('OFF')
  if SYSDSN(r3Øio) = 'OK'         /* check report dsn validity     */
  Then "DELETE "r3Øio" PURGE"
  if SYSDSN(r3Øss) = 'OK'         /* check report dsn validity     */
  Then "DELETE "r3Øss" PURGE"
  if SYSDSN(r3Øcp) = 'OK'         /* check report dsn validity     */
  Then "DELETE "r3Øcp" PURGE"
  if SYSDSN(r3Øom) = 'OK'         /* check report dsn validity     */
```

```
  Then "DELETE "r30om" PURGE"
  if SYSDSN(r30pr) = 'OK'          /* check report dsn validity     */
  Then "DELETE "r30pr" PURGE"
  if SYSDSN(r30pp) = 'OK'          /* check report dsn validity     */
  Then "DELETE "r30pp" PURGE"
  if SYSDSN(r30uu) = 'OK'          /* check report dsn validity     */
  Then "DELETE "r30uu" PURGE"
  "ALLOC FILE(s30uu) DA("r30uu")",
 " UNIT(SYSALLDA) NEW TRACKS SPACE(9,9) CATALOG",
 " REUSE LRECL(70) RECFM(F B) BLKSIZE(27930)"
  "ALLOC FILE(s30io) DA("r30io")",
 " UNIT(SYSALLDA) NEW TRACKS SPACE(9,9) CATALOG",
 " REUSE LRECL(195) RECFM(F B) BLKSIZE(27885)"
 Io.1  =left(' ',8,' '),
        ||'I/O Activity Report for USS jobs/tasks'||left(' ',15,' ')
 Io.2  ='  '
 Io.3  =left(' ',8,' ')||'Report produced on',
        ||left(' ',1,' ')||left(date(),11),
        ||left(' ',1,' ')||left('at ',3,' ')||left(time(),10)
 Io.4  =left(' ',133,' ')||left('-',17,'-')||left(' ',2,' '),
        ||left('Enclave DASD I/O timing',25)||left('-',14,'-')
 Io.5  =left('',93,' ')||left('-- Delays --',14)||left(' ',8,' '),
        ||left('Blocks',9)||left('Connect',7)||left(' ',2, ' '),
        ||left('Dependent enclave',31)||left('Independent enclave',19)
 Io.6   =left('OBS',4)||left('Date',12)||left('Time',9),
        ||left('User id',9)||left('Job id',8),
        ||left('Job name',9)||left('Step name',11),
        ||left('Process',8)||left('Program',18),
        ||left('ds enqueue',13)||left('allocation',13),
        ||left('transfer',11)||left('time',8),
        ||left('connect',10)||left('disconn.',10)||left('pending',11),
        ||left('connect',10)||left('disconn.',10)||left('pending',8)
  Io.7  =left(' ',4,' ')||left('-',187,'-')
   "EXECIO * DISKW s30io (STEM Io.)"
  "ALLOC FILE(s30ss) DA("r30ss")",
 " UNIT(SYSALLDA) NEW TRACKS SPACE(9,9) CATALOG",
 " REUSE LRECL(215) RECFM(F B) BLKSIZE(27950)"
Sum.1  =left(' ',8,' ')||'Summary Report for USS jobs/tasks',
        ||left(' ',15,' ')
Sum.2  ='  '
Sum.3  =left(' ',8,' ')||'Report produced on',
        ||left(' ',1,' ')||left(date(),11),
        ||left(' ',1,' ')||left('at ',3,' ')||left(time(),10)
Sum.4  =left(' ',93,' ')||left('-- Timings --',13),
        ||left(' ',7,' ')||left('-- Time --',12)||left(' ',11,' '),
        ||left('Service',10)||left("Blocks X'ferred",16),
        ||left('USS I/O',11)||left('USS',7)||left('Connect',11),
        ||left('----PAGING COUNTS--',20)
Sum.5   =left('OBS',4,)||left('Date',12)||left('Time',9),
        ||left('User id',9)||left('Job id',8),
```

```
         ||left('Job name',9)||left('Step name',11),
         ||left('Process',8)||left('Program',19),
         ||left('RC',4)||left('TCB',5)||left('SRB',4)||left('Total',6),
         ||left('Execution',13)||left('Elapsed',17)||left('units',10),
         ||left('I/O',8)||left('Excp',6)||left('blocks',9),
         ||left('calls',12)||left('time',8),
        ||left('PAGE    SWAP     VIO',20)
Sum.6   =left(' ',4,' ')||left('-',207,'-')
   "EXECIO * DISKW s30ss (STEM Sum.)"
  "ALLOC FILE(s30cp) DA("r30cp")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(9,9) CATALOG",
  "REUSE LRECL(150) RECFM(F B) BLKSIZE(27900)"
Cpu.1  =left(' ',8,' '),
        ||'Processor Report for USS jobs/tasks'||left(' ',15,' ')
Cpu.2  ='  '
Cpu.3  =left(' ',8,' ')||'Report produced on',
        ||left(' ',1,' ')||left(date(),11),
        ||left(' ',1,' ')||left('at ',3,' ')||left(time(),10),
        ||left(' ',33,' ')||left('-',8,'-')||left(' ',11,' '),
        ||left('  PROCESSOR TIMINGS  ',21)||left(' ',10,' '),
        ||left('-',8,'-')
Cpu.4  =left(' ',110,' ')||left('Hiperspace',12),
        ||left('Preemptable',13)||left('Enclaves',8)
Cpu.5   =left('OBS',4,)||left('Date',12)||left('Time',9),
        ||left('User id',9)||left('Job id',8),
        ||left('Job name',9)||left('Step name',11),
        ||left('Process',8)||left('Program',12),
        ||left('Total',8)||left('TCB',6)||left('SRB',6),
        ||left('I/O',6)||left('RCT',5),
        ||left("X'fer",6)||left('Crypto',8)||left('SRB',4),
        ||left('indpt.',7)||left('dep.',6)||left('enq.',4)
Cpu.6   =left(' ',4,' ')||left('-',144,'-')
   "EXECIO * DISKW s30cp (STEM Cpu.)"
  "ALLOC FILE(s30omvs) DA("r30om")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(9,9) CATALOG",
  "REUSE LRECL(280) RECFM(F B) BLKSIZE(27720)"
Omv.1  =left(' ',8,' '),
        ||'Process I/O Report for USS jobs/tasks'||left(' ',15,' ')
Omv.2  ='  '
Omv.3  =left(' ',8,' ')||'Report produced on',
        ||left(' ',1,' ')||left(date(),11),
        ||left(' ',1,' ')||left('at ',3,' ')||left(time(),10)
Omv.4  =left(' ',128,' ')||left('Standard file I/O',21),
        ||left('Pipe file I/O',19)||left('Special file I/O',20),
        ||left('Remote socket I/O',19)||left('Path look calls',20),
        ||left('Path gen. calls',19)||left('Message queues byte',23),
        ||left("sync()",12)
Omv.5   =left('OBS',4,)||left('Date',12)||left('Time',9),
        ||left('User id',9)||left('Job id',8),
        ||left('Job name',9)||left('Step name',11),
```

```
        ||left('Program',9)||left('Proc.ID',9),
        ||left('Parent ID',12)||left('User ID',9),
        ||left('Syscalls',9)||left('CPU',6)||left('Dir.i/o',13),
        ||left('read',9)||left('write',11)||left('read',9),
        ||left('write',11)||left('read',9)||left('write',11),
        ||left('read',9)||left('write',8)||left('logical',9),
        ||left('physical',11)||left('logical',9)||left('physical',14),
        ||left('sent',6)||left('received',14)||left('calls',12)
Omv.6   =left(' ',4,' ')||left('-',270,'-')
   "EXECIO * DISKW s30omvs (STEM Omv.)"
  "ALLOC FILE(s30pr) DA("r30pr")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(9,9) CATALOG",
  "REUSE LRECL(240) RECFM(F B) BLKSIZE(27840)"
Prf.1  =left(' ',8,' '),
        ||'Performance Report for USS jobs/tasks'||left(' ',15,' ')
Prf.2  ='  '
Prf.3  =left(' ',8,' ')||'Report produced on',
        ||left(' ',1,' ')||left(date(),11),
        ||left(' ',1,' ')||left('at ',3,' ')||left(time(),10)
Prf.4  =left(' ',86,' ')||left('-',12,'-')||left('Service units   ',17),
        ||left('-',12,'-')||left(' ',6,' '),
        ||left("Transaction timings (msec)",28),
        ||left('Independent enclave',25)||left('Exec',9),
        ||left('-',9,'-')||left('  WLM Info  ',12)||left('-',9,'-'),
        ||left('    Perf.flags',14)
Prf.5  =left('OBS',4,)||left('Date',12)||left('Time',9),
        ||left('User id',9)||left('Job id',8),
        ||left('Job name',9)||left('Step name',11),
        ||left('Proc.ID',9)||left('Program',13),
        ||left('TOTAL',12)||left('CPU',10)||left('SRB',10),
        ||left('I/O',10)||left('MSO',7)||left('active',11),
        ||left('RS resident',14)||left('num.',5),
        ||left('time',6)||left('cpu su',7)||left('count',6),
        ||left('eligible',9)||left('workload',9)||left('serv cl.',10),
        ||left('res grp.',11)||left('rpt cl.',10),
        ||left('pf1 pf2',9)
Prf.6   =left(' ',4,' ')||left('-',234,'-')
   "EXECIO * DISKW s30pr (STEM Prf.)"
  "ALLOC FILE(s30pp) DA("r30pp")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(9,9) CATALOG",
  "REUSE LRECL(225) RECFM(F B) BLKSIZE(27900)"
Stg.1  =left(' ',8,' '),
        ||'Storage & Paging Report for USS jobs/tasks'
Stg.2  ='  '
Stg.3  =left(' ',8,' ')||'Report produced on',
        ||left(' ',1,' ')||left(date(),11),
        ||left(' ',1,' ')||left('at ',3,' ')||left(time(),47),
        ||left('-',26,'-')||left(' P A G I N G   C O U N T S ',28),
        ||left('-',26,'-')||left(' ',6,' ')||left('-',14,'-'),
        ||left(' STORAGE MAP ',14)||left('-',14,'-')
```

```
Stg.4  =left(' ',8Ø,' ')||left('Region',16)||left('Aux',1Ø),
        ||left('ES',5)||left('Common',7)||left('LPA',4),
        ||left('Hiper-page',11)||left('Pages',1Ø)||left('VIO pages',2Ø),
        ||left('SWAP',9)||left('Getmain size',14),
        ||left('Priv.',9)||left('User subpools',14),
        ||left('lsqa & swa',11)||left('data',4)
Stg.5  =left('OBS',4,)||left('Date',12)||left('Time',9),
        ||left('User id',9)||left('Job id',8),
        ||left('Job name',9)||left('Step name',11),
        ||left('Proc.ID',9)||left('Program',9)||left("size(K)",8),
        ||left('Total',6)||left('In',4)||left('Out',6)||left('In',4),
        ||left('Out',6)||left('In',5)||left('In',5)||left('In',4),
        ||left('Out',4)||left('stolen',7)||left('TOT',5),
        ||left('In',4)||left('Out',4)||left('Reclm',7),
        ||left('TOT',5)||left('In',4)||left('Out',5)||left('bot.',5),
        ||left('top',5)||left('<16M',5)||left('>16M',1Ø),
        ||left('<16M',5)||left('>16M',5)||left('<16M',5),
        ||left('>16M',6)||left('space',5)
Stg.6   =left(' ',4,' ')||left('-',22Ø,'-')
  "EXECIO * DISKW s3Øpp (STEM Stg.)"
/*-----------------------------------------------------------------*/
/* Part 2: Read  SMF record type 3Ø.4,                             */
/*         process selected sections & write out to report file    */
/*-----------------------------------------------------------------*/
  'EXECIO *  DISKR SMF3Ø (STEM x. FINIS'
    numeric digits 2Ø
    Do i = 1 to x.Ø
    smftype  = c2d(SUBSTR(x.i,2,1))             /*  SMF record type*/
    smfstype = c2d(SUBSTR(x.i,19,2))            /*   Record subtype*/
    IF smftype = '3Ø'  Then
    Do
   smfdate = SUBSTR(c2x(SUBSTR(x.i,7,4)),3,5)     /* Unpack SMF date*/
   smftime = smf(c2d(SUBSTR(x.i,3,4)))            /* Decode SMF time*/
   sysid   = SUBSTR(x.i,11,4)               /* System identification*/
     wid   = SUBSTR(x.i,15,4)                        /* Subsystem id*/
/*    Subsys  Section                                              */
   sof = c2d(SUBSTR(x.i,21,4))           /* Offset to subsys section*/
   sln = c2d(SUBSTR(x.i,25,2))           /* Lenght to subsys section*/
   son = c2d(SUBSTR(x.i,27,2))           /*Number to subsys sections*/
 IF sof <> Ø AND sln <> Ø Then do
    sof=sof -3
    rvn = SUBSTR(x.i,sof,2)                        /*  Product level*/
    pnm = SUBSTR(x.i,sof+2,8)                      /*   Product name*/
 END
/*            Job / Session ID section                           */
   iof = c2d(SUBSTR(x.i,29,4))        /*Offset to job/sess. section */
   iln = c2d(SUBSTR(x.i,33,2))        /*Length of job/sess. section */
   ion = c2d(SUBSTR(x.i,35,2))        /*Number of job/sess. sections*/
 IF iof <> Ø AND iln <> Ø Then do
    iof=iof -3
```

```
    jbn =  SUBSTR(x.i,iof,8)                  /* Job or session name*/
    pgm =  SUBSTR(x.i,iof+8,8)                /*        Program name*/
    rsd = SUBSTR(c2x(SUBSTR(x.i,iof+12,4)),3,5)
    stm =  SUBSTR(x.i,iof+16,8)               /*           Step name*/
    uif =  SUBSTR(x.i,iof+24,8)               /* User identification*/
    jnm =  SUBSTR(x.i,iof+32,8)               /*  JES job identifier*/
    stn =  c2d(SUBSTR(x.i,iof+40,2))          /*         Step number*/
    ssn =  c2d(SUBSTR(x.i,iof+164,4))         /*     Sub-Step Number*/
    exn =  SUBSTR(x.i,iof+168,16)             /*Exec'ed program name*/
    ttp =  SUBSTR(c2x(SUBSTR(x.i,iof+168,16)),15,2)
       SELECT
        when ttp = '40' then pgmtype='MVS'
        otherwise pgmtype='USS'
       END
    ast = smf(c2d(SUBSTR(x.i,iof+48,4)))     /* start of alloc event*/
    alloc =   c2d(SUBSTR(x.i,iof+48,4))      /*  start of allocation*/
    pps = smf(c2d(SUBSTR(x.i,iof+52,4)))     /*   program start time*/
    load =    c2d(SUBSTR(x.i,iof+52,4))      /*  program fetch event*/
    sit = smf(c2d(SUBSTR(x.i,iof+56,4)))     /*   init selected time*/
    init=     c2d(SUBSTR(x.i,iof+56,4))      /*   step initiate time*/
    rst = smf(c2d(SUBSTR(x.i,iof+64,4)))     /*      reader end time*/
    read=     c2d(SUBSTR(x.i,iof+64,4))      /*read-in or logon time*/
    ret = smf(c2d(SUBSTR(x.i,iof+72,4)))     /*      reader end time*/
    rend=     c2d(SUBSTR(x.i,iof+72,4))      /*      unformatted ret*/
    term   =  c2d(SUBSTR(x.i,3,4))           /*     termination time*/
/* time calc */
    dsenqtm = smf(alloc-init)             /*         ds enqueue delay*/
    aloctm  = smf(load-alloc)             /*allocation delay duration*/
    exectm  = smf(term-load)              /*       execution duration*/
    elapstm = smf(term-init)              /*    step elapsed duration*/
    std = SUBSTR(c2x(SUBSTR(x.i,iof+60,4)),3,5)  /*        init date*/
    red = SUBSTR(c2x(SUBSTR(x.i,iof+76,4)),3,5)  /*      reader date*/
    grp = SUBSTR(x.i,iof+100,8)                  /*   RACF group id.*/
    rud = SUBSTR(x.i,iof+108,8)                  /*    RACF user id.*/
    tid = SUBSTR(x.i,iof+116,8)                  /*RACF terminal id.*/
    psn = SUBSTR(x.i,iof+132,8)                  /*       Step name */
 END
/*     I/O activity section                                        */
   uof = c2d(SUBSTR(x.i,37,4))             /*   Offset to I/O section*/
   uln = c2d(SUBSTR(x.i,41,2))             /*   Length of I/O section*/
   uon = c2d(SUBSTR(x.i,43,2))             /*  Number of I/O sections*/
 IF uof <> 0 AND uln <> 0 Then do
    uof=uof -3
    inp = c2d(SUBSTR(x.i,uof,4))          /*    Card reads in DD Data*/
    tep = c2d(SUBSTR(x.i,uof+4,4))        /* Total blocks transferred*/
    tcn = c2d(SUBSTR(x.i,uof+18,4))
    tcn =tcn*128E-6;                      /*Total device connect time*/
    aic = c2d(SUBSTR(x.i,uof+32,4))
    aic =aic*128E-6;                      /*    DASD I/O connect time*/
    aid = c2d(SUBSTR(x.i,uof+36,4))
```

```
    aid =aid*128E-6;                    /* DASD I/O disconnect time*/
    aiw = c2d(SUBSTR(x.i,uof+40,4))
    aiw =aiw*128E-6;                    /*    DASD I/O pending time*/
    ais = c2d(SUBSTR(x.i,uof+44,4))     /*     DASD I/O start count*/
    eic = c2d(SUBSTR(x.i,uof+48,4))
    eic =eic*128E-6;                    /*DASD I/O conn.independent*/
    eid = c2d(SUBSTR(x.i,uof+52,4))
    eid =eid*128E-6;                    /* DASD I/O discon.indepen.*/
    eiw = c2d(SUBSTR(x.i,uof+56,4))
    eiw =eiw*128E-6;                    /*      DASD I/O disconnect*/
  END
/*       Completion  code section                                    */
   tof = c2d(SUBSTR(x.i,45,4))           /*   Offset to CC section */
   tln = c2d(SUBSTR(x.i,49,2))           /*   Length of CC section */
   ton = c2d(SUBSTR(x.i,51,2))           /*   Number of CC sections*/
 IF tof <> 0 AND tln <> 0 Then do
    tof=tof -3
    scc = c2d(SUBSTR(x.i,tof,2))           /* Step completion code*/
    sti = c2d(SUBSTR(x.i,tof+2,2))         /*  Step term.indicator*/
 END
/*      Processor Section                                            */
   cof = c2d(SUBSTR(x.i,53,4))           /*   Offset to CPU section */
   cln = c2d(SUBSTR(x.i,57,2))           /*   Length of CPU section */
   con = c2d(SUBSTR(x.i,59,2))           /*   Number of CPU sections*/
 IF cof <> 0 AND cln <> 0 Then do
    cof=cof -3
    cpt = c2d(SUBSTR(x.i,cof+4,4))      /* Step cpu time under TCB  */
    cps = c2d(SUBSTR(x.i,cof+8,4))      /* Step cpu time under SRB  */
    iip = c2d(SUBSTR(x.i,cof+44,4))     /* I/O interrupts time      */
    rct = c2d(SUBSTR(x.i,cof+48,4))     /* Region control task time */
    hpt = c2d(SUBSTR(x.i,cof+52,4))     /* Hiperspace transfer time */
    csc = c2d(SUBSTR(x.i,cof+56,4))     /* Crypto service count     */
    asr = c2d(SUBSTR(x.i,cof+68,4))     /* Preemptable/Client SRB   */
    enc = c2d(SUBSTR(x.i,cof+72,4))     /* Independent encl.CPU time*/
    det = c2d(SUBSTR(x.i,cof+76,4))     /* Dependent encl. CPU time */
    cep = c2d(SUBSTR(x.i,cof+80,4))     /* CPU time while enqueue   */
 totcpu= cpt + cps + iip + rct + hpt + csc + asr + enc + det + cep
 END
/*      Storage & Paging Section                                     */
   rof = c2d(SUBSTR(x.i,53,4))           /*   Offset to STOR section */
   rln = c2d(SUBSTR(x.i,57,2))           /*   Length of STOR section */
   ron = c2d(SUBSTR(x.i,59,2))           /*   Number of STOR sections*/
 IF rof <> 0 AND rln <> 0 Then do
    rof=rof -3
    sfl = c2d(SUBSTR(x.i,rof+2,1))      /* Storage flags            */
    prv = c2d(SUBSTR(x.i,rof+4,2))      /* Getmain size from bottom */
    sys = c2d(SUBSTR(x.i,rof+6,2))      /* Getmain size from top    */
    pgi = c2d(SUBSTR(x.i,rof+8,4))      /* unblk. pages: in from aux*/
    pgo = c2d(SUBSTR(x.i,rof+12,4))     /* unblk. pages: out to aux */
    cpm = c2d(SUBSTR(x.i,rof+16,4))     /* read data from expanded  */
```

```
    nsw = c2d(SUBSTR(x.i,rof+20,4))     /* no.ASID swap sequence    */
    psi = c2d(SUBSTR(x.i,rof+24,4))     /* pages swapped:in from aux*/
    pso = c2d(SUBSTR(x.i,rof+28,4))     /* pages swapped: out to aux*/
    vpi = c2d(SUBSTR(x.i,rof+32,4))     /* number of vio page ins   */
    vpo = c2d(SUBSTR(x.i,rof+36,4))     /* number of vio page outs  */
    vpr = c2d(SUBSTR(x.i,rof+40,4))     /* number of vio reclaims   */
    cpi = c2d(SUBSTR(x.i,rof+44,4))     /* common area page-ins     */
    hpi = c2d(SUBSTR(x.i,rof+48,4))     /* hiper page ins           */
    lpi = c2d(SUBSTR(x.i,rof+52,4))     /* lpa area page-ins        */
    hpo = c2d(SUBSTR(x.i,rof+56,4))     /* hiper page outs          */
    pst = c2d(SUBSTR(x.i,rof+60,4))     /* pages stolen away        */
    psc = c2d(SUBSTR(x.i,rof+64,8))     /* no.of cpu page seconds   */
    rgb = c2d(SUBSTR(x.i,rof+72,4))     /* private area size <16M   */
    erg = c2d(SUBSTR(x.i,rof+76,4))     /* private area size >16M   */
    arb = c2d(SUBSTR(x.i,rof+80,4))     /* max vir.lsqa and swa <16M*/
    ear = c2d(SUBSTR(x.i,rof+84,4))     /* max vir.lsqa and swa >16M*/
    urb = c2d(SUBSTR(x.i,rof+88,4))     /* max vir.user subpools<16M*/
    eur = c2d(SUBSTR(x.i,rof+92,4))     /* max vir.user subpools>16M*/
    rgn = c2d(SUBSTR(x.i,rof+96,4))     /* region size              */
    dsv = c2d(SUBSTR(x.i,rof+100,4))    /* data space storage used  */
    pie = c2d(SUBSTR(x.i,rof+104,4))    /* unblk. pages:  in from es*/
    poe = c2d(SUBSTR(x.i,rof+108,4))    /* unblk. pages:  out to es.*/
    bia = c2d(SUBSTR(x.i,rof+112,4))    /* blocks paged: in from aux*/
    boa = c2d(SUBSTR(x.i,rof+116,4))    /* blocks paged:  out to aux*/
    bie = c2d(SUBSTR(x.i,rof+120,4))    /* blocked pages: in from es*/
    boe = c2d(SUBSTR(x.i,rof+124,4))    /* blocked pages: out to es.*/
    kia = c2d(SUBSTR(x.i,rof+128,4))    /* blocks pages: in from aux*/
    koa = c2d(SUBSTR(x.i,rof+132,4))    /* blocks pages:  out to aux*/
 /* psf = c2d(SUBSTR(x.i,rof+144,8))       cpu page seconds         */
 /* pai = c2d(SUBSTR(x.i,rof+152,4))       shared pages paged in AUX*/
    pei = c2d(SUBSTR(x.i,rof+156,4))    /* shared pages paged in EXP*/
    ers = c2x(SUBSTR(x.i,rof+160,8))    /* EXP residency time       */
    mem = c2x(SUBSTR(x.i,rof+168,8))    /* MEMLIMIT value in MB     */
    mls = c2x(SUBSTR(x.i,rof+170,1))    /* MEMLIMIT source          */
    psc = psc * 1.024                   /* % 1000000 for sec. value */
 page = pgi+pgo+bia+boa+kia+koa+pie+poe+bie+boe+cpi+lpi+hpi+hpo
 swap = psi + pso                       /*         total swap count */
 vio = vpi + vpo + vpr                  /*          total vio count */
 auxin = pgi + bia + kia                /*         page in from aux.*/
 auxout= pgo + boa + koa                /*          page out to aux.*/
 esin=pie + bie                         /*          page in from es.*/
 esout=poe + boe                        /*           page out to es.*/
 END
/*    Performance  Section                                          */
   pof = c2d(SUBSTR(x.i,77,4))           /* Offset to PERF. section */
   pln = c2d(SUBSTR(x.i,81,2))           /* Length of PERF. section */
   pon = c2d(SUBSTR(x.i,83,2))           /* Number of PERF. sections*/
 IF pof <> 0 AND pln <> 0 Then do
    pof=pof -3
    srv = c2d(SUBSTR(x.i,pof,4))        /* Total service units      */
```

```
    csu = c2d(SUBSTR(x.i,pof+4,4))      /* CPU service units        */
    srb = c2d(SUBSTR(x.i,pof+8,4))      /* SRB service units        */
    io  = c2d(SUBSTR(x.i,pof+12,4))     /* I/O service units        */
    mso = c2d(SUBSTR(x.i,pof+16,4))     /* MSO service units        */
    tat = c2d(SUBSTR(x.i,pof+20,4))     /* SRM transaction time     */
    res = c2d(SUBSTR(x.i,pof+28,4))     /* SRM residency   time     */
    trs = c2d(SUBSTR(x.i,pof+32,4))     /* SRM transactions count   */
    wlm = SUBSTR(x.i,pof+36,8)          /* Workload Name            */
    scn = SUBSTR(x.i,pof+44,8)          /* Service Class Name       */
    grn = SUBSTR(x.i,pof+52,8)          /* Resource Group Name      */
    rcn = SUBSTR(x.i,pof+60,8)          /* Reporting Class Name     */
    eta = c2d(SUBSTR(x.i,pof+68,4))     /* Independent Enclave time */
    esu = c2d(SUBSTR(x.i,pof+72,4))     /* Independent Enclave cpusu*/
    etc = c2d(SUBSTR(x.i,pof+76,4))     /* Independent Enclave count*/
    pfl =     SUBSTR(x.i,pof+80,16)     /* Scheduling env. name     */
    jqt = c2d(SUBSTR(x.i,pof+96,4))     /* JCL conversion time      */
    rqt = c2d(SUBSTR(x.i,pof+100,4))    /* hold time - affinity     */
    hqt = c2d(SUBSTR(x.i,pof+104,4))    /* hold time - no affinity  */
    sqt = c2d(SUBSTR(x.i,pof+108,4))    /* execution eligible time  */
    pf1 = c2x(SUBSTR(x.i,pof+112,1))    /* Performance flags - 1    */
     SELECT
      when pf1 = 80 then note1 ='Reset before initiation          '
      when pf1 = 40 then note1 ='Reset after  initiation          '
      when pf1 = 20 then note1 ='Immediate initiation             '
      when pf1 = 10 then note1 ='Job has been restarted           '
      when pf1 = 08 then note1 ='Remote Sys. Data incomplete      '
      when pf1 = 04 then note1 ='Job executing in a WLM batch init'
      when pf1 = 02 then note1 ='Serv. class CPU-critical         '
      when pf1 = 01 then note1 ='Serv. class stg-critical         '
      otherwise  nop
     END
    pf2 = c2x(SUBSTR(x.i,pof+113,1))     /* Performance flags - 2    */
     SELECT
      when pf2 = 80 then note2 ='AS designated stg-critical        '
      when pf2 = 40 then note2 ='AS cannot be managed to trxn goals'
      when pf2 = 20 then note2 ='AS is currently CPU-protected     '
      when pf2 = 10 then note2 ='AS is currently stg-protected     '
      otherwise nop
     END
    jpn = SUBSTR(x.i,pof+116,8)         /*    Subsys collection name */
                                        /*  % 1000000 for sec. value */
    active =  tat * 1.024               /*           task active time*/
    reside =  res * 1.024               /*   resident in real storage*/
    sqt1   =  sqt * 1.024
 END
/*  USS section                                                     */
   opo = c2d(SUBSTR(x.i,125,4))             /*Offset to USS section */
   opl = c2d(SUBSTR(x.i,129,2))             /*Length of USS section */
   opn = c2d(SUBSTR(x.i,131,2))             /*Number of USS sections*/
   opm = c2d(SUBSTR(x.i,133,4))             /*Number of USS sections*/
```

```
 IF opo <> Ø AND opl <> Ø and opn <> Ø then do
    totio.i = Ø
    totcal.i = Ø
  do w = Ø to opn -1
    opo1 = (opo + (w*opl)) - 3
    opi = c2d(SUBSTR(x.i,opo1,4))        /* Proc. ID                 */
    opp.w = c2d(SUBSTR(x.i,opo1+72,4))   /* Parent process ID no.    */
    opg = c2d(SUBSTR(x.i,opo1+4,4))      /* Process Group ID         */
    oui = c2d(SUBSTR(x.i,opo1+8,4))      /* Process User ID          */
    oug = c2d(SUBSTR(x.i,opo1+12,4))     /* Process User Group ID    */
    osi = c2d(SUBSTR(x.i,opo1+16,4))     /* Process Session ID       */
    osc = c2d(SUBSTR(x.i,opo1+2Ø,4))     /* Number of USS Syscalls   */
    ost = c2d(SUBSTR(x.i,opo1+24,4))     /* Total CPU time           */
    odr.w = c2d(SUBSTR(x.i,opo1+28,4))   /* No.dir. I/O blocks read p*/
    ofr.w = c2d(SUBSTR(x.i,opo1+32,4))   /* No.dir. I/O blocks read s*/
    ofw.w = c2d(SUBSTR(x.i,opo1+36,4))   /* No.dir. I/O blocks wrt. s*/
    opr.w = c2d(SUBSTR(x.i,opo1+4Ø,4))   /* No.dir. I/O blocks read i*/
    opw.w = c2d(SUBSTR(x.i,opo1+44,4))   /* No.dir. I/O blocks wrt. i*/
    osr.w = c2d(SUBSTR(x.i,opo1+48,4))   /* No.dir. I/O blocks read c*/
    osw.w = c2d(SUBSTR(x.i,opo1+52,4))   /* No.dir. I/O blocks wrt. c*/
    okr.w = c2d(SUBSTR(x.i,opo1+76,4))   /* I/O blocks read -Remote  */
    okw.w = c2d(SUBSTR(x.i,opo1+8Ø,4))   /* I/O blocks wrt. -Remote  */
    oll.w = c2d(SUBSTR(x.i,opo1+56,4))   /* Look calls - path log.   */
    olp.w = c2d(SUBSTR(x.i,opo1+6Ø,4))   /* Look calls - path phys.  */
    ogl.w = c2d(SUBSTR(x.i,opo1+64,4))   /* Gen. calls - path log.   */
    ogp.w = c2d(SUBSTR(x.i,opo1+68,4))   /* Gen. calls - path phys.  */
    oms.w = c2d(SUBSTR(x.i,opo1+84,4))   /* Message queues BY. sent  */
    omr.w = c2d(SUBSTR(x.i,opo1+88,4))   /* Message queues BY. rec.  */
    osy.w = c2d(SUBSTR(x.i,opo1+92,4))   /* No.sync() function calls */
   time = SUBSTR(smftime,1,7)
totio.i = totio.i + odr.w + ofr.w + ofw.w + opr.w + opw.w ,
          + osr.w + osw.w ,
          + okr.w + okw.w
totcal.i= totcal.i + oll.w + olp.w + ogl.w + ogp.w + osy.w
c3Ømvs =right(i,3,'Ø')  right(Date('N',smfdate,'J'),11),
         left(smftime,8) right(rud,8)    left(jnm,8),
         left(jbn,8)     left(stm,8)     left(exn,9),
         right(opi,8)    right(opp.w,9)  right(oui,9),
         right(osc,6)    right(ost,6)    right(odr.w,9),
         right(ofr.w,9)  right(ofw.w,9)  right(opr.w,9),
         right(opw.w,9)  right(osr.w,9)  right(osw.w,9),
         right(okr.w,9)  right(okw.w,9)  right(oll.w,9),
         right(olp.w,9)  right(ogl.w,9)  right(ogp.w,9),
         right(oms.w,9)  right(omr.w,9)  right(osy.w,9)
  PUSH c3Ømvs
    "EXECIO 1 DISKW s3Ømvs"
/*      Match/Merge portion of SMF 3Ø.4                              */
 c3Øout = left(smfdate,5) left(time,1Ø),
          right(opi,9)    left(jbn,8),
          left(stm,8)     left(exn,16) left(smftime,8)
```

```
  PUSH c3Øout
    "EXECIO 1 DISKW s3Øuu"
/*   EXCP Section                                                  */
   eof = c2d(SUBSTR(x.i,93,4))          /*  Offset to EXCP section */
   eln = c2d(SUBSTR(x.i,97,2))          /*  Length of EXCP section */
   eon = c2d(SUBSTR(x.i,99,2))          /*  Number of EXCP sections*/
   eor = c2d(SUBSTR(x.i,1Ø1,2))         /*  Number of EXCP sections*/
   eos = c2d(SUBSTR(x.i,1Ø5,4))         /*  Number of EXCP sections*/
 IF eof <> Ø AND eln <> Ø AND eon <> Ø Then do
    eof=eof -3
    totexcp.i = Ø
  do k = Ø to eon -1
    incr = (eof + (k*eln))
    ddn.k = SUBSTR(x.i,incr+4,8)
    blk.k = c2d(SUBSTR(x.i,incr+12,4))
    totexcp.i =totexcp.i + blk.k
   end
   end
 end
  END
/*      Summary Report (no.1)                                      */
   c3Øsum= right(i,3,'Ø')   right(Date('N',smfdate,'J'),11),
         left(smftime,8)  right(rud,8)      left(jnm,8),
         left(jbn,8)      left(stm,8)       right(opi,8),
         left(exn,16)     right(scc,4),
         right(cpt,4)     right(cps,4)      right(totcpu,5),
         left(exectm,12)  left(elapstm,12)  right(srv,9),
         right(tep,8)     right(totexcp.i,7),
         right(totio.i,7) right(totcal.i,7),
         right(tcn,1Ø)    right(page,7),
         right(swap,7)    right(vio,7)
  PUSH c3Øsum
    "EXECIO 1 DISKW s3Øss"
/*      I/O Report (no.2)                                          */
  c3Øio= right(i,3,'Ø') right(Date('N',smfdate,'J'),11),
         left(smftime,8),
         right(rud,8)      left(jnm,8),
         left(jbn,8)       left(stm,8),
         right(opi,8)      left(exn,16),
         right(dsenqtm,12) right(aloctm,12),
         right(tep,8)      right(tcn,9),
         right(aic,9)      right(aid,9) right(aiw,9),
         right(eic,9)      right(eid,9) right(eiw,9)
  PUSH c3Øio
    "EXECIO 1 DISKW s3Øio"
/*      Processor Report (no.3)                                    */
   c3Øcp = right(i,3,'Ø') right(Date('N',smfdate,'J'),11),
         left(smftime,8)  right(rud,8)  left(jnm,8),
         left(jbn,8)      left(stm,8)   right(opi,8),
         left(exn,1Ø),
```

```
         right(totcpu,6)  right(cpt,5)  right(cps,5),
         right(iip,5)     right(rct,5)  right(hpt,5),
         right(csc,5)     right(asr,5),
         right(enc,5)     right(det,5)  right(cep,5)
  PUSH c30cp
    "EXECIO 1 DISKW s30cp"
/*      Storage & Paging Report (no.4)                              */
 c30pp= right(i,3,'0')  right(Date('N',smfdate,'J'),11),
       left(smftime,8)  right(rud,8)     left(jnm,8),
       left(jbn,8)      left(stm,8)      right(opi,8),
       left(exn,10)     right(rgn,5)     right(page,4),
       right(auxin,4)   right(auxout,4)  right(esin,4),
       right(esout,4)   right(cpi,4)     right(lpi,4),
       right(hpi,4)     right(hpo,4)     right(pst,4),
       right(vio,4)     right(vpi,4)     right(vpo,4),
       right(vpr,4)     right(swap,4)    right(psi,4),
       right(pso,4)     right(prv,4)     right(sys,4),
       right(rgb,4)     right(erg,4)     right(urb,10),
       right(eur,4)     right(arb,4)     right(ear,4),
 right(dsv,4)                           /* data space storage used */
  PUSH c30pp
    "EXECIO 1 DISKW s30pp"
/*      Performance  Report (no.5)                                  */
    c30pr= right(i,3,'0')  right(Date('N',smfdate,'J'),11),
       left(smftime,8)  right(rud,8)     left(jnm,8),
       left(jbn,8)      left(stm,8)      right(opi,8),
       left(exn,9)      right(srv,9)     right(csu,9),
       right(srb,9)     right(io,9)      right(mso,9),
       right(active,12) right(reside,12) right(trs,5),
       right(eta,5)     right(esu,5)     right(etc,5),
       right(sqt1,10)   left(wlm,9)      left(scn,9),
       left(grn,9)      left(rcn,9)      right(pf1,3) right(pf2,3)
  PUSH c30pr
 "EXECIO 1 DISKW s30pr"
  end
 end
/* Close & free all allocated files  */
 "EXECIO 0 DISKW s30pr(FINIS "
 "EXECIO 0 DISKW s30uu(FINIS "
 "EXECIO 0 DISKW s30io(FINIS "
 "EXECIO 0 DISKW s30ss(FINIS "
 "EXECIO 0 DISKW s30cp(FINIS "
 "EXECIO 0 DISKW s30pp(FINIS "
 "EXECIO 0 DISKW s30omvs(FINIS "
 say
 say 'I/O Activity report dsn . . . . :'r30io
 say 'Summary Report dsn . . . . . .:'r30ss
 say 'Processor usage Report dsn . . .:'r30cp
 say 'USS Process Report dsn . . . . .:'r30om
 say 'Performance Report dsn . . . . .:'r30pr
```

```
 say 'Storage & Paging Report dsn  . .:'r30pp
 say
 "free FILE(SMF30 s30uu s30io s30ss s30cp s30omvs s30pr s30pp)"
exit
SMF: PROCEDURE
/* REXX - convert an SMF time to hh:mm:ss:hd format   */
arg time
    time1    = time % 100
    hh       = time1 % 3600
    hh       = RIGHT("0"||hh,2)
    mm       = (time1 % 60) - (hh * 60)
    mm       = RIGHT("0"||mm,2)
    ss       = time1 - (hh * 3600) - (mm * 60)
    ss       = RIGHT("0"||ss,2)
    fr       = time //  1000
    fr       = RIGHT("0"||fr,2)
    rtime  = hh||":"||mm||":"||ss||":"||fr
    return rtime
```

USSFILE EXEC

```
/* REXX EXEC (USSFILE):read and format USS File System SMF record */
ADDRESS TSO
 userid=SYSVAR(SYSUID)
/* Part 1: Handle file allocation & dataset existence and         */
/*         print report header and labels                         */
Address TSO
  userid=SYSVAR(SYSUID)
  r92ti  =userid||'.ussfile.rep'        /* USS close file Report   */
  r92fi  =userid||'.r92fi.rep'          /* Temp file               */
  r92mt  =userid||'.mat92.out'          /* Match/Merge file        */
  x = MSG('OFF')
  if SYSDSN(r92ti) = 'OK'             /* check report dsn validity */
  Then "DELETE "r92ti" PURGE"
  if SYSDSN(r92fi) = 'OK'             /* check report dsn validity */
  Then "DELETE "r92fi" PURGE"
  if SYSDSN(r92mt) = 'OK'             /* check report dsn validity */
  Then "DELETE "r92mt" PURGE"
  "ALLOC FILE(ussti) DA("r92ti")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(60,15) CATALOG",
  "REUSE LRECL(240) RECFM(F B) BLKSIZE(27840)"
 Out.1 = left(' ',8,' '),
         ||'USS File System Read & Write Activity Report ',
         ||left(' ',15,' ')
 Out.2 = '  '
 Out.3 =left(' ',8,' ')||'Report produced on',
      ||left(' ',1,' ')||left(date(),11),
      ||left(' ',1,' ')||left('at ',3,' ')||left(time(),10)
 Out.4 = left(' ',177,' ')||left('No.Calls',12),
```

```
      ||left('I/O blocks',23)||left('Bytes',5)
 Out.5 =left('Date',12)||left('Time',9)||left('User id',9),
      ||left('Job name',9)||left('Step name',11),
      ||left('Proc.ID',8)||left('Open time',13)||left('Close time',13),
      ||left('Opened',13)||left('Path name',64)||left("File #",7),
      ||left('Device',7)||left('Read',7)||left('Write',7),
      ||left("Dir",4)||left('read',5)||left('write',10)||left('read',7),
      ||left('write',7)||left('Record flag',18)
 Out.6  =left('-',234,'-')
 "EXECIO * DISKW ussti (STEM Out. FINIS)"
 "free FILE(ussti)"
  "ALLOC FILE(uss11) DA("r92fi")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(300,90) CATALOG",
  "REUSE LRECL(250) RECFM(F B) BLKSIZE(27750)"
  "ALLOC FILE(uss92) DA("r92mt")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(450,90) CATALOG",
  "REUSE LRECL(235) RECFM(F B) BLKSIZE(27965)"
/* Part 2: Read  SMF record type 92.11,                              */
/*         process selected section & write out to report file       */
 'EXECIO * DISKR SMF92 (STEM x. FINIS'
    numeric digits 20
    do i = 1 to x.0
/*              SMF Record Header Section                            */
  smftype  = c2d(SUBSTR(x.i,2,1))               /*  SMF record type*/
  smfstype = c2d(SUBSTR(x.i,19,2))              /*   Record subtype*/
  IF smftype = '92'  Then
  Do
   smfdate = SUBSTR(c2x(SUBSTR(x.i,7,4)),3,5)   /*  Unpack SMF date*/
   smftime = cmf(c2d(SUBSTR(x.i,3,4)))          /*  Decode SMF time*/
   sysid   = SUBSTR(x.i,11,4)             /*  System identification*/
   wid     = SUBSTR(x.i,15,4)             /*   Subsystem identifier*/
/*              Self Defining Section                                */
   sof   = c2d(SUBSTR(x.i,25,4))      /*Offset to subsystem section*/
   sln   = c2d(SUBSTR(x.i,29,2))      /*Length of subsystem section*/
   son   = c2d(SUBSTR(x.i,31,2))     /*Number of subsystem sections*/
   iof   = c2d(SUBSTR(x.i,33,4)) /*Offset of identification section*/
   iln   = c2d(SUBSTR(x.i,37,2)) /*Length of identification section*/
   ion   = c2d(SUBSTR(x.i,39,2)) /*Number of identification section*/
   dof   = c2d(SUBSTR(x.i,41,4))        /*   Offset of data section*/
   dln   = c2d(SUBSTR(x.i,45,2))        /*   Length of data section*/
   don   = c2d(SUBSTR(x.i,47,2))        /*  Number of data sections*/
    IF sof <> 0 AND son <> 0 Then do
       sof=sof-3
/*              Subsystem Section                                    */
       typ    = c2d(SUBSTR(x.i,sof,2))    /*        Record subtype,*/
                                          /*       11 = File close.*/
       rvn    = SUBSTR(x.i,sof+2,2)       /*  Record version number*/
       pnm    = SUBSTR(x.i,sof+4,8)       /* Product name - OpenMVS*/
       osl    = SUBSTR(x.i,sof+12,8)      /*      MVS product level*/
       SELECT
```

```
          when rvn = 1 then version='OMVS - SP4.3  '
          when rvn = 2 then version='OMVS - SP5.1  '
          when rvn = 3 then version='OMVS - SP5.2.2'
        END
      end
    IF iof <> 0 and iln <> 0 Then do
       iof=iof-3
/*              Identification Section                              */
       jbn   =  SUBSTR(x.i,iof,8)            /*              Job Name*/
       rst = cmf(c2d(SUBSTR(x.i,iof+8,4)))   /*     Reader start time*/
       rsd = SUBSTR(c2x(SUBSTR(x.i,iof+12,4)),3,5) /*  Rdr start date*/
       stm =  SUBSTR(x.i,iof+16,8)           /*             Step name*/
       rgd =  SUBSTR(x.i,iof+24,8)           /*          SAF group Id*/
       rud =  SUBSTR(x.i,iof+32,8)           /*           SAF user Id*/
       uid =  c2d(SUBSTR(x.i,iof+40,4))      /*      USS real user Id*/
       gid =  c2d(SUBSTR(x.i,iof+44,4))      /*     USS real group Id*/
       pid =  c2d(SUBSTR(x.i,iof+48,4))      /*        USS process Id*/
       pgd =  c2d(SUBSTR(x.i,iof+52,4))      /*  USS process group Id*/
       ssd =  c2d(SUBSTR(x.i,iof+56,4))      /*        USS session Id*/
       api =  c2d(SUBSTR(x.i,iof+60,4))      /* USS anchor process Id*/
       apg =  c2d(SUBSTR(x.i,iof+64,4))      /*USS anchor proc grp Id*/
       asg =  c2d(SUBSTR(x.i,iof+68,4))      /* USS anchor session Id*/
    end
    IF dof <> 0 and dos <> 0 Then do
       dof=dof-3
/*   Data Section for File Close subtype (subtype 11)               */
       cto =  SUBSTR(st(c2x(SUBSTR(x.i,dof,8))),12,15) /*Time of open*/
       ctc =  SUBSTR(st(c2x(SUBSTR(x.i,dof+8,8))),12,15) /*Time close*/
       cty =  c2d(SUBSTR(x.i,dof+16,1))       /*            File type*/
       cfg =      SUBSTR(x.i,dof+17,1)        /*          Record flag*/
       cf2 =  X2B(c2x(SUBSTR(x.i,dof+17,1)))  /*     Record flag bin.*/
       ctk =  c2d(SUBSTR(x.i,dof+20,4))       /*      Open file token*/
       cin =  c2d(SUBSTR(x.i,dof+24,4))       /*         Inode number*/
       cdn =  c2d(SUBSTR(x.i,dof+28,4))       /* Unique device number*/
       csr =  c2d(SUBSTR(x.i,dof+32,4))       /*                Reads*/
       csw =  c2d(SUBSTR(x.i,dof+36,4))       /*               Writes*/
       cdi =  c2d(SUBSTR(x.i,dof+40,4))       /* Directory I/O blocks*/
       cir =  c2d(SUBSTR(x.i,dof+44,4))       /*      I/O blocks read*/
       ciw =  c2d(SUBSTR(x.i,dof+48,4))       /*   I/O blocks written*/
       cbr =  c2d(SUBSTR(x.i,dof+52,8))       /*           Bytes read*/
       cbw =  c2d(SUBSTR(x.i,dof+60,8))       /*        Bytes written*/
       SELECT
        when cfg ='00000000'b then flag ='cached file       '
        when cfg ='01000000'b then flag ='local socket      '
        when cfg ='01100000'b then flag ='network socket    '
        when cfg ='00100000'b then flag ='client socket     '
        when cfg ='00110000'b then flag ='server socket     '
        when cfg ='10000000'b then flag ='generated by VNode'
        otherwise flag = 'error'
       END
```

```
     end
    select
      when dln > 68 Then cpn =  SUBSTR(x.i,dof+68,64) /*   Pathname*/
      otherwise cpn = 'path name not available'
    end
   timedif = dif(cto,ctc)                  /*  Time file was opened*/
   time =substr(smftime,1,7)
   u92out = right(Date('N',rsd,'J'),11)   left(smftime,8),
            left(rud,8)   left(jbn,8)     left(stm,8),
            right(pid,9)  left(cto,12)    left(ctc,12),
         left(timedif,12) left(cpn,64)    right(cin,5),
            right(cdn,5)  right(csr,5)    right(csw,5),
            right(cdi,5)  right(cir,5)    right(ciw,5),
            right(cbr,8)  right(cbw,8)    right(flag,18),
            left(rsd,5)
    PUSH u92out
      "EXECIO 1 DISKW uss11"
/* Match/Merge portion of SMF 92.11 - omit TCPIP (long - running) */
 IF jbn ¬= "TCPIP"   Then do                /* Omit TCP/IP records*/
   u92comp= left(rsd,5)  left(time,10) right(pid,9),
            left(jbn,8)  left(stm,8)   left(cpn,64),
            left(cto,12) left(ctc,12)  left(timedif,12),
            right(cin,5) right(cdn,5)  right(csr,5),
            right(csw,5) right(cdi,5)  right(cir,5),
            right(ciw,5) right(cbr,8)  right(cbw,8) right(flag,18)
    PUSH u92comp
      "EXECIO 1 DISKW uss92"
     end
    end
   end
 "EXECIO 0 DISKW uss11(FINIS "
 "EXECIO 0 DISKW uss92(FINIS "
 "EXECIO 0 DISKW ussti(FINIS "
 say
 say 'USS File Read & Write Activity Report dsn . .:'r92ti
 say
 "free FILE(SMF92 uss11 uss92 ussti)"
 exit
  return
CMF: procedure expose smf_time hundrethtime
arg hundrethtime
hund   =hundrethtime%1000
seconds=hundrethtime%100
minutes=seconds%60
hours=right(minutes%60,2,"0")
minutes=right(minutes-60*hours,2,"0")
seconds=right(seconds-60*minutes-60*60*hours,2,"0")
hund   =Right(hund,2,0)
/* time=hours":"minutes":"seconds":"hund */
time=hours":"minutes":"seconds
```

```
return time
ST: Procedure
/* STCK timestamp format converted               */
/* The BLSUXTOD proc is described in "z/OS       */
/* V1R3 MVS IPCS Customization"                  */
arg todtime
If todtime    <> '0000000000000000000' Then
  Do
    TOD_Value = X2C(todtime)
    Returned_Date = '-------------------------'
    address LINKPGM "BLSUXTOD TOD_Value Returned_Date"
  End
Else
  Returned_Date = '0000000000000000000'
 /* Returned_Date = '' */
Return Returned_Date
DIF: procedure
/* Dif: REXX subroutine to find the              */
/* difference between two timestamps             */
 arg t.1,t.2
  do j=1 to 2
   _=arg(j)':0:0'
    parse var _ h ":" m ':' s ":"
    s.j= 3600 * h + 60 * m + s
  end
diff=s.2-s.1+(s.1>s.2)*86400
return right(diff%3600,2,0)':'|| ,
       right(diff//3600%60,2,0)":"|| ,
       translate(format(diff//3600//60,2),0,' ')
return
```

RMFUSS EXEC

```
/* REXX EXEC (RMFUSS) read and format RMF record 74                */
/* subtype 3: USS Kernel Activity                                   */
/* subtype 6: HFS Global Activity                                   */
/*          : Buffer pool statistics                                */
/*          : HFS File system statistics                            */
ADDRESS TSO
/* Part 1: Handle file allocation & dataset existence and           */
/*         print report header and labels                           */
  userid=SYSVAR(SYSUID)
  r7431  =userid||'.rmfsysc.rep'
  r7432  =userid||'.rmfproc.rep'
  r7433  =userid||'.rmfipc.rep'
  r7434  =userid||'.rmfmmap.rep'
  r7435  =userid||'.rmflibs.rep'
  r74gl  =userid||'.rmfglob.rep'
  r74bf  =userid||'.rmfbuff.rep'
```

```
 r74hf  =userid||'.rmfhfsf.rep'
   x = MSG('OFF')
 if SYSDSN(r7431) = 'OK'         /* check report dsn validity      */
 Then "DELETE "r7431" PURGE"
 if SYSDSN(r7432) = 'OK'         /* check report dsn validity      */
 Then "DELETE "r7432" PURGE"
 if SYSDSN(r7433) = 'OK'         /* check report dsn validity      */
 Then "DELETE "r7433" PURGE"
 if SYSDSN(r7434) = 'OK'         /* check report dsn validity      */
 Then "DELETE "r7434" PURGE"
 if SYSDSN(r7435) = 'OK'         /* check report dsn validity      */
 Then "DELETE "r7435" PURGE"
 if SYSDSN(r74gl) = 'OK'         /* check report dsn validity      */
 Then "DELETE "r74gl" PURGE"
 if SYSDSN(r74bf) = 'OK'         /* check report dsn validity      */
 Then "DELETE "r74bf" PURGE"
 if SYSDSN(r74hf) = 'OK'         /* check report dsn validity      */
 Then "DELETE "r74hf" PURGE"
 "ALLOC FILE(global) DA("r74gl")",
 "UNIT(SYSALLDA) NEW TRACKS SPACE(9,2) CATALOG",
 "REUSE LRECL(95) RECFM(F B) BLKSIZE(2793Ø)"
Gl.1  ='HFS Global statistics'
Gl.2  ='  '
Gl.3  ='Report produced on',
       ||left(' ',1,' ')||left(date(),11),
       ||left(' ',1,' ')||left('at ',3,' ')||left(time(),1Ø)
Gl.4  ='  '
Gl.5  =left(' ',28,' ')||left('-- Storage limits (MB) --',33),
      ||left('-- File I/O --',15)||left(' -- Metadata I/O --',19)
Gl.6  =left('RMF interval TOD',21),
      ||left("VIRT(MAX)",12)||left('Used',8),
      ||left("FIXED(MIN)",12)||left('Used',9),
      ||left('cache',9)||left('dasd',8),
      ||left('cache',9)||left('dasd',4)
Gl.7  =left('-',95,'-')
   "EXECIO * DISKW global (STEM Gl.)"
 "ALLOC FILE(buffer) DA("r74bf")",
 "UNIT(SYSALLDA) NEW TRACKS SPACE(9,2) CATALOG",
 "REUSE LRECL(95) RECFM(F B) BLKSIZE(2793Ø)"
Bf.1  ='Buffer pool statistics'
Bf.2  ='  '
Bf.3  ='Report produced on',
       ||left(' ',1,' ')||left(date(),11),
       ||left(' ',1,' ')||left('at ',3,' ')||left(time(),1Ø)
Bf.4  ='  '
Bf.5  =left(' ',26,' ')||left('Number',9)||left('Buffer',8),
       ||left(' -- Pool size --',24)||left('Data',6),
       ||left(' -- I/O activity --',19)
Bf.6  =left('RMF interval TOD',21)||left('Pool',6)||left('buff.',9),
       ||left('size',8)||left('pages',8)||left('MB',5),
```

```
        ||left('fixed',9)||left('spaces',7),
        ||left('total',7)||left('fixed',6)||left("not fix.",8)
 Bf.7  =left('-',94,'-')
   "EXECIO * DISKW buffer (STEM Bf.)"
  "ALLOC FILE(hfsfil) DA("r74hf")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(9,2) CATALOG",
  "REUSE LRECL(235) RECFM(F B) BLKSIZE(27965)"
 Hf.1  ='HFS File system statistics'
 Hf.2  ='  '
 Hf.3  ='Report produced on',
        ||left(' ',1,' ')||left(date(),11),
        ||left(' ',1,' ')||left('at ',3,' ')||left(time(),10)
 Hf.4  ='  '
 Hf.5  =left(' ',83,' ')||left('---- Allocation (pages) ----',40),
       ||left('------ File I/O ------',29),
       ||left('- Matadata I/O -',24),
       ||left('-- Index read/write --',32),
       ||left('-- Index Events --',24)
 Hf.6  =left('RMF interval TOD',21)||left('File system name',31),
       ||left('Mount TOD',28)||left('System',10),
       ||left('Data',6)||left('Attr.dir',11)||left('Cached',11),
       ||left('seq.',7)||left('random',10)||left('cache',10),
       ||left('dasd',8)||left('cache',10)||left('dasd',10),
       ||left('hit',8)||left('miss',10)||left('hit',8)||left('miss',7),
       ||left('new level',12)||left('split',9)||left('join',4)
 Hf.7  =left('-',231,'-')
 "EXECIO * DISKW hfsfil (STEM Hf.)"
  "ALLOC FILE(rmf01) DA("r7431")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(2,1) CATALOG",
  "REUSE LRECL(75) RECFM(F B) BLKSIZE(27975)"
Sc.1  ='USS System call activity'
Sc.2  ='  '
Sc.3  ='Report produced on',
       ||left(' ',1,' ')||left(date(),11),
       ||left(' ',1,' ')||left('at ',3,' ')||left(time(),10)
Sc.4  ='  '
Sc.5  =left(' ',28,' ')||left('--  Count  --',15),
      ||left(' ',10,' ')||left('--  CPU Time --',17)
Sc.6  =left('RMF interval TOD',23)||left('Total',7)||left('Min.',6),
       ||left('Avg.',7)||left('Max.',6)||left('Total',8)||left('Min.',7),
      ||left('Avg.',7)||left('Max.',4)
Sc.7  =left('-',75,'-')
   "EXECIO * DISKW rmf01 (STEM Sc.)"
  "ALLOC FILE(rmf02) DA("r7432")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(2,1) CATALOG",
  "REUSE LRECL(75) RECFM(F B) BLKSIZE(27975)"
Pc.1  ='USS Process activity'
Pc.2  ='  '
Pc.3  ='Report produced on',
      ||left(' ',1,' ')||left(date(),11),
```

```
      ||left(' ',1,' ')||left('at ',3,' ')||left(time(),10)
Pc.4  ='  '
Pc.5  =left(' ',21,' ')||left('-',6,'-')||left(' Count',7),
      ||left('-',6,'-')||left(' ',9,' '),
      ||left('-',6,'-')||left(' Overruns',10)||left('-',6,'-')
Pc.6  =left(' ',13,' ')||left('Parm',4)
Pc.7  =left('Interval',12,' ')||left('value',7)||left('Total',9),
      ||left('Min.',6)||left('Avg.',7)||left('Max.',6),
      ||left('Total',8)||left('Min.',7)||left('Avg.',7)||left('Max.',4)
Pc.8  =left('-',75,'-')
   "EXECIO * DISKW rmf02 (STEM Pc.)"
  "ALLOC FILE(rmf03) DA("r7433")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(2,1) CATALOG",
  "REUSE LRECL(80) RECFM(F B) BLKSIZE(27920)"
Ip.1  ='USS Inter Process Communication'
Ip.2  ='  '
Ip.3  ='Report produced on',
      ||left(' ',1,' ')||left(date(),11),
      ||left(' ',1,' ')||left('at ',3,' ')||left(time(),10)
Ip.4  ='  '
Ip.5  =left(' ',25,' ')||left('-',6,'-')||left(' Count',7),
      ||left('-',6,'-')||left(' ',9,' '),
      ||left('-',6,'-')||left(' Overruns',10)||left('-',6,'-')
Ip.6  =left(' ',17,' ')||left('Parm',4)
Ip.7  =left('Interval',16,' ')||left('value',7)||left('Total',9),
      ||left('Min.',6)||left('Avg.',7)||left('Max.',6),
      ||left('Total',8)||left('Min.',7)||left('Avg.',7)||left('Max.',4)
Ip.8  =left('-',80,'-')
   "EXECIO * DISKW rmf03 (STEM Ip.)"
  "ALLOC FILE(rmf04) DA("r7434")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(2,1) CATALOG",
  "REUSE LRECL(90) RECFM(F B) BLKSIZE(27990)"
Mm.1  ='USS Memory map'
Mm.2  ='  '
Mm.3  ='Report produced on',
      ||left(' ',1,' ')||left(date(),11),
      ||left(' ',1,' ')||left('at ',3,' ')||left(time(),10)
Mm.4  ='  '
Mm.5  =left(' ',35,' ')||left('-',6,'-')||left(' Count',7),
      ||left('-',6,'-')||left(' ',9,' '),
      ||left('-',6,'-')||left(' Overruns',10),
      ||left('-',6,'-')
Mm.6  =left(' ',27,' ')||left('Parm',4)
Mm.7  =left('Interval',26,' ')||left('value',8)||left('Total',9),
      ||left('Min.',6)||left('Avg.',7)||left('Max.',6),
      ||left('Total',8)||left('Min.',7)||left('Avg.',7)||left('Max.',4)
Mm.8  =left('-',90,'-')
   "EXECIO * DISKW rmf04 (STEM Mm.)"
  "ALLOC FILE(rmf05) DA("r7435")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(2,1) CATALOG",
```

```
  "REUSE LRECL(9Ø) RECFM(F B) BLKSIZE(2799Ø)"
Lb.1  ='USS Shared library region & queued signals'
Lb.2  ='  '
Lb.3  ='Report produced on',
      ||left(' ',1,' ')||left(date(),11),
      ||left(' ',1,' ')||left('at ',3,' ')||left(time(),1Ø)
Lb.4  ='  '
Lb.5  =left(' ',35,' ')||left('-',6,'-')||left(' Count',7),
      ||left('-',6,'-')||left(' ',9,' '),
      ||left('-',6,'-')||left(' Overruns',1Ø),
      ||left('-',6,'-')
Lb.6  =left(' ',25,' ')||left('Parm',4)
Lb.7  =left('Interval',24,' ')||left('value',8)||left('Total',9),
      ||left('Min.',6)||left('Avg.',7)||left('Max.',6),
      ||left('Total',8)||left('Min.',7)||left('Avg.',7)||left('Max.',4)
Lb.8  =left('-',9Ø,'-')
   "EXECIO * DISKW rmfØ5 (STEM Lb.)"
/* Part 2: Read  SMF record type 74                                  */
/*         process selected sections & write out to report file      */
 'EXECIO * DISKR SMF74 (STEM x. FINIS'
    do i = 1  to x.Ø
   smftype  = c2d(SUBSTR(x.i,2,1))              /* SMF record type */
   smfstype = c2d(SUBSTR(x.i,19,2))             /* Record subtype  */
   smfdate  = SUBSTR(c2x(SUBSTR(x.i,7,4)),3,5)  /* Unpack SMF date */
   smftime  = smf(c2d(SUBSTR(x.i,3,4)))         /* Decode SMF time */
   sid      = SUBSTR(x.i,11,4)                  /* SID             */
   ssi      = SUBSTR(x.i,15,4)                  /* Subsystem ID.   */
   pps = c2d(SUBSTR(x.i,25,4))       /*  Offset to RMF product sect.*/
   prl = c2d(SUBSTR(x.i,29,2))       /*  Lenght of RMF product sect.*/
   prn = c2d(SUBSTR(x.i,31,2))       /*  Number of RMF product sect.*/
 IF pps  <> Ø AND prn  <> Ø Then do
    pps =pps -3
    prd =     SUBSTR(x.i,pps+2,8)
    sam = c2d(SUBSTR(x.i,pps+24,4))
    mvs =     SUBSTR(x.i,pps+4Ø,8)
    ptn = c2d(SUBSTR(x.i,pps+5Ø,1))
    srl = c2d(SUBSTR(x.i,pps+51,1))
    oil = c2d(SUBSTR(x.i,pps+76,2))
    syn = c2d(SUBSTR(x.i,pps+78,2))
    gie = st(c2x(SUBSTR(x.i,pps+8Ø,8)))
    xnm =     SUBSTR(x.i,pps+88,8)
    snm =     SUBSTR(x.i,pps+96,8)
  END
  IF smfstype = '6'  Then call RMF746
/* Part 2a: Read SMF record type 74.3 (Kernel record)                */
  IF smfstype = '3'  Then
  Do
   po = c2d(SUBSTR(x.i,33,4))      /*  Offset to USS data section   */
   pl = c2d(SUBSTR(x.i,37,2))      /*  Length of USS data section   */
   pn = c2d(SUBSTR(x.i,39,2))      /*  Number of USS data sections  */
```

```
 IF po  <> 0 AND pn  <> 0 Then do
    po =po -3
  cycu  = c2d(substr(x.i,po,4))          /*      Cycle units elapsed*/
  cyct  = c2d(substr(x.i,po+4,4))        /*Cycle time-Mon.III option*/
  flg   = c2x(substr(x.i,po+8,4))        /*         Processing flags*/
 SELECT
  when flg = '80000000' then procflag ='OMVS terminated/reinstated'
  when flg = '40000000' then procflag ='MAXPROCSYS    change'
  when flg = '20000000' then procflag ='MAXUIDS       change'
  when flg = '10000000' then procflag ='MAXPROCUSER   change'
  when flg = '08000000' then procflag ='IPCSMSGNIDS   change'
  when flg = '04000000' then procflag ='IPCSEMNIDS    change'
  when flg = '02000000' then procflag ='IPCSHMNIDS    change'
  when flg = '01000000' then procflag ='IPCSHMSPACES  change'
  when flg = '00800000' then procflag ='MAXMMAPAREA   change'
  when flg = '00400000' then procflag ='MAXSHAREPAGES change'
  when flg = '00200000' then procflag ='SHRLIBRGNSIZE change'
  when flg = '00100000' then procflag ='MAXQUEUEDSIGS change'
  otherwise nop
 END
/* Part 2a1: Read Kernel System call activity portion              */
  sysc  = flt(c2x(substr(x.i,po+12,4)))  /*    Tot. USS system calls*/
  scmn  = c2d(substr(x.i,po+16,4))       /*    Min. USS system calls*/
  scmx  = c2d(substr(x.i,po+20,4))       /*    Max. USS system calls*/
  avgcall = sysc / sam                   /*    Avg. USS system calls*/
  cpu   = flt(c2x(substr(x.i,po+24,4)))  /* Tot. CPU time (syscalls)*/
  ctmn  = c2d(substr(x.i,po+28,4))       /* Min. CPU time (syscalls)*/
  ctmx  = c2d(substr(x.i,po+32,4))       /* Max. CPU time (syscalls)*/
  avgcpu  = cpu  / sam                   /* Avg. CPU time - s.calls */
/* Part 2a2: Read Kernel Process activity portion                  */
  opr  = flt(c2x(substr(x.i,po+36,4)))  /*Tot. overruns - MAXPROCSYS*/
  aopr = avg(opr)                       /*Avg. overruns - MAXPROCSYS*/
  opmn = c2d(substr(x.i,po+40,4))       /*Min. overruns - MAXPROCSYS*/
  opmx = c2d(substr(x.i,po+44,4))      /*Max. overuruns - MAXPROCSYS*/
  ous  = flt(c2x(substr(x.i,po+48,4)))   /*  Tot. overruns - MAXUIDS*/
  aous = avg(ous)                        /*  Avg. overruns - MAXUIDS*/
  oumn = c2d(substr(x.i,po+52,4))        /*  Min. overruns - MAXUIDS*/
  oumx = c2d(substr(x.i,po+56,4))        /*  Max. overruns - MAXUIDS*/
  opru = flt(c2x(substr(x.i,po+60,4)))  /*Tot.overruns - MAXPROCUSER*/
  aopru = avg(opru)                    /*Avg. overruns - MAXPROCUSER*/
  ormn  = c2d(substr(x.i,po+64,4))     /*Min. overruns - MAXPROCUSER*/
  ormx  = c2d(substr(x.i,po+68,4))     /*Max. overruns - MAXPROCUSER*/
  maxp  = c2d(substr(x.i,po+72,2))     /*Max. USS  processes        */
  maxu  = c2d(substr(x.i,po+74,2))     /*Max. USS  users            */
  mxpu  = c2d(substr(x.i,po+76,2))     /*Max. USS  processes/user   */
  curp  = flt(c2x(substr(x.i,po+80,4)))   /*  Tot. USS  processes   */
  avgcurp = curp / sam                    /*  Avg. USS  processes   */
  cpmn  = c2d(substr(x.i,po+84,2))        /*  Min. USS  processes   */
  cpmx  = c2d(substr(x.i,po+86,2))        /*  Max. USS  processes   */
  curu  = flt(c2x(substr(x.i,po+88,4)))   /* Tot.USS users/interval */
```

```
  avgcuru = curu / sam                    /* Tot.USS users/interval */
  cumn  = c2d(substr(x.i,po+92,2))        /* Min.USS users/cycle    */
  cumx  = c2d(substr(x.i,po+94,2))        /* Max.USS users/cycle    */
/* Part 2a3: Read Kernel Inter-process communication portion        */
  mmsg = c2d(substr(x.i,po+96,4))        /*          Max.IPCMSGNIDS */
  msem = c2d(substr(x.i,po+100,4))       /*          Max.IPCSEMNIDS */
  mshm = c2d(substr(x.i,po+104,4))       /*          Max.IPCSHMNIDS */
  mspg = c2d(substr(x.i,po+108,4))       /*          Max.IPCSHMPAGES*/
  cmsg = flt(c2x(substr(x.i,po+112,4)))  /* Tot. IPCMSGNIDS/interval*/
  acmsg = avg(cmsg)                      /* Avg. IPCMSGNIDS/interval*/
  cmmn = c2d(substr(x.i,po+116,4))       /*    Min. IPCMSGNIDS/cycle*/
  cmmx = c2d(substr(x.i,po+120,4))       /*    Max. IPCMSGNIDS/cycle*/
  csem = flt(c2x(substr(x.i,po+124,4)))  /* Tot. IPCSEMNIDS/interval*/
  acsem = avg(csem)                      /* Avg. IPCSEMNIDS/interval*/
  csmn = c2d(substr(x.i,po+128,4))       /*    Min. IPCSEMNIDS/cycle*/
  csmx = c2d(substr(x.i,po+132,4))       /*    Max. IPCSEMNIDS/cycle*/
  cshm = flt(c2x(substr(x.i,po+136,4)))  /*  Tot.IPCSHMNIDS/interval*/
  acshm = avg(cshm)                      /*  Avg.IPCSHMNIDS/interval*/
  chmn = c2d(substr(x.i,po+140,4))       /*     Min.IPCSHMNIDS/cycle*/
  chmx = c2d(substr(x.i,po+144,4))       /*     Max.IPCSHMNIDS/cycle*/
  cspg = flt(c2x(substr(x.i,po+148,4)))  /* Tot.IPCSHMPAGES/interval*/
  acspg = avg(cspg)                      /* Avg.IPCSHMPAGES/interval*/
  cgmn = c2d(substr(x.i,po+152,4))       /*    Min.IPCSHMPAGES/cycle*/
  cgmx = c2d(substr(x.i,po+156,4))       /*    Max.IPCSHMPAGES/cycle*/
  omsg = flt(c2x(substr(x.i,po+160,4)))  /*  Tot.over IPCMSGNIDS/int*/
  aomsg = avg(omsg)                      /*  Avg.over IPCMSGNIDS/int*/
  ommn = c2d(substr(x.i,po+164,4))       /*Min.over.IPCMSGNIDS/cycle*/
  ommx = c2d(substr(x.i,po+168,4))       /*Max.over IPCMSGNIDS/cycle*/
  osem = flt(c2x(substr(x.i,po+172,4)))  /*Tot.over IPCSEMNIDS/int  */
  aosem = avg(osem)                      /*Avg.over IPCSEMNIDS/int  */
  osmn = c2d(substr(x.i,po+176,4))       /*Min.over IPCSEMNIDS/cycle*/
  osmx = c2d(substr(x.i,po+180,4))       /*Max.over IPCSEMNIDS/cycle*/
  oshm = flt(c2x(substr(x.i,po+184,4)))  /*Tot.over IPCSHMNIDS/int  */
  aoshm= avg(oshm)                       /*Avg.over IPCSHMNIDS/int  */
  ohmn = c2d(substr(x.i,po+188,4))       /*Min.over IPCSHMNIDS/cycle*/
  ohmx = c2d(substr(x.i,po+192,4))       /*Max.over IPCSHMNIDS/cycle*/
  ospg = flt(c2x(substr(x.i,po+196,4)))  /*Tot.over IPCSHMPAGES/int */
  aospg = avg(ospg)                      /*Avg.over IPCSHMPAGES/int */
  ogmn = c2d(substr(x.i,po+200,4))      /*Min.over IPCSHMPAGES/cycle*/
  ogmx = c2d(substr(x.i,po+204,4))      /*Max.over IPCSHMPAGES/cycle*/
/* Part 2a4: Read Kernel Memory map portion                         */
  mmap  = c2d(substr(x.i,po+208,4))      /*     Max. MAXMMAPAREA    */
  cmap  = flt(c2x(substr(x.i,po+212,4))) /*     Tot. MAXMMAPAREA/int*/
  acmap = avg(cmap)                      /*     Avg. MAXMMAPAREA/int*/
  camn  = c2d(substr(x.i,po+216,4))      /*   Min. MAXMMAPAREA/cycle*/
  camx  = c2d(substr(x.i,po+220,4))      /*   Max. MAXMMAPAREA/cycle*/
  omap  = flt(c2x(substr(x.i,po+224,4)))  /*Tot.over MAXMMAPAREA/int*/
  aomap = avg(omap)                       /*Avg.over MAXMMAPAREA/int*/
  oamn  = c2d(substr(x.i,po+228,4))     /*Min.over MAXMMAPAREA/cycle*/
  oamx  = c2d(substr(x.i,po+232,4))     /*Max.over MAXMMAPAREA/cycle*/
```

```
  mpag  = c2d(substr(x.i,po+236,4))      /*       Max. MAXSHAREPAGES*/
  cpag  = flt(c2x(substr(x.i,po+240,4))) /*       Tot. MAXSHAREPAGES*/
  acpag = avg(cpag)                      /*       Avg. MAXSHAREPAGES*/
  cxmn  = c2d(substr(x.i,po+244,4))      /* Min. MAXSHAREPAGES/cycle*/
  cxmx  = c2d(substr(x.i,po+248,4))      /* Max. MAXSHAREPAGES/cycle*/
  opag  = flt(c2x(substr(x.i,po+252,4))) /*Totover MAXSHAREPAGES/int*/
  aopag = avg(opag)                      /*Avgover MAXSHAREPAGES/int*/
  oxmn  = c2d(substr(x.i,po+256,4))   /*Min.over MAXSHAREPAGES/cycle*/
  oxmx  = c2d(substr(x.i,po+260,4))   /*Max.over MAXSHAREPAGES/cycle*/
/* Part 2a5: Read Kernel Shared libs and queued signals             */
  mslr = c2d(substr(x.i,po+264,4))       /*       Max.SHRLIBRGNSIZE */
  cslr = flt(c2x(substr(x.i,po+268,4)))  /*   Tot.SHRLIBRGNSIZE/int */
  acslr = avg(cslr)                      /*   Avg.SHRLIBRGNSIZE/int */
  clmn = c2d(substr(x.i,po+272,4))       /* Min.SHRLIBRGNSIZE/cycle */
  clmx = c2d(substr(x.i,po+276,4))       /* Max.SHRLIBRGNSIZE/cycle */
  oslr = flt(c2x(substr(x.i,po+280,4)))  /*   Totover SHRLIBRGNSIZE */
  aoslr = avg(oslr)                      /*   Avgover SHRLIBRGNSIZE */
  olmn = c2d(substr(x.i,po+284,4))       /*   Minover SHRLIBRGNSIZE */
  olmx = c2d(substr(x.i,po+288,4))       /*   Maxover SHRLIBRGNSIZE */
  mqds = c2d(substr(x.i,po+292,4))       /*           MAXQUEUEDSIGS */
  oqds = flt(c2x(substr(x.i,po+296,4)))  /*Totover MAXQUEUEDSIGS/int*/
  aoqds = avg(oqds)                      /*Avgover MAXQUEUEDSIGS/int*/
  oqmn = c2d(substr(x.i,po+300,4))    /* Minover MAXQUEUEDSIGS/cycle*/
  oqmx = c2d(substr(x.i,po+304,4))    /* Maxover MAXQUEUEDSIGS/cycle*/
  avg1= trunc(avgcurp,4)
  avg2= trunc(avgcuru,4)
  avgs= trunc(avgcall,2)
  avgc= trunc(avgcpu,4)
 interval = right(Date('N',smfdate,'J'),11)  left(smftime,8)
  PUSH interval
    "EXECIO 1 DISKW rmf02"
  PUSH interval
    "EXECIO 1 DISKW rmf03"
  PUSH interval
    "EXECIO 1 DISKW rmf04"
  PUSH interval
    "EXECIO 1 DISKW rmf05"
 syscall = right(Date('N',smfdate,'J'),11)  left(smftime,8),
           right(SYSC,7) right(SCMN,5) right(avgs,5) right(SCMX,6),
           right(CPU,6)  right(CTMN,6) right(avgc,6) right(CTMX,5)
       PUSH syscall
    "EXECIO 1 DISKW rmf01"
 process = left('Processes:',10) right(maxp,6) right(curp,6),
           right(cpmn,6)         right(avg1,6) right(cpmx,6),
           right(opr,6)          right(opmn,6) right(aopr,6),
           right(opmx,6)
       PUSH process
    "EXECIO 1 DISKW rmf02"
 users = left('Users    :',10) right(maxu,6) right(curu,6),
         right(cumn,6)         right(avg2,6) right(cumx,6),
```

```
        right(ous,6)          right(oumn,6) right(aous,6),
        right(oumx,6)
      PUSH users
   "EXECIO 1 DISKW rmf02"
procuser =left('Proc/User:',10) right(mxpu,6) left(' ',27,' '),
          right(opru,6)         right(ormn,6),
          right(aopru,6)        right(ormx,6)
      PUSH procuser
   "EXECIO 1 DISKW rmf02"
 line = left(' ',3,' ')
    PUSH line
   "EXECIO 1 DISKW rmf02"
msgid =left('Message    ID:',14) right(mmsg,6),
  right(cmsg,6) right(cmmn,6) right(acmsg,6) right(cmmx,6),
  right(omsg,6) right(ommn,6) right(aomsg,6) right(ommx,6)
  PUSH msgid
  "EXECIO 1 DISKW rmf03"
semaphore =left('Semaphor   ID:',14) right(msem,6),
  right(csem,6) right(csmn,6) right(acsem,6) right(camx,6),
  right(osem,6) right(osmn,6) right(aosem,6) right(osmx,6)
  PUSH semaphore
  "EXECIO 1 DISKW rmf03"
sharemem =left('Shared mem.ID:',14) right(mshm,6),
  right(cshm,6) right(chmn,6) right(acshm,6) right(chmx,6),
  right(oshm,6) right(ohmn,6) right(aoshm,6) right(ohmx,6)
  PUSH sharemem
  "EXECIO 1 DISKW rmf03"
sharepgs =left('Shared mem.PG:',14) right(mspg,6),
  right(cspg,6) right(cgmn,6) right(acspg,6) right(cgmx,6),
  right(ospg,6) right(ogmn,6) right(aospg,6) right(ogmx,6)
  PUSH sharepgs
  "EXECIO 1 DISKW rmf03"
 line = left(' ',3,' ')
    PUSH line
   "EXECIO 1 DISKW rmf03"
memmap =left('Memory map storage pages:',25) right(mmap,6),
  right(cmap,6) right(camn,6) right(acmap,6) right(camx,6),
  right(omap,6) right(oamn,6) right(aomap,6) right(oamx,6)
  PUSH memmap
  "EXECIO 1 DISKW rmf04"
sharestorp =left('Shared storage pages    :',25) right(mpag,6),
  right(cpag,6) right(cxmn,6) right(acpag,6) right(cxmx,6),
  right(opag,6) right(oxmn,6) right(aopag,6) right(oxmx,6)
  PUSH sharestorp
  "EXECIO 1 DISKW rmf04"
 line = left(' ',3,' ')
    PUSH line
   "EXECIO 1 DISKW rmf04"
sharelib =left('Shared library region:',22) right(mslr,6),
  right(cslr,6) right(clmn,6) right(acslr,7) right(clmx,6),
```

```
   right(oslr,6) right(olmn,6) right(aoslr,7) right(olmx,6)
   PUSH sharelib
   "EXECIO 1 DISKW rmfØ5"
 signal =left('Queued signals       :',22) right(mqds,6),
   right(' ',28,' '),
   right(oqds,6) right(oqmn,6) right(aoqds,7) right(oqmx,6)
   PUSH signal
   "EXECIO 1 DISKW rmfØ5"
  line = left(' ',3,' ')
     PUSH line
    "EXECIO 1 DISKW rmfØ5"
     END
   end
  end
    "EXECIO Ø DISKW rmfØ1(FINIS "
    "EXECIO Ø DISKW rmfØ2(FINIS "
    "EXECIO Ø DISKW rmfØ3(FINIS "
    "EXECIO Ø DISKW rmfØ4(FINIS "
    "EXECIO Ø DISKW rmfØ5(FINIS "
    "EXECIO Ø DISKW global(FINIS "
    "EXECIO Ø DISKW buffer(FINIS "
    "EXECIO Ø DISKW hfsfil(FINIS "
 say
 say 'HFS Global statistics report dsn  . . . . . .:'r74gl
 say 'Buffer pool statistics report dsn . . . . . .:'r74bf
 say 'HFS File system statistics report dsn . . . . .:'r74hf
 say 'System call activity report dsn . . . . . .:'r7431
 say 'Process activity report dsn . . . . . .:'r7432
 say 'Inter Process Communication report dsn  . . . . .:'r7433
 say 'Memory map report dsn . . . . . .:'r7434
 say 'Shared library region & queued signals report dsn:'r7435
 say
   "free FILE(SMF74 global buffer hfsfil)"
   "free FILE(rmfØ1 rmfØ2 rmfØ3 rmfØ4 rmfØ5)"
 exit
RMF746:
/* Part 2b: Read SMF record type 74.6 (HFS record)                  */
   do = c2d(SUBSTR(x.i,33,4))      /*  Offset to HFS global data sec.*/
   dl = c2d(SUBSTR(x.i,37,2))      /*  Length of HFS global data sec.*/
   dn = c2d(SUBSTR(x.i,39,2))      /*  Number of HFS global data sec.*/
   bo = c2d(SUBSTR(x.i,41,4))      /*       Offset to HFS buffer sec.*/
   bl = c2d(SUBSTR(x.i,45,2))      /*       Length of HFS buffer sec.*/
   bn = c2d(SUBSTR(x.i,47,2))      /*       Number of HFS buffer sec.*/
   fo = c2d(SUBSTR(x.i,49,4))      /*  Offset to HFS file system sec.*/
   fl = c2d(SUBSTR(x.i,53,2))      /*  Length of HFS file system sec.*/
   fn = c2d(SUBSTR(x.i,55,2))      /*  Number of HFS file system sec.*/
/* Part 2b1: Read SMF record type 74.6 (HFS Section)                */
 IF do  <> Ø AND dn  <> Ø Then do
  do =do -3
  gmxv = c2d(SUBSTR(x.i,do,4))           /*      VIRTUAL(MAX) -   MB)*/
```

```
  gusv = c2d(SUBSTR(x.i,do+4,4)) /* pages of VIR assigned to IO buff.*/
  gmnf = c2d(SUBSTR(x.i,do+8,4))         /*         FIXED(MIN) -  MB */
  gusf = c2d(SUBSTR(x.i,do+12,4))        /*   permanently fixed pages*/
  gmc  = flt(c2x(SUBSTR(x.i,do+16,8)))   /*          metadata - cache*/
  gmnc = flt(c2x(SUBSTR(x.i,do+24,8)))   /*           metadata - dasd*/
  g1c  = flt(c2x(SUBSTR(x.i,do+32,8)))   /*        first page - cache*/
  g1nc = flt(c2x(SUBSTR(x.i,do+40,8)))   /*         first page - dasd*/
  glrc = c2d(SUBSTR(x.i,do+48,4))     /*Ret. code DisplayBufferLimits*/
  glrs = c2d(SUBSTR(x.i,do+52,4))     /*Reas.code DisplayBufferLimits*/
  gsrc = c2d(SUBSTR(x.i,do+56,4))     /*Ret. code DisplayGlobalStats */
  gsrs = c2d(SUBSTR(x.i,do+60,4))     /*Reas.code DisplayGlobalStats */
  gsfl =     SUBSTR(x.i,do+64,1)          /* Global data status flags*/
    SELECT
     when gsfl ='10000000'b then ffl='Kernel not ready    '
     when gsfl ='01000000'b then ffl='No buffer limit data'
     when gsfl ='00100000'b then ffl='No global data      '
     when gsfl ='00010000'b then ffl='Partial global data '
     otherwise  ffl='Reserved'
    END
  mbv  = (gusv * 4) / 1024
  mbuse= trunc(mbv,2)
 END
  glb = right(Date('N',smfdate,'J'),11)  left(smftime,8),
        right(gmxv,8)   right(mbuse,8)   right(gmnf,8),
        right(gusf,8)   right(g1c,8)     right(g1nc,8),
        right(gmc,8)    right(gmnc,8)
      PUSH glb
   "EXECIO 1 DISKW global"
/* Part 2b2: Read SMF record type 74.6 (Buffer section)              */
 IF bo  <> 0 AND bn  <> 0 Then do
    bo =bo -3
    do k = 0 to bn  -1
       num = k + 1                            /*          pool number*/
       incr  = (bo + (k*bl))
       gsb.k = c2d(SUBSTR(x.i,incr,2))        /*size of buffers in bp*/
       gnds.k= c2d(SUBSTR(x.i,incr+2,2))      /*   data spaces for bp*/
       gsbp.k= c2d(SUBSTR(x.i,incr+4,4))      /*    poll size - pages*/
       gsbf.k= c2d(SUBSTR(x.i,incr+8,4))      /*  fixed buffers in bp*/
       gbf.k = flt(c2x(SUBSTR(x.i,incr+16,8)))    /* buffer was fixed*/
       gbnf.k= flt(c2x(SUBSTR(x.i,incr+24,8)))    /* buffer not fixed*/
       nb.k= gsbp.k /gsb.k                        /*   no. of buffers*/
       mb.k= (gsbp.k * 4) / 1024                  /*   pool size - MB*/
       mm.k=trunc(mb.k,2)
       tot.k=gbf.k+gbnf.k                         /*  total I/O count*/
   SELECT
    when k  = 0 then
    inter = right(Date('N',smfdate,'J'),11)||left(' ',1,' '),
            ||left(smftime,8)
    otherwise
    inter =left(' ',20,' ')
```

```
   END
  buf.k= inter  right(num,4) right(nb.k,5) right(gsb.k,8),
         right(gsbp.k,8)     right(mm.k,6) right(gsbf.k,5),
         right(gnds.k,8)     right(tot.k,6),
         right(gbf.k,6)      right(gbnf.k,6)
      PUSH buf.k
   "EXECIO 1 DISKW buffer"
    end
   line = left(' ',3,' ')
      PUSH line
   "EXECIO 1 DISKW buffer"
 END
/* Part 2b3: Read SMF record type 74.6 (HFS file section)           */
 IF fo  <> Ø AND fn  <> Ø Then do
  fo =fo -3
   do j = Ø to fn  -1
    inc  = (fo + (j*fl))
    fsnl.j= c2d(SUBSTR(x.i,inc+44,1))
    fsnm.j=     SUBSTR(x.i,inc,fsnl.j)     /*        File system name*/
    fsfØ.j=     SUBSTR(x.i,inc+45,1)       /*       File Status flags*/
     SELECT
      when fsfØ.j ='1ØØØØØØØ'b then fflag='No HFS file system stat.'
      when fsfØ.j ='Ø1ØØØØØØ'b then fflag='Mount time changed'
      when fsfØ.j ='ØØ1ØØØØØ'b then fflag='File system now mounted'
     otherwise  fflag='Reserved'
     END
    fctm.j= st(c2x(SUBSTR(x.i,inc+48,8)))  /* RMF TOD timestamp STCK */
    fmtm.j= st(c2x(SUBSTR(x.i,inc+56,8)))  /*   Mount timestamp STCK */
    fsf.j = c2d(SUBSTR(x.i,inc+64,4))      /*    Size of file system */
    fpf.j = c2d(SUBSTR(x.i,inc+68,4))      /*   Number of data pages */
    fpd.j = c2d(SUBSTR(x.i,inc+72,4))      /*   Attr.directory pages */
    fpc.j = c2d(SUBSTR(x.i,inc+76,4))      /*Data buffer pages cached*/
    fsfi.j= flt(c2x(SUBSTR(x.i,inc+8Ø,8)))   /*       Sequential I/O */
    frfi.j= flt(c2x(SUBSTR(x.i,inc+88,8)))   /*           Random I/O */
    fmc.j = flt(c2x(SUBSTR(x.i,inc+96,8)))   /*  Metadata from cache */
    fmnc.j= flt(c2x(SUBSTR(x.i,inc+1Ø4,8)))  /*   Metadata from DASD */
    f1c.j = flt(c2x(SUBSTR(x.i,inc+112,8)))  /*First page from cache */
    f1nc.j= flt(c2x(SUBSTR(x.i,inc+12Ø,8)))  /* First page from DASD */
    fint.j= flt(c2x(SUBSTR(x.i,inc+128,8)))  /*       Index new tops */
    fis.j = flt(c2x(SUBSTR(x.i,inc+136,8)))  /*         Index splits */
    fij.j = flt(c2x(SUBSTR(x.i,inc+144,8)))  /*          Index joins */
    firh.j= flt(c2x(SUBSTR(x.i,inc+152,8)))  /* Index page read hits */
    firm.j= flt(c2x(SUBSTR(x.i,inc+16Ø,8)))  /*  Index page read miss*/
    fiwh.j= flt(c2x(SUBSTR(x.i,inc+168,8)))  /* Index page write hits*/
    fiwm.j= flt(c2x(SUBSTR(x.i,inc+176,8)))  /* Index page write miss*/
    fsrc.j= c2d(SUBSTR(x.i,inc+184,4))       /* Ret.code Disp.FSStats*/
    fsrs.j= c2d(SUBSTR(x.i,inc+184,4))       /* Reas.codeDisp.FSStats*/
   SELECT
    when j  = Ø then
    inter = right(Date('N',smfdate,'J'),11)||left(' ',1,' '),
```

```
            ||left(smftime,8)
    otherwise
    inter =left(' ',2Ø,' ')
   END
  fil.j= inter  left(fsnm.j,3Ø) right(fmtm.j,25),
         right(fsf.j,8)  right(fpf.j,8)  right(fpd.j,8),
         right(fpc.j,8)  right(fsfi.j,8) right(frfi.j,8),
         right(f1c.j,8)  right(f1nc.j,8) right(fmc.j,8),
         right(fmnc.j,8) right(firh.j,8) right(firm.j,8),
         right(fiwh.j,8) right(fiwm.j,8) right(fint.j,8),
         right(fis.j,8)  right(fij.j,8)
      PUSH fil.j
   "EXECIO 1 DISKW hfsfil"
     end
   line = left(' ',3,' ')
      PUSH line
   "EXECIO 1 DISKW hfsfil"
 END
 return
SMF: procedure
/* REXX - convert an SMF time  */
arg time
    time1    = time % 1ØØ
    hh       = time1 % 36ØØ
    hh       = RIGHT("Ø"||hh,2)
    mm       = (time1 % 6Ø) - (hh * 6Ø)
    mm       = RIGHT("Ø"||mm,2)
    ss       = time1 - (hh * 36ØØ) - (mm * 6Ø)
    ss       = RIGHT("Ø"||ss,2)
    otime  = hh||":"||mm||":"||ss             /* Compose SMF time*/
    return otime
FLT: PROCEDURE
/* This REXX EXEC computes a floating-point's value */
arg  float
float = X2C(float)     /* convert float to a hexadecimal string */
float_size = LENGTH(float)                    /* size in Bytes  */
Byte_Ø = SUBSTR(float, 1, 1)
 select
   when BITAND(Byte_Ø, '8Ø'x) == 'ØØ'x then sign = '+'
   when BITAND(Byte_Ø, '8Ø'x) == '8Ø'x then sign = '-'
 end
exponent = C2D(BITAND(Byte_Ø, '7F'x)) - 64
fraction = Ø
power    = -1
do i = 2 to float_size
   if i = 9 then iterate              /* skip bits 64-71  */
   Next_Byte   = C2D(SUBSTR(float, i, 1))
   left_Digit  = Next_Byte %  16
   fraction    = fraction + left_Digit * 16**power
   right_Digit = Next_Byte // 16
```

```
   power       = power - 1
   fraction    = fraction + right_Digit * 16**power
   power       = power - 1
end
interpret 'value =' sign ( fraction * 16 ** exponent )
val1=trunc(value,Ø)
return val1
ST: PROCEDURE
/* STCK timestamp format converted               */
/* The BLSUXTOD proc is described in "z/OS       */
/* V1R3 MVS IPCS Customization"                  */
arg todtime
If todtime    <> 'ØØØØØØØØØØØØØØØØØØØØ' Then
  Do
    TOD_Value = X2C(todtime)
    Returned_Date = '-------------------------'
    address LINKPGM "BLSUXTOD TOD_Value Returned_Date"
  End
Else
  Returned_Date = ''
Return Returned_Date
AVG: procedure expose sam
/* REXX - calculate avg. value*/
arg var
   SELECT
    when var  ¬=  Ø Then do
       a  = var / sam
       b  = trunc(a,4)
      end
     otherwise  b = Ø.ØØØ
   END
 return b
```

MATCH EXEC

```
/* REXX EXEC (MATCH) match & merge records: SMF 3Ø.4 and 92.11    */
  ADDRESS TSO
/* Part 1: Handle file allocation & dataset existence and         */
/*         print report header and labels                          */
  userid=SYSVAR(SYSUID)
  r3Øuu  =userid||'.mat3Øa.out'         /* Portion of SMF 3Ø.4 rec.*/
  r92uu  =userid||'.mat92a.out'         /* Portion of SMF 92.11 rec*/
  r3Ø92  =userid||'.match.rep'          /* Match & Merge Report    */
  x = MSG('OFF')
  if SYSDSN(r3Ø92) = 'OK'
  Then "DELETE "r3Ø92" PURGE"
  "ALLOC FILE(list1) DA("r3Øuu")" shr
  "ALLOC FILE(list2) DA("r92uu")" shr
  "ALLOC FILE(outdd) DA("r3Ø92")",
```

```
     " UNIT(SYSALLDA) NEW TRACKS SPACE(60,30) CATALOG",
     " REUSE LRECL(240) RECFM(F B) BLKSIZE(27840)"
 Mt.1  =left(' ',8,' '),
        ||'Composite I/O Activity Report for USS jobs/tasks',
        ||left(' ',15,' ')
 Mt.2  ='  '
 Mt.3  =left(' ',8,' ')||'Report produced on',
        ||left(' ',1,' ')||left(date(),11),
        ||left(' ',1,' ')||left('at ',3,' ')||left(time(),10)
 Mt.4  =left(' ',181,' ')||left('No.Calls',15),
        ||left('I/O blocks',20)||left('Bytes',5)
 Mt.5  ='  '
 Mt.6  =left(' Date',12)||left('Time',13)||left('Proc.Id',8),
        ||left('Job name',9)||left('Step name',10)||left('Program',9),
        ||left('Path name',65)||left('Open time',13),
        ||left('Close time',13)||left('Opened',13),
        ||left("File #",7)||left('Device',7),
        ||left('Read',7)||left('Write',7),
        ||left("Dir",4)||left('read',5)||left('write',10),
        ||left('read',7)||left('write',6)||left('Record flag',18)
  Mt.7 =left('-',240,'-')
       "EXECIO * DISKW outdd (STEM Mt.)"
/* Part 2: Match & Merge job/task identification (SMF 30.4) to     */
/*         USS close file data (SMF 92.11)                         */
  "EXECIO *  DISKR LIST1   (FINIS STEM l1."
  "EXECIO *  DISKR LIST2   (FINIS STEM l2."
 i = 1
 k = 1
 z = 1
 do while (i <= l1.0) &  (k <= l2.0)
  select
    when substr(l1.i,1,40) = substr(l2.k,1,40) then
      do
        datt  = substr(l1.i,1,5)
        p9.z=substr(l2.k,45,200)
         select
           when z > 1 then  p3=left(' ',59,' ')
           otherwise
        p3=right(Date('N',datt,'J'),11),
           substr(l1.i,63,8),
           substr(l1.i,16,38)
         end
        mr =p3 p9.z
        PUSH MR
         "EXECIO 1 DISKW outdd"
        k = k+1
        z = z+1
      end
     when (substr(l1.i,1,40) < substr(l2.k,1,40)) | (k > l2.0) then
       do           /* record is on LIST1 only  */
```

```
      /* say "i 30    " i substr(l1.i,1,50) */
         i = i+1
        end
      when (substr(l1.i,1,40) > substr(l2.k,1,40)) | (i > l1.0) then
        do              /* record is on LIST2 only   */
        /*   say "92 k    " k substr(l2.k,1,50) */
             k = k+1
             z = 1
          end
       end
   end
   "EXECIO 0 DISKW outdd(FINIS "
say
say 'Composite I/O Activity Report for USS jobs/tasks dsn :'r3092
say
   "free FILE(outdd list1 list2)"
    if SYSDSN(r30uu) = 'OK'
      Then "DELETE "r30uu" PURGE"
    if SYSDSN(r92uu) = 'OK'
      Then "DELETE "r92uu" PURGE"
Exit
```

Mile Pekic
Systems Programmer (Serbia and Montenegro)

Read spool data from a C/C++ program

On many occasions, I have found it necessary from a program environment to access SYSOUT data directly from the JES spool. In the past, this was usually accomplished with an external writer program that would pull SYSOUT data from the JES spool, based on output class and held status. More recently, the SYSOUT API (SAPI) subsystem interface call has made accessing JES SYSOUT data more flexible. The SAPI interface (subsystem function code 79) offers many more dataset selection criteria than the older SYSOUT dataset call (subsystem function code 1) so that access to JES SYSOUT data can now be considered without having to set up special external writers to manage the output.

Traditionally, the access to SYSOUT data, regardless of whether the access is via a SYSOUT dataset call or the SYSOUT API, would be done through an Assembler program that would manage the following:

- Dataset selection
- Dataset dynamic allocation
- Dataset open
- Data read
- Dataset close
- Dataset dynamic unallocation.

This article provides a sample IBM C/C++ program that provides those operations for all JES SYSOUT datasets that match the specified criteria. The only Assembler code (also provided) is a small function (called by the C/C++ program) that returns the SAPI SSS2 control block (which contains the JES block token and the spool dataset name) of the 'next' SYSOUT dataset meeting the selection criteria. Once a block token has been obtained, sufficient information is available to allocate, open, and read JES SYSOUT data.

FUNCTIONAL OVERVIEW

For the supplied example SPOOLRD C/C++ program, the passed program parameters can be used to select JES SYSOUT datasets based on job name and further qualified by optionally specifying additional selection criteria of either job number and/or step name and/or DDname. The program comments provide several example PARM= specifications. The SPOOLRD program sets the SSS2SEL1 and SSS2SEL3 flag fields to indicate the jobs that will be eligible to have their dataset's block tokens returned to the calling program. The SPOOLRD program will further use step name and/or DDname (if it has been supplied) to further restrict whether or not a supplied dataset's block token will be used for accessing SYSOUT data. The SSS2DKPE and

SSS2RNPT flags of the SSS2DSP1 field are set on to indicate that the returned dataset should be kept on spool (SSS2DKPE) and that it should not be returned to this SAPI address space again (SSS2RNPT).

When the SPOOLRD program has properly populated the SAPI SSS2 control block with the required selection criteria information, it calls the GETSSS2() function to locate the next available spool dataset matching the selection criteria. The primary piece of information used by SPOOLRD from the GETSSS2() function call is the block token value for the selected JES spool dataset (although the returned dataset name is also important). SPOOLRD uses the returned block token and dataset name to dynamically allocate the spool dataset (providing that the dataset meets all the other selection criteria including step name and DDname). After the dataset is allocated, it is opened and the content is read, record-by-record, by the SPOOLRD program. In the example program, the matching datasets' output is written to SYSPRINT. Appropriate separators are used to distinguish the output from different spool datasets. When SPOOLRD finishes reading the content of each spool dataset, the dataset is closed and unallocated.

PROGRAM COMPILATION, LINKAGE, AND EXECUTION

The SPOOLRD program can be compiled and prelinked in either C or C++ mode. A reentrant (RENT) compile option is recommended. Be sure to convert [to X'AD' and] to X'BD' in the SPOOLRD C source code before running the compile. An example C compile job is shown below:

```
//PROCS    JCLLIB ORDER=(CBC.SCBCPRC)
//STEP1   EXEC EDCC,CPARM='LIST',
// CPARM2='RENT,NOSEARCH',
// CPARM3='NOMAR,NOSEQ,NOOPT,LANGLVL(EXTENDED),SOURCE,LONGNAME,SSCOMM',
//          INFILE=c.source.pds(SPOOLRD),
//          OUTFILE='object.code.pds(SPOOLRDO),DISP=SHR',
//          SYSLBLK=8000
```

The code below provides a sample prelink job:

```
//PLKED1  EXEC PGM=EDCPRLK,PARM='UPCASE',REGION=2048K
```

```
//SYSMSGS   DD  DSNAME=CEE.SCEEMSGP(EDCPMSGE),DISP=SHR
//SYSLIB    DD  DUMMY
//SYSOBJ    DD  DSN=object.code.pds,DISP=SHR
//SYSMOD    DD  DSN=object.code.pds(SPOOLRD),DISP=SHR
//SYSOUT    DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSIN     DD  *
  INCLUDE SYSOBJ(SPOOLRDO)
```

The *c.source.pds* is the dataset containing the SPOOLRD C program, *object.code.pds* represents the target object code dataset, and *SYSLBLK* should specify the block size of the target object code dataset. GETSSS2 can be assembled with a standard assembly job – be sure to include CEE.SCEEMAC and SYS1.MACLIB in the SYSLIB DD for the assembly.

When the SPOOLRD program has been compiled and prelinked, and the GETSSS2 function has been assembled, create the SPOOLRD load module using the linkedit job shown below:

```
//IEWL      EXEC  PGM=HEWLHØ96,PARM='XREF,LIST,MAP,RENT'
//SYSPRINT DD    SYSOUT=*
//SYSUT1    DD    UNIT=SYSDA,SPACE=(CYL,(2,1))
//OBJECT    DD    DSN=object.code.pds,DISP=SHR
//SYSLIB    DD    DSN=CEE.SCEELKED,DISP=SHR
//SYSLMOD   DD    DSN=auth.load.library,DISP=SHR
//SYSLIN    DD    *
   INCLUDE OBJECT(SPOOLRD)    OBJ MODULE AFTER SPOOLRD PRELINK
   INCLUDE OBJECT(GETSSS2)
   SETCODE AC(1)
   ENTRY   CEESTART
   NAME    SPOOLRD(R)
```

SPOOLRD should now be ready to extract JES spool datasets. Below is sample JCL to invoke SPOOLRD:

```
//SPOOLRD  EXEC PGM=SPOOLRD,
//  PARM='jobname=TESTJOB,jobno=jobØ1234'
//STEPLIB  DD   DSN=auth.load.library,DISP=SHR
//SYSPRINT DD   SYSOUT=*
```

The parameter data passed to SPOOLRD is not case sensitive. All parameter data is converted to upper case by the SPOOLRD program. JOBNAME= is required in the PARM= data. All other selection criteria (JOBNO=, STEPNAME=, DDNAME=) are optional and can be used as required.

CONCLUSION

This program has proved useful. It can be easily modified so you can add your own program parameters to include other dataset selection criteria. Give it a look over and make the changes necessary to meet your own requirements. It should be fun!

GETSSS2

```
*********************************************************************
*   Routine:     GETSSS2                                            *
*   Function:    Use the incoming SSS2 to make a SAPI (SSI 79       *
*                function) call to allow the return of a JES spool  *
*                dataset browse token.                              *
*   Arguments:   address of a SSS2 control block                    *
*                address of a subsystem name return area            *
*   Return:      int - Ø for success, ØØ for failure.               *
*   C usage:     i = GETSSS2(&sss2, &ssname);                       *
*********************************************************************
GETSSS2  CSECT
GETSSS2  AMODE 31
GETSSS2  RMODE 24
         ENTRY GETSSS2
         EDCPRLG BASEREG=R12,DSALEN=WORKLEN
         B     SETBASE                  BYPASS ADDRESS CONSTANT
MODBASE  DC    A(GETSSS2)               ESTABLISH MODULE ADDRESS
SETBASE  EQU   *
         L     R11,MODBASE              GET BASE ADDRESS
         DROP  R12
         LR    R12,R11                  COPY BASE ADDRESS
         LA    R11,4Ø95(,R12)           ESTABLISH SECOND ...
         LA    R11,1(,R11)                 BASE REG ADDRESS
         USING GETSSS2,R12,R11          SET MODULE ADDRESSABILITY
         USING BTOKWORK,R13             ADDRESSABILITY TO TEMP STORAGE
         ST    R1,PARMØ                 SAME INCOMING PARM ADDRESS
         LTR   R1,R1                    PARMS OK?
         BZ    RETNEGØ9                 NO - RETURN -9
         L     R2,Ø(,R1)                GET SSS2 DATA AREA ADDR
         LTR   R2,R2                    PARM1 OK?
         BZ    RETNEGØ9                 NO - RETURN -9
         ST    R2,PARM1                 SAVE SSS2 DATA AREA ADDR
         L     R2,4(,R1)                GET SUBSYSTEM NAME AREA ADDR
         LTR   R2,R2                    PARM2 OK?
         BZ    RETNEGØ9                 NO - RETURN -9
         ST    R2,PARM2                 SAVE SUBSYSTEM NAME AREA ADDR
         L     R4,PARM1                 GET SSS2 DATA AREA ADDRESS
         USING SSS2,R4                  SET ADDRESSABILITY
         L     R15,16                   GET CVT ADDRESS
         L     R15,Ø(,R15)              GET TCB/ASCB AREA ADDRESS
```

```
         L     R15,4(,R15)              GET CURRENT TCB
         L     R15,TCBJSCB-TCB(,R15)    GET JSCB ADDRESS?
         L     R15,JSCBSSIB-IEZJSCB(,R15) GET SSIB ADDRESS?
         MVC   SSNMSAVE(8),=8C' '       INITIALIZE THE AREA
         MVC   SSNMSAVE(4),SSIBSSNM-SSIB(R15) MV SSNAME
         L     R1,PARM2                 GET SUBSYSNAME AREA ADDR
         XC    Ø(8,R1),Ø(R1)            CLEAR THE AREA
         MVC   Ø(4,R1),SSNMSAVE         COPY SUBSYSNAME
         LA    R3,SSOBAREA              GET SSOB ADDRESS
         USING SSOBEGIN,R3              SET ADDRESSABILITY
         XC    SSOB(SSOBHSIZ),SSOB      CLEAR THE SSOB
         MVC   SSOBID(4),=C'SSOB'       SET SSOB ID
         MVC   SSOBFUNC(2),=AL2(SSOBSOU2) SET FUNCTION ID
         MVC   SSOBLEN(2),=AL2(SSOBHSIZ) SET SSOB HEADER SIZE
         ST    R4,SSOBINDV              SET SSS2 ADDRESS
         MODESET MODE=SUP
         LA    R1,SSOBAREA              GET SSOB ADDRESS
         O     R1,=X'8ØØØØØØØ'          TURN ON X'8Ø' BIT
         ST    R1,SSOBPTR               SAVE SSOB PTR
         LA    R1,SSOBPTR               POINT TO SSOB PTR
         IEFSSREQ ,                     MAKE SUBSYSTEM REQUEST
         LR    R9,R15                   SAVE THE RETURN CODE
         MODESET MODE=PROB
         LTR   R9,R9                    SUBSYSTEM REQUEST OK?
         BNZ   SSREQAB                  NO - LET'S END
         XR    R8,R8                    CLEAR R8
         ICM   R8,B'ØØØ1',SSOBRETN+3    GET RETURN CODE
         C     R8,=F'Ø'                 ALL'S WELL?
         BE    NOMSG                    YES - BYPASS MESSAGE
         C     R8,=F'4'                 END OF DATA?
         BE    NOMSG                    YES - BYPASS MESSAGE
         MVC   WTO1WRK(WTO1LEN),WTO1LST COPY WTO MODEL
         MVC   WTO1WRK+4(13),=C'SAPI RC(XXXX)'
         ST    R8,DBL2
         UNPK  DBL1(9),DBL2(5)
         NC    DBL1(8),=8X'ØF'
         TR    DBL1(8),=C'Ø123456789ABCDEF'
         MVC   WTO1WRK+4+8(4),DBL1+4
*   IF IT IS DESIRED TO GET MORE INFORMATION ABOUT A SAPI SSI         *
*   FAILURE, AND YOU WOULD LIKE TO ISSUE A CONSOLE MESSAGE WITH       *
*   MORE INFORMATION, UNCOMMENT THE FOLLOWING CODE LINE CONTAINING    *
*   '*===>'                                                           *
*===>    WTO   MF=(E,WTO1WRK)
NOMSG   EQU   *
         C     R8,=F'44'                SSOBRETN TOO HIGH?
         BH    ERRUNKWN                 YES - UNKNOWN
         B     BRTBL1(R8)               PROCESS ACCORDINGLY
BRTBL1  EQU   *
         B     REQOK                    SSOBRETN=ØØ
         B     REQEODS                  SSOBRETN=Ø4
```

```
         B     REQINVA                  SSOBRETN=Ø8
         B     REQUNAV                  SSOBRETN=12
         B     REQDUPJ                  SSOBRETN=16
         B     REQIDST                  SSOBRETN=2Ø
         B     REQAUTH                  SSOBRETN=24
         B     REQTKNM                  SSOBRETN=28
         B     REQLERR                  SSOBRETN=32
         B     REQICLS                  SSOBRETN=36
         B     REQBDIS                  SSOBRETN=4Ø
         B     REQCLON                  SSOBRETN=44
SSREQAB  EQU   *
         LR    R8,R9
         MVC   WTO1WRK(WTO1LEN),WTO1LST COPY WTO MODEL
         MVC   WTO1WRK+4(17),=C'SAPI RC(XXXX) - 2'
         ST    R8,DBL2
         UNPK  DBL1(9),DBL2(5)
         NC    DBL1(8),=8X'ØF'
         TR    DBL1(8),=C'Ø123456789ABCDEF'
         MVC   WTO1WRK+4+8(4),DBL1+4
*   IF IT IS DESIRED TO GET MORE INFORMATION ABOUT A SAPI SSI         *
*   FAILURE, AND YOU WOULD LIKE TO ISSUE A CONSOLE MESSAGE WITH       *
*   MORE INFORMATION, UNCOMMENT THE FOLLOWING CODE LINE CONTAINING    *
*   '*===>'                                                           *
*===>    WTO   MF=(E,WTO1WRK)
ERRUNKWN EQU  *
REQINVA  EQU  *
REQUNAV  EQU  *
REQDUPJ  EQU  *
REQIDST  EQU  *
REQAUTH  EQU  *
REQTKNM  EQU  *
REQLERR  EQU  *
REQICLS  EQU  *
REQBDIS  EQU  *
REQCLON  EQU  *
         ST    R8,RETCODE
         B     RETURN                   RETURN
REQEODS  EQU   *
         ST    R8,RETCODE               SAVE RETURN CODE
         OI    SSS2MSC1,SSS2CTRL        SET TERMINATION FLAG
         MODESET MODE=SUP
         LA    R1,SSOBPTR               POINT TO SSOB PTR
         IEFSSREQ ,                     MAKE SUBSYSTEM REQUEST
         LR    R9,R15                   SAVE THE RETURN CODE
         MODESET MODE=PROB
         LTR   R9,R9                    IEFSSREQ WAS OK?
         BZ    RETURN                   YES - CONTINUE TERMINATION
         WTO   'GETSSS2  - Thread termination failed'
         B     RETURN                   RETURN
REQOK    EQU   *
```

```
RETURNØØ EQU   *
         MVC   RETCODE(4),=F'Ø'          SET RETURN CODE TO Ø
         B     RETURN                    RETURN
RETNEGØ9 EQU   *
         MVC   RETCODE(4),=F'-9'         SET RETURN CODE TO -9
         B     RETURN                    RETURN
RETURN   EQU   *
         L     R15,RETCODE               LOAD RETURN CODE
         EDCEPIL
*   CONSTANTS                                                        *
WTO1LST  WTO   '                                                      X
                                                                     ',X
               ROUTCDE=(1),DESC=(6),MF=L
WTO1LEN  EQU   *-WTO1LST
         LTORG
BTOKWORK EDCDSAD
WORKFLDS DS    CL8Ø
RETCODE  DS    F
PARMØ    DS    F                         INCOMING PARM ADDRESS
PARM1    DS    F                         ADDR OF PARM1 (SSS2 BUFF ADDR)
PARM2    DS    F                         ADDR OF PARM2 (SSNM BUFF ADDR)
WTO1WRK  DS    CL(WTO1LEN)
SSNMSAVE DS    CL8
DBL1     DS    2D
DBL2     DS    2D
SSOBAREA DS    ØD,CL(SSOBHSIZ)
SSOBPTR  DS    F
WORKLEN  EQU   *-BTOKWORK
         PRINT NOGEN
         CVT   DSECT=YES
         IEFSSOBH ,
         IAZSSS2 DSECT=YES
         IEFJESCT TYPE=DSECT
         IKJTCB ,
         IEZJSCB ,
         IEFJSSIB ,
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
```

```
R14      EQU   14
R15      EQU   15
         END
```

SPOOLRD.C

```
/*
 * The SPOOLRD program is designed to read SYSOUT data that resides on
 * JES spool (either JES2 or JES3).  The SYSOUT data to be read is
 * determined by PARM data passed to the program.  At a minimum, the
 * program expects to see a JOBNAME= specified in the PARM data.  For
 * example:
 * //SPOOLRD  EXEC PGM=SPOOLRD,PARM='JOBNAME=TESTJB'
 * In the above example, SPOOLRD would output (to the SYSPRINT DD) the
 * spool resident output for every job currently existing on JES spool
 * with a jobname of TESTJB.
 * Optionally, you can extend the PARM data to include
 * JOBNO=, and/or STEPNAME=, and/or DDNAME=
 * keywords to restrict the returned data.  Here are some example
 * PARMs:
 *    PARM='JOBNAME=TESTJB,JOBNO=JOBØ1534,STEPNAME=STEP1,DDNAME=SYSUT1'
 *    PARM='JOBNAME=MYJOB,STEPNAME=LIST'
 *    PARM='JOBNAME=BACKUP,DDNAME=SYSPRINT'
 *    PARM='JOBNAME=MYJOB,STEPNAME=LIST,DDNAME=OUTPUT'
 *    PARM='JOBNAME=TESTJB,JOBNO=JOBØ2367'
 * The first example will list the spool SYSOUT data for a job named
 * TESTJB that has a job, stc, or tsu number of Ø1534, in step(s)
 * named STEP1, and with a DDname of SYSUT1.
 * The second example will list all spool SYSOUT data for every job
 * named MYJOB but will restrict the listed data to only those steps
 * with a name of LIST.
 * The third example will list spool SYSOUT data for all steps that
 * have a DDname of SYSPRINT for every job with a jobname of BACKUP.
 * The fourth example will list spool SYSOUT data for DDname OUTPUT
 * from steps with a stepname of LIST for every job with a jobname of
 * MYJOB.
 * The fifth example will list spool SYSOUT for every DDname in all
 * steps for a job with a name of TESTJB and a job, stc, or tsu number
 * Ø2367.
 * These examples demonstrate the flexibility of selection available.
 * The JOBNAME= parameter specification supports SAPI (SSI 79 function
 * call) defined masking.  Valid wildcard masking characters are
 * '*' to mask multiple characters and '?' to mask any single
 * character.
 * The SPOOLRD program makes use of the GETSSS2() supplied Assembler
 * code function that returns the spool dataset browse token that is
 * used by dynamic allocation to allocate the spool dataset for read
 * access.
```

```
 * SPOOLRD is compatible with either IBM C or C++.
 */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#pragma pack(packed)
struct SSS2 {
  short int      SSS2LEN;        /* I.Length of Sysout extension     */
  unsigned char  SSS2VER;        /* I.SSOB version                   */
  char           SSS2REAS;       /* O.Reason code associated with    */
  unsigned char  SSS2EYE[4];     /* I.Eye catcher                    */
  unsigned char  SSS2APPL[8];    /* I.For application use.  Either   */
  unsigned char  SSS2APL1[20];   /* I.For application use.           */
  unsigned char  SSS2TYPE;       /* I.Type of call                   */
  int            _filler1 : 24;  /* Reserved for future use and must */
  struct {
    void          *_SSS2ECBP; /* I.ECB address (see above) */
    } SSS2INPT;
  unsigned char  SSS2RBA[8];     /* IO.Relative Byte Address of      */
  unsigned char  SSS2UFLG;       /* IBM.User disposition flags       */
  unsigned char  _filler2[2];    /* Reserved for future use and must */
  unsigned char  SSS2SEL1;       /* IS.Dataset selection flags       */
  unsigned char  SSS2SEL2;       /* IS.More dataset selection flags  */
  unsigned char  SSS2SEL3;       /* IS.More dataset selection flags  */
  unsigned char  SSS2SEL4;       /* IS.More dataset selection flags  */
  unsigned char  SSS2SEL5;       /* IS.More dataset selection flags  */
  unsigned char  _filler3[2];   /*Reserved for future use and must be*/
  unsigned char  SSS2MSC1;       /* IS.Current dataset misc. flags   */
  unsigned char  _filler4[3];   /*Reserved for future use and must be*/
  unsigned char  SSS2JOBN[8];    /*IS*.Jobname used for selection (if*/
  unsigned char  SSS2JBIL[8];    /* IS.Low jobid used for selection  */
  unsigned char  SSS2JBIH[8];    /* IS.High jobid used for selection */
  unsigned char  SSS2CREA[8];    /* IS*.Owning userid used for       */
  unsigned char  SSS2PRMO[4][8]; /* IS*.1 to 4 PRMODEs used for      */
  unsigned char  SSS2DEST[18];   /* IS*.Destination value used for   */
  unsigned char  _filler5[18];   /* Reserved for future use and must */
  unsigned char  SSS2DES2[18];   /* IBM.Destination value used for   */
  unsigned char  SSS2PGMN[8];    /* IS*.User writer name used for    */
  unsigned char  SSS2FORM[8][8]; /*IS*.Form numbers used for selection*/
  unsigned char  _filler6[8];    /* Reserved for future use and must */
  unsigned char  _filler7[8];    /* Reserved for future use and must */
  unsigned char  SSS2CLSL[36];   /* IS.Sysout class list used for    */
  unsigned char  SSS2CLAS;       /* IBM.New class if SSS2SETC is on. */
  unsigned char  _filler8[7];    /* Really reserved for future SYSOUT*/
  int            SSS2LMIN;       /* IS.Minimum line count for dataset*/
  unsigned char  SSS2LMAX[4];    /* IS.Maximum line count for dataset*/
  int            SSS2PMIN;       /* IS.Minimum page count for dataset*/
  unsigned char  SSS2PMAX[4];    /* IS.Maximum page count for dataset*/
  unsigned char  SSS2FCB[4];     /* IS.FCB image name used for       */
```

```
unsigned char  SSS2UCS[4];     /* IS.UCS image name used for       */
unsigned char  SSS2CHAR[4][4]; /* IS.Printer translate table       */
unsigned char  SSS2MOD[4];     /*IS.Modify image used for selection*/
unsigned char  SSS2FLSH[4];   /*IS.Flash cartridge ID for selection*/
void          *SSS2SECT;       /* I.Zero or an address of where the*/
unsigned char  SSS2AGE[4];     /* IS.Minimum age of datasets to be */
unsigned char  SSS2VOL[4][6];  /* IS.List of SPOOL volume serial   */
void          *SSS2CTKN;      /*IS.Address of client token @RØ5LOPI*/
unsigned char  SSS2ODST[8];   /*IS*.Origin node name used  @OW297Ø7*/
int            _filler9[9];   /*Reserved for future use and @OW297Ø7*/
struct {
  unsigned char  _SSS2DSP1;    /*ID.Flags describing the disposition*/
  unsigned char  _filler1Ø[3];  /* Reserved for future use and must*/
  } SSS2DISP;
unsigned char  SSS2DCLS;       /* ID.New class                     */
unsigned char  _filler11[7];   /* Really reserved for future use   */
unsigned char  SSS2DFOR[8];    /* ID.New forms                     */
unsigned char  SSS2DPGM[8];    /* ID.New user writer name          */
unsigned char  SSS2DDES[18];   /* ID.New destination.              */
short int      SSS2CLFT;      /*ID.Number of copies left to process*/
int            _filler12[12];  /* Reserved for future use and must */
unsigned char  SSS2JEST[12];   /* O.JES token associated with this */
void          *SSS2BTOK;       /* O.Address of a JES initialized   */
short int      SSS2COPY;       /*O.Total number of copies requested*/
short int      _filler13;     /*Reserved for future use and must be*/
char           SSS2CPYG[8];    /* O.Copy groups                    */
unsigned char  SSS2JOBR[8];    /* O.Jobname of selected job        */
unsigned char  SSS2JBIR[8];    /* O.Job ID of selected job         */
unsigned char  SSS2OJBI[8];    /* O.Original jobid of selected job.*/
unsigned char  SSS2CRER[8];    /* O.Owning userid of dataset       */
unsigned char  SSS2JDVT[8];    /* O.JCL Definition Vector Table    */
unsigned char  SSS2PRMR[8];    /* O.PRMODE of dataset selected     */
unsigned char  SSS2DESR[18];   /*O.Destination of selected dataset.*/
unsigned char  SSS2PGMR[8];    /* O.Writer name of selected dataset*/
unsigned char  SSS2FORR[8];    /* O.Form number of selected dataset*/
unsigned char  SSS2TJN[8];     /*O.APPC Transaction Program Jobname*/
unsigned char  SSS2TJID[8];    /* O.APPC Transaction Program Job ID*/
unsigned char  SSS2DSN[44];    /* O.Dataset name of selected data  */
short int      _filler14;      /* Reserved for future use and must */
int            SSS2SEGM;       /* O.Segment id (zero if dataset not*/
int            SSS2WRTN;       /* O.SWB Processing Error - Return  */
int            SSS2WRSN;       /* O.SWB Processing Error - Reason  */
unsigned char  SSS2CLAR;       /*O.Sysout class of selected dataset*/
unsigned char  _filler15[7];   /*Really reserved for future use and*/
short int      SSS2MLRL;       /* O.Maximum logical record length  */
unsigned char  SSS2DSID[8];    /* O.DSID for the selected dataset  */
unsigned char  SSS2RET1;       /* O.Returned flags                 */
unsigned char  SSS2RET2;       /* O.Returned flags                 */
unsigned char  SSS2RET3;       /* O.Returned flags                 */
```

```
  unsigned char  SSS2RET4;       /* O.Returned flags                */
  unsigned char  SSS2RET5;       /* O.Queue where dataset   @OW32461 */
  unsigned char  _filler16;      /* Reserved for future use @OW32461 */
  int            SSS2LNCT;       /* O.Line count                    */
  int            SSS2PGCT;       /* O.Page count                    */
  int            SSS2BYCT[2];   /*O.Byte count after blank truncation*/
  int            SSS2RCCT;       /* O.Record count (JES3 only)      */
  unsigned char  SSS2PRCD[8];    /* O.Procname for the step creating */
  unsigned char  SSS2STPD[8];    /* O.Stepname for the step creating */
  unsigned char  SSS2DDND[8];    /* O.DDNAME for the dataset creation*/
  unsigned char  SSS2SWBT[8];    /* O.Token used for SJFREQ services.*/
  void          *SSS2SWTU;       /*O.Address of the SWBTU block. This*/
  unsigned char  SSS2PRIV[8];    /*IO.Copied from/to SAPPRIV if JES2,*/
  unsigned char  SSS2CHR1[4];    /* O.Printer translate table 1     */
  unsigned char  SSS2CHR2[4];    /* O.Printer translate table 2     */
  unsigned char  SSS2CHR3[4];    /* O.Printer translate table 3     */
  unsigned char  SSS2CHR4[4];    /* O.Printer translate table 4     */
  unsigned char  SSS2OGNM[26];   /* O.JES2 output group name        */
  unsigned char  _filler17[2];  /*Reserved for future use and must be*/
  unsigned char  SSS2RMOD[4];    /* O.Printer copy modification     */
  char           SSS2MODT;      /*O.Printer table reference character*/
  unsigned char  SSS2RFLS[4];    /* O.Printer flash cartridge ID    */
  char           SSS2FLSC;       /* O.Number of flash copies        */
  char           SSS2PRIO;       /* O.Dataset priority              */
  char           SSS2LINC;       /* O.Lines/page (JES2 only)        */
  unsigned char  SSS2TOD[4];     /* O.Date and time of dataset      */
  int            SSS2CDS;        /* O.Count of work units (JOEs/OSEs)*/
  void          *SSS2NJED;       /* O.Address of NJE dataset header. */
  unsigned char  SSS2FCBR[4];    /*O.Forms Control Buf (FCB) @OW32461 */
  unsigned char  SSS2UCSR[4];   /*O.Univ Character Set (UCS) @OW32461 */
  void          *SSS2DSTR;       /*O.Address of dataset token @OW36Ø19*/
  int            _filler18[9];  /*Reserved for future use and @OW36Ø19*/
  unsigned char  SSS2PNAM[2Ø];   /* O.Programmer name from the JOB   */
  unsigned char  SSS2ROOM[8];    /* O.Job level room number         */
  unsigned char  SSS2NOTN[8];    /* O.Job notify node               */
  unsigned char  SSS2NOTU[8];    /* O.Job notify userid             */
  void          *SSS2ACCT;       /* O.Address of encoded accounting  */
  unsigned char  SSS2XEQ[8];     /* O.Node where job executed       */
  unsigned char  SSS2ORG[8];     /* O.Node where job entered network */
  int            SSS2TIME;       /* O.Time on input processor for the*/
  unsigned char  SSS2DATE[4];    /* O.Date on input processor for the*/
  unsigned char  SSS2SYS[8];     /* O.System name of the MVS image   */
  unsigned char  SSS2MBR[4];     /* O.Member name of the JES2 image  */
  void          *SSS2NJEJ;       /* O.Address of NJE job header.     */
  unsigned char  SSS2NACT[8];    /* O.Net account (from /*NETACCT)   */
  int            _filler19[1Ø];  /*Reserved for future use and must be*/
  };
#define SSS2ECBP  SSS2INPT._SSS2ECBP
#define SSS2DSP1  SSS2DISP._SSS2DSP1
```

```
/* Values for field "SSS2VER" */
#define SSS2IVER 1    /* Initial version number    @RØ5LOPI   */
#define SSS2VCTP 2    /* Version supporting                   */
#define SSS2CVER Øx02 /* Current version number    @RØ5LOPI   */
```

Editor's note: this article will be concluded next month.

Rudy Douglas
System Programmer (Canada)

IBM is modernizing COBOL applications by bridging its mainframe-oriented COBOL and WebSphere products to EJB and Service-Oriented Architectures (SOA) in new versions of its Enterprise COBOL and WebSphere Studio Enterprise developer products.

With Version 3.3 of Enterprise COBOL, the language can be extended to Web applications, Simple Object Access Protocol (SOAP), and HTTP. Programmers can generate outbound XML from a COBOL data structure and interoperate with EJB. Developers can write EJB in COBOL 3.3 on WebSphere z/OS.

WebSphere Studio Enterprise Developer 5.1.1 features integrated development environment capabilities for both Java 2EE and COBOL development. The product supports SOAs in that SOAP-based access can be developed for mainframe applications. A mainframe system, for example, could be linked to a Windows program.

WebSphere applications can communicate with CICS applications without having to separately buy WebSphere Studio Application Developer Enterprise Integrator

For further information contact your local IBM representative.

* * *

Mainstar Software has announced Release 6.2.02 of Catalog RecoveryPlus (CR+), its ICF catalog management software.

The new release builds on the 'REORG While Open' feature, expanding 24x7 ICF catalog availability. The 'REPAIR While Open' feature will allow installations to repair user catalogs while they are open and in-use. When REPAIR is coded with the REORG While Open facility, users are not required to quiesce applications (batch and online) that have datasets open through the user catalog, and the catalog can remain open to the Catalog Address Space (CAS) on any shared access systems.

For further information contact:
Mainstar Software, PO Box 4132, Bellevue, WA 98009-4132, USA.
Tel: (425) 455 3589.
URL: http://www.mainstar.com/pdf/010-0101_CAT6201_RA.pdf.

* * *

Software AG has announced XML Business Integration Portfolio, which will enable enterprises to make data from legacy systems available to users, the Web, and business applications.

The portfolio consists of three packages – an Enterprise Legacy Integrator to turn transactions into 'events' and expose them as Web services; an Enterprise Service Integrator to connect and manage these events within a service-oriented architecture; and an Enterprise Information Integrator that uses metadata to provide a single view of information in disparate systems.

For further information contact:
Software AG, 11190 Sunrise Valley Drive, Reston, VA 20191, USA.
Tel: (703) 860 5050.
URL: http://www2.softwareag.com/Corporate/News/latestnews/20040527_XML_Business_Integration_page.asp.

* * *