May 2004

In this issue

3  Audit trail of IPLs
5  Boosting VSAM performance with SMB
25 JCL tips – part 2
33 IXFP Snapshot performance tips
49 Displaying the virtual storage map from IPCS
76 MVS news

© Xephon Inc 2004
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.

Disclaimer
Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, EXECs, and other contents of this journal before making any use of it.

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.
Audit trail of IPLs

We use the following program to write an audit trail of IPLs for our systems.

It creates a record containing sysid, IPL date/time, and the unit and volser the system was IPLed from. This currently runs on an OS/390 2.10 system.

It is started from each system’s COMMNDxx member:

`COM='S IPLLOG'`

which starts an STC:

```plaintext
// IPLLOG PROC
// S1 EXEC PGM=IPLLOG
// STEPLIB DD DISP=SHR,DSN=SYS2.LINKLIB
// IPLLOG DD DISP=MOD,DSN=SYS6.IPLLOG
```

SYS6.IPLLOG is a dataset (RECFM FB, LRECL 80) that is accessible to and shared by all systems. Records with the following format are written for each invocation:

```plaintext
*** MVS1 IPLED 2003.317 AT 13.27 FROM PRS2AS(499B)
*** MVS2 IPLED 2004.001 AT 10.24 FROM PRS2AA(4990)
*** MVS3 IPLED 2004.031 AT 12.30 FROM PRS2AS(499B) etc
```

PROGRAM SOURCE

```plaintext
***********************************************************************
*         'IPLLOG' - CREATE A LOG OF IPL DATES                        *
***********************************************************************
IPLLOG   CSECT
*PLLOG   AMODE 31
*PLLOG   RMODE ANY
BAKR  R14,Ø                   SAVE CALLER DATA ON STACK
LR    R12,R15                 GET ENTRY POINT
USING IPLLOG,R12              ESTABLISH ADDRESSABILITY
L     R15,16                  CVTPTR
L     R2,196(,R15)            CVTSMCA
MVC   OUTSYSID,16(R2)         SMF ID INTO OUTPUT LINE
LA    R3,340(,R2)             LOCATE IPL DATE (JULIAN)
LA    R4,336(,R2)             LOCATE IPL TIME
ED    JULIAN,1(R3)
MVC   OUTYYDDD,JULIAN+1       JULIAN DATE INTO OUTPUT LINE
```
L R1,Ø(R4) LOAD IPL TIME
SLR R0, R0
D R0, =F360000' CALCULATE HH
CVD R1, DWORD
OI DWORD+7, X'0F'
UNPK OUTHH, DWORD+6(2)
LR R1, R0
SLR R0, R0
D R0, =F6000' CALCULATE MM
CVD R1, DWORD
OI DWORD+7, X'0F'
UNPK OUTMM, DWORD+6(2)
L R2, 48(R15) GET SYSRES UCB POINTER
MVC OUTVOLID, 28(R2) VOLID INTO OUTPUT LINE
UNPK UNPKFLD(5), 4(3, R2) UNPK BINARY CCUU + 1 BYTE
TR UNPKFLD(4), TRTAB-240 MAKE IT DISPLAYABLE HEX
MVC OUTCCUU, UNPKFLD GET UNIT ADDRESS
OPEN (IPLOUT,(OUTPUT)) WRITE THE INFO
CLOSE (IPLOUT)
XR R15, R15 SET RC=Ø
PR , RETURN

***********************************************************************
* WORKAREA                                                        *
***********************************************************************
OUTREC DC CL8'*** XXXX IPL ED 2ØYY.DDD AT HH.MM FROM VOLID (CCUU)'
OUTSYSID EQU OUTREC+4, 4
OUTYYDDD EQU OUTREC+17, 6
OUTHH EQU OUTREC+27, 2
OUTMM EQU OUTREC+30, 2
OUTVOLID EQU OUTREC+38, 6
OUTCCUU EQU OUTREC+45, 4
JULIAN DC XL7'F021204B202020' CONTAINS EDIT PATTERN
DWORD DS D
UNPKFLD DS CL5
* IPLOUT DCB DDNAME=IPLLOG, X
DSORG=PS, X
RECFM=FB, X
LRECL=80, X
MACRF=PM
*
TRTAB DC C'0123456789ABCDDEF'
*
YREGS
*
END

Grant Carson
Senior Mainframe Architect (UK) © Xephon 2004
Boosting VSAM performance with SMB

Ever since its introduction some 30 years ago, VSAM has been a popular and reliable data storage construct on MVS systems. VSAM is still the cornerstone of on-line applications such as IMS and CICS, and is widely used in ISV packages and in-house-written batch applications. However, with 24x7 operation becoming a necessity, batch windows must shrink in order to lessen their impact on on-line systems. The most effective way to cut down the batch window is to optimize I/O, and this article examines the results of a sample tuning exercise. Of course, it is a well-known fact that the fastest I/O is the one that is never issued. That is to say, an application that is having I/O performance problems will perform better if we can cut down on the number of I/Os. The whole concept of SMS automatic blocksize according to DASD device was to reduce the number of I/Os for that specific DASD device type. Similarly, for VSAM data, automatic effective buffering can significantly reduce the number of I/Os, response time, and elapsed job time, thereby improving application performance.

It is well known that VSAM is very important to most installations, yet it is rarely utilized optimally. One consequence of this is that jobs accessing VSAM files almost always run longer than necessary. Thus, tuning native VSAM datasets is still an important part of the overall tuning process at many installations. Almost certainly, the largest performance gains can be achieved with good VSAM buffering – it is in fact the single most important aspect of VSAM tuning and will achieve the biggest performance boost. If implemented correctly, these buffering methodologies will greatly reduce disk I/Os, reduce CPU time, and lead to better job turnaround time. Now, with the advent of System Managed Buffering (SMB), high performance can be achieved through standard OS/390 system interfaces, with virtually no application programmer effort, and with no JCL changes. System Managed Buffering is a feature of DFSMSdfp, directed at support for batch application processing, and is intended as a
means of achieving two things. The first one is to update the current defaults for processing VSAM datasets. This is necessary in order to utilize current hardware technology to effect the processing of VSAM data. The second one is to initiate a buffering technique, other than that specified by the application program, that would improve application performance.

SYSTEM MANAGED BUFFERING

Before we see how SMB works and how you can take advantage of it, it might be useful to understand the overall buffering picture for VSAM files. There are two ways of addressing buffering of VSAM data offered by OS/390 and z/OS. The first method uses Non-Shared Resources (NSR), where buffers are dedicated to the processing of a single VSAM file. NSR means that each VSAM file in a task will have its own dedicated buffers assigned within the program address space, and, hence, will not share them with any other VSAM file that is open within the task. NSR is also the automatic default type of VSAM buffering logic. On the other hand, using Logically Shared Resources (LSR) allows the sharing of buffer pools among multiple VSAM files. While both NSR and LSR can be defined within an application, there is a significant difference when it comes to how an application maximizes its use of these buffer pools to process data – sequential processing of data works best using NSR and random data processing works best using LSR. Does this mean that applications need to be changed? Sometimes they need changing. Moreover, sometimes it is necessary to understand the type, access method, format, and options of the file. For example, do dataset options call for key access (KEY), sequential access (SEQ), addresses access (ADR), or access to CI (CNV)?

System Managed Buffering (SMB) for VSAM datasets is a fairly new facility introduced with DFSMS Version 1.4 for KSDS files only. This was enhanced with DFSMS 1.5 to include all types of VSAM file. Basically, the system decides how many buffers to use for data and index portions (the case of NSR) or buffer
pools size (the case of LSR), with four basic buffer allocation algorithms that can be chosen or specified:

- **Direct Optimized (DO)** – SMB optimizes for totally random record access. This is appropriate for applications that access records in a dataset in totally random order. This technique will override the user specification for using NSR buffering with an LSR buffering implementation. Random-access VSAM processing is automatically directed to use LSR, which will eliminate buffer stealing, exploit look-aside processing, ESA hiperspaces, and in-core indexes. The DO technique is elected if the ACB specifies only the MACRF=(DIR) option for accessing the dataset. If either SEQ or SKP are specified, in combination with DIR or independently, DO is not selected. The selection can be overridden by the user specification of ACCBIAS=DO on the AMP= parameter of the associated DD statement. Note should be taken of the fact that the MACRF type of access is just an intention. The real type of access is declared per I/O operation in the RPL.

- **Direct Weighted (DW)** – SMB optimizes for mixed-mode processing (both direct and sequential), but ‘weights’ the buffer allocations for key-direct. This will provide minimum read-ahead buffers for sequential retrieval and maximum index buffers for direct requests. The size of the dataset is a minor factor in the storage that is required for buffering. This technique requires approximately 100KB of processor storage for buffers, with a default of 16MB.

- **Sequential Optimized (SO)** – SMB optimizes for sequential processing. It is appropriate for applications reading the entire dataset from the first to last record or a large percentage in sequential order. The size of the dataset is not a factor in the processor virtual storage that is required for buffering. Approximately 500KB of processor virtual storage, defaulted to above the 16MB line, is required for buffers for this technique.

- **Sequential Weighted (SW)** – SMB optimizes for mixed-
mode processing (both direct and sequential), but ‘weights’ the buffer allocations for sequential. It will use read-ahead buffers for sequential and provide additional index buffers for direct requests. The read-ahead will not be the large amount of data transferred as with SO. The size of the dataset is a minor factor in the amount of processor virtual storage that buffering requires. This technique requires approximately 100KB of processor virtual storage for buffers, with the default above 6MB.

General discussion and guidelines related to processing with each technique are fully documented in *VSAM Demystified* (SG24-6105).

The change-over to SMB is easy enough – it can be simply done by defining an extended format dataset through an SMS data class with RECORD_ACCESS_BIAS=SYSTEM/USER. Or, if you prefer JCL changes, it can be invoked in a specific job stream by specifying ACCBIAS on the AMP parameter for the dataset’s DD statement.

In the first case, the technique that will be defaulted to by the system is based on the application specification for the type of access intention ACB MACRF=(DIR,SEQ,SKP) and influenced by the specifications in the associated Storage Class (SC) for direct millisecond response, direct bias, sequential millisecond response, and sequential bias.

In the second case, the technique is externally specified by using the ACCBIAS JCL subparameters of the AMP DD parameter – probably the easiest and best option is the ACCBIAS=SYSTEM option. You can specify ACCBIAS equal to one of the following values:

- **USER** – bypass SMB. This is the default if you code no specification for the ACCBIAS subparameter. This default is *not* used when the data class specifies RECORD_ACCESS_BIAS.
- **SYSTEM** – force the system to determine the buffering technique.
One can also explicitly request a specific buffer allocation algorithm by specify the SMB buffer processing as SO/SW/DO/DW. One of the problems with SMB arises in situations where you have a batch program that does skip-sequential, sequential, and random processing all in the same run. In many such cases, that we have seen it’s often been a good compromise just to default to ACCBIAS=SYSTEM. For a detailed description of each AMP option see *MVS: JCL Reference* (SA22-7597).

During a testing phase we turned on Systems Managed Buffering (through DATACLASS) for a large VSAM file, but in order to see how SMB works, as well as to prevent production problems, we decided to bypass SMB processing by specifying RECORD_ACCESS_BIAS=USER and later on we used JCL’s AMP parameter ACCBIAS=SO (see below):

```
VSAM file buffers & buffering management

<table>
<thead>
<tr>
<th>Records</th>
<th>Access:</th>
<th>Buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job</td>
<td>Run date</td>
<td>Elapsed time</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>MYJOB</td>
<td>17 Oct 2003 00:49:24:46</td>
<td>PROD.HISTFILE.DATA</td>
</tr>
<tr>
<td>(1) MYJOB</td>
<td>17 Oct 2003 00:49:24:46</td>
<td>PROD.HISTFILE.INDEX</td>
</tr>
<tr>
<td>MYJOB</td>
<td>18 Oct 2003 00:42:45:09</td>
<td>PROD.HISTFILE.DATA</td>
</tr>
<tr>
<td>MYJOB</td>
<td>18 Oct 2003 00:42:45:09</td>
<td>PROD.HISTFILE.INDEX</td>
</tr>
<tr>
<td>MYJOB</td>
<td>19 Oct 2003 00:50:07:19</td>
<td>PROD.HISTFILE.DATA</td>
</tr>
<tr>
<td>(2) MYJOB</td>
<td>19 Oct 2003 00:50:07:19</td>
<td>PROD.HISTFILE.INDEX</td>
</tr>
<tr>
<td>MYJOB</td>
<td>21 Oct 2003 00:50:57:52</td>
<td>PROD.HISTFILE.DATA</td>
</tr>
<tr>
<td>MYJOB</td>
<td>21 Oct 2003 00:50:57:52</td>
<td>PROD.HISTFILE.INDEX</td>
</tr>
<tr>
<td>MYJOB</td>
<td>22 Oct 2003 00:26:22:79</td>
<td>PROD.HISTFILE.DATA</td>
</tr>
<tr>
<td>(3)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Notes:

(1) Job access statistics before converting dataset to extended format.

(2) Dataset converted to extended format with Rec_Acc_Bias=USER (bypass SMB).

(3) JCL AMP parameter override of data class definition (ACCBIAS=SO).

The order of precedence for specifying values that decide if and how SMB will be invoked is this: JCL specifications, then the data class Record_Access_Bias parameter, then the storage class parameters, then the MACRF values. That is, whatever is specified in the JCL will always take precedence. This also means that one may wish to tell a lie to VSAM about intent (for example direct versus sequential processing) and SMB will be fooled. Because SMB is not taking any sample of behaviour, it relies on the access intent of the OPEN. However, telling a lie is not a wise thing to do: incorrect use of a buffering strategy will result in a significant increase in I/O, thus causing long-running batch jobs and poor performance (see below):

<table>
<thead>
<tr>
<th>Buffering</th>
<th>EXCPs</th>
<th>Clock time (min)</th>
<th>CPU time (sec)</th>
<th>CONN (k)</th>
<th>Buffers (D/I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSR - Default</td>
<td>402472</td>
<td>41.4</td>
<td>251.16</td>
<td>1099</td>
<td>4</td>
</tr>
<tr>
<td>ACCBIAS=SO</td>
<td>34991</td>
<td>26.0</td>
<td>233.66</td>
<td>918</td>
<td>49</td>
</tr>
<tr>
<td>ACCBIAS=DO</td>
<td>793868</td>
<td>45.0</td>
<td>294.28</td>
<td>1342</td>
<td>0</td>
</tr>
</tbody>
</table>

Gain using SMB (%): 91.3 37.19 9.96 13.73
(Do vs. Default): 97.2 8.7 17.16 22.11

SMB RESTRICTIONS AND POTENTIAL PROBLEMS

There are two main restrictions to SMB. The first one is that
SMB support is currently limited to extended format VSAM files that use NSR buffering. To be in extended format, the dataset must be system managed (SMS) and use a data class defined with DSNTYPE=EXT. On the other hand, SMB will get involved only when NSR buffering is specified by the application program, ACB MACRF=(NSR). It will not get involved with the MACRF parameters RST (ACB reset option), UBF (USER buffering), GSR (Global Shared Resources), LSR (Local Shared Resources), RLS (Record Level Sharing), or ICI (Improved Control Interval processing). For releases prior to z/OS 1.3 DFSMS, processing the dataset through the alternate index of the path specified in the DDname is not supported. When the conditions above are not satisfied, the job does not abend, but the SMB services are not used and no messages are issued.

The second restriction is that SMB is invoked at dataset open processing only: after the initial decision is made during that process, SMB has no further involvement.

Thus far two basic storage-related problems have emerged, especially regarding the use of the ACCBIAS=DO option. SMB ACCBIAS=DO is in fact equivalent to BLSR in that, in both cases, VSAM LSR buffer pools are built for each dataset opened with this technique in a single application program. The size of the pool is based on the actual dataset size at the time the pool is created. A separate pool is built for both data and index components, if applicable, for each dataset. There is no capability for a single pool to be shared by multiple datasets. The index pool is sized to accommodate all records in the index component. The data pool is sized to accommodate approximately 20% of the user records in the dataset. This also means that the processor virtual storage requirement will increase with each OPEN after records have been added and the dataset has been extended beyond its previous size. Thus, for very large VSAM KSDS files, a program or job step might abend with ACCBIAS=DO because of storage problems unless SMB’s default options regarding buffer pool allocations are overridden.
Again, two options are available to tackle this problem. Increasing the job’s region size to support the buffers (think multiple megabytes just for the buffers) might avoid abends. Then again, it might not help, as was the case with a very large VSAM KSDS file we were testing, even though we had increased the job’s region size to the maximum possible.

On the other hand, the use of the SMBVSP parameter on the AMP=parameter (not present in the data class specification) can alleviate the storage impact since it restricts the amount of virtual storage to be obtained for buffers when opening the dataset. It is used to override the default buffer space to be obtained, which is calculated assuming that 20% of the data accounts for 80% of the accesses. The buffer space acquired is split across two LSR pools – one for the index and one for the data.

There is also an additional AMP parameter that can be used in conjunction with the SMBVSP parameter, and it can help to reduce the storage problems. The SMBHWT parameter can be used to provide buffering in hiperspace in combination with virtual buffers for the data component. These buffers may be allocated for the base data component of the sphere. If the CI size of the data component is not a multiple of 4KB, both virtual space and hiperspace are wasted. It can be specified as an integer from 0 to 99. The value specified acts as a weighting factor for the number of hiperspace buffers to be established. This can reduce the size required for an application region, but does have implications related to processor cycle requirements. That is, all application requests must orient to a virtual buffer address. If the required data is in a hiperspace buffer, the data must be moved to a virtual buffer after ‘stealing’ a virtual buffer and moving that buffer to a Least Recently Used (LRU) hiperspace buffer.

Finally, if the optimum amount of storage required for this option is not available, SMB will reduce the number of buffers and retry the request. The retry capability for the DO technique was added in z/OS 1.3 DFSMS. For data, SMB will make two
attempts, with a reduced amount and a minimum amount. For an
index, SMB reduces the amount of storage only once, to a
minimum amount. If all attempts fail, the DW technique is used.
The system issues an IEC161I message to advise that this has
happened.

If you are running a 24-bit program (amode=rmode=24) be
aware that the storage for buffers for SMB techniques are
obtained above 16 megabytes (above the line), and in order to
prevent problems IBM recommends that RMODE31=NONE be
specified on the AMP= parameter for those datasets using
SMB.

IDENTIFYING JOBS THAT MIGHT BENEFIT FROM SMB

The jobs that might benefit from SMB are those with certain
application characteristics, most important of which are a data
reference pattern and options specified by the application
program (ACB MACRF). The best candidates are long-running
jobs as well as jobs with a high execute channel program
(EXCP) count.

SMF type 64 records are probably used more frequently than
any other data source for tuning VSAM applications. Using
these records you can identify the programs with the highest
amount of VSAM activity (such as number of EXCPs, retrievals,
inserts, deletes, CI and CA splits, insert strategy), analyse the
effectiveness of buffer usage, and determine whether the
dataset is being used concurrently by other jobs or tasks. To
determine candidates for SMB, we have used SMF type 64
records to obtain information about the SMB candidate’s
processing characteristics, including jobname, cluster/
component name, change in number of EXCPs, and ACB
MACRF fields. In addition, SMF type 64 records indicate
whether a reduced or minimum amount of resource is being
used for a data pool and whether DW is used. Bits 5–7 of
SMF64RSC, which were previously reserved, are used to give
more information about Direct Optimization (DO).

A detailed description of the layout of the SMF type 64 record
can be obtained from the *MVS System Management Facilities (SMF)* (SA22-7630) manual. One can also find the type 64 subtype descriptions in macro IDASMF64 in SYS1.MACLIB.

**CODE**

Based on record descriptions obtained from the above mentioned manual, a sample SMB report writer was written. The code is a two-part stream. In the first part (COPYSMF) selected SMF records (selection being defined by INCLUDE’s condition) are copied from the SMF dataset to a file that can be used as a base of archived records. In the second part, SMB64, the captured records are formatted by invoking SMB EXEC and two reports are produced.

Each report consists of two sets of variables. The first set is a fixed one consisting of the variables that uniquely identify the VSAM file or job being monitored. This set is meant to be used across all reports. The pool of variables in this set contains generated observation number, job name, date stamp, dataset allocation elapsed time, cluster/component name, total number of records, number of records retrieved in a job run, and number of EXCPs. The second set of printed variables is area specific and pertains only to the VSAM file performance domain being monitored. Note should be taken of the fact that elapsed time in these reports is not the execution clock time (wall time) that we are accustomed to thinking of. This ‘elapsed’ time in fact represents the length of time the file was kept open (for details see APAR OW43854).

The first report shows standard VSAM file attributes and processing activity as well as the type of access to the record – key, rba (relative byte addresses) or cnv(access the dataset by control interval), dataset addressability, and format. As already stated, there are some restrictions when considering the use of SMB. This report shows whether there are any restrictions – user buffering, ICI processing, alternate index, NSR required, and/or if extended format is required.
The second report is a VSAM file buffer management report and it provides buffering-related information such as number of buffers used per component (system determined or user defined), buffer space, addressing mode for buffers (24/31 mode), as well as whether or not the buffers have been fixed in real storage. The more interesting part of the report provides answers to questions like: Is there any method to find out whether SMB gets invoked at all? Wouldn’t it be nice not only to know that SMB is invoked but also how much (and what) it does to the job (or datasets). This report provides the answer to these two questions by means of SMB-related information. Was SMB invoked at all? (no, yes: by JCL or SYSTEM); which optimization technique was used? (DO, DW, SO, SW, or none); and in conjunction with that, what data reference pattern was used: sequential access (records were requested in either ascending or descending sequence), direct access (records were randomly requested), skip sequential (records were processed in sequence but some records may have been skipped), or a combination of these? In the case of the Direct Optimized (DO) technique, additional indicators are available, such as the amount of virtual storage set by the SMBVSP parameter, whether hiperspace buffers were used, whether insufficient virtual storage problems occurred, indicators of whether a reduced or minimum amount of resource is being used for a data pool, and whether DW is used (the case of retry technique).

SMBJOB

//DEL EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=X
//SYSIN DD *
//       DELETE hlq.SMF64.DATA
//      SET MAXCC=0
/*
//COPYSMF EXEC PGM=ICETOOL
//TOOLMSG DD SYSOUT=* 
//DFSMSG DD SYSOUT=* 
//RAWSMF DD DSN=your.smf.dataset,DISP=SHR
//SMF64 DD DSN=hlq.SMF64.DATA,
//       SPACE=(CYL,(x,y)),UNIT=SYSDA,
SMB EXEC

/* REXX EXEC to read SMF 64 records - VSAM Component/Cluster Status */
signal ON ERROR
/*------------------------------------------------------------------*/
/* Part 1: Handle file allocation & dataset existence and */
/* print report header and labels */
/*------------------------------------------------------------------*/
Address TSO

userid=SYSVAR(SYSUID)
r64fa =userid||'.r64fa.rep' /* File processing/attribute*/
r64bf =userid||'.r64bf.rep' /* Buffering report */
x = MSG('OFF')
IF SYSDSN(r64fa) = 'OK'
THEN "DELETE "r64fa" PURGE"
IF SYSDSN(r64bf) = 'OK'
THEN "DELETE "r64bf" PURGE"
"ALLOC FILE(S64FA) DA("R64FA")",
" UNIT(SYSALLDA) NEW TRACKS SPACE(90, 30) CATALOG",
" REUSE RELEASE LRECL(245) RECFM(F B)"
"ALLOC FILE(S64BF) DA("R64BF")",
" UNIT(SYSALLDA) NEW TRACKS SPACE(90, 30) CATALOG",
" REUSE RELEASE LRECL(205) RECFM(F B)"
fi.1  =left(' ',8,' '),'VSAM file processing & attribute report'||left(' ',15,' ')
fi.2  ='
fi.3  =left(' ',8,' ')||'Report produced on',
      ||left(' ',1,' ')||left(date(),11),
      ||left(' ',1,' ')||left(' at ',3,' ')||left(time(),10)
fi.4  ='
fi.5  =left(' ',83,' ')||left(' # of',4)||left(' ',2,' '),
After reading the document, here is the plain text representation:

```
|| left('Records in this run:', 20) || left('  ', 10, ' '),
|| left('Access by:', 10) || left('  ', 16, ' ') || left('Cl', 2),
|| left('  ', 5, ' ') || left('Index', 5),
|| left('  ', 6, ' ') || left('-- Split -- Insert', 20),
|| left('  ', 8, ' ') || left('Data set:', 9),
|| left('  ', 9, ' ') || left('Res.', 4)
fi.6 = left('obs', 3) right('Job name', 8) left('Run date', 11),
  left('Elapsed time', 14) left('Cluster/Component name', 30),
  right('Excip', 6) right('Records', 9) right('ret'ved', 8),
  left('delete insert update', 21) right('Key', 3),
  right('Rba', 3) right('Cnv', 3) right('Cl', 3),
  left('Recl', 4) right('kl', 3) right('size', 4),
  right('level', 8) right('Cl', 7) right('CA', 9),
  center('strategy', 8) right('ext.', 5),
  left('Address.', 8) left('Format', 8) left('sharing', 7),
  left('Restriction', 12)

fi.7 = left('  ', 242, '  ')
"EXEClO * DISKW s64fa (STEM fi.)"
bf.1 = left(' ', 8, ' '),
  || 'VSAM file buffers & buffering management'|| left('  ', 15, ' ')
bf.2 =
bf.3 = left('  ', 8, ' ') || 'Report produced on',
  || left('  ', 1, ' ') || left('date()', 11),
  || left('  ', 3, ' ') || left('at ', 3, ' ') || left('time()', 10)
bf.4 = left('  ', 76, ' ')
  || left('# of Records', 13),
  || left('  ', 8, ' ') || left('Access:', 8) || left('  ', 8, ' '),
  || left('Buffers:', 8) || left('  ', 30, ' ') || left('RS', 2),
  || left('  ', 5, ' ') || left('Direct Optimized (DO) SMB parms:', 32)
bf.5 = left('obs', 3) right('Job name', 8) left('Run date', 11),
  left('Elapsed time', 14) left('Cluster/Component name', 30),
  right('Records', 9) right('ret'ved', 8),
  right('Excip', 6) left('mode', 4),
  left('bias', 4) left('SMB', 5),
  left('used user space data index', 31),
  left('bit fixed', 10) right('vsp', 5),
  right('hwt', 5) right('b31', 5),
  right('cb31', 5) right('ivs', 5),
  right('rer', 5) right('mer', 5) right('hyp', 5)
bf.6 = left('  ', 205, '  ')
"EXEClO * DISKW s64bf (STEM bf.)"
```
if rty <> '40'x then do /* record type */
smfdate = substr(c2x(substr(x.i,7,4)),3,5) /* unpack smf date */
smftime = smf(c2d(substr(x.i,3,4))) /* decode smf time */
term = c2d(substr(x.i,3,4)) /* termination time */
jobn = substr(x.i,15,8) /* jobname */
rst = smf(c2d(substr(x.i,23,4))) /* decode rst time */
init = c2d(substr(x.i,23,4)) /* initiate time */
rsd = substr(c2x(substr(x.i,27,4)),3,5) /* unpack rsd date */
/* Situation indicator */
/* */
rin = x2b(c2x(substr(x.i,39,1)))
z1 = substr(rin,1,1)
z2 = substr(rin,2,1)
z3 = substr(rin,3,1)
z4 = substr(rin,4,1)
z5 = substr(rin,5,1)
z6 = substr(rin,6,1)
z7 = substr(rin,7,1)
if z1 =1 & z6 =1 then sit='Close on Abend'
else if z1 =1 then sit='Component closed'
else if z2 =1 then sit='Vol switched'
else if z3 =1 then sit='No space avail'
else if z4 =1 then sit='Cat or CRA rec'
else if z5 =1 then sit='Closed type=t'
else if z6 =1 then sit='Abend process'
else if z7 =1 then sit='Close VVDS or ICF'
else sit='logic error'
/* Indicator of component being processed */
/* */
dty = x2b(c2x(substr(x.i,40,1))) /* dataset attributes */
w1 = substr(dty,1,1) /* component type */
w2 = substr(dty,2,1) /* component type */
w3 = substr(dty,3,1) /* file format */
w4 = substr(dty,4,1) /* file compression */
w5 = substr(dty,5,1) /* rls */
w6 = substr(dty,6,1) /* rls : mmf */
w7 = substr(dty,7,1) /* file addressibility */
select
when w1 =1 then comp = 'Data'
when w2 =1 then comp = 'Index'
end
select
when w3 =1 then form = 'Extended format'
otherwise form = 'Standard format'
end
select
when w4 =1 then com = 'Compressed'
otherwise com = 'Non compressed'
end

select
when w5 =1 then rls  = 'RLS in effect             '
when w6 =1 then rls  = 'RLS in effect MMF disabled'
otherwise       rls  = 'Non rls       '
end

select
when w7 =1 then addr = 'Extended addressable ds'
otherwise       addr = 'Standard addressibility'

end

dnm = strip(substr(x.i,85,44))            /* dataset name         */
hlq = substr(dnm,1,3)                     /* ds hlq construct     */
hlqt= substr(dnm,1,11)                    /* test  ds hlq         */
chr = c2d(substr(x.i,131,4))              /* current high rba/ci  */
esl = c2d(substr(x.i,135,2))              /* extent segment length*/
#extents = esl / 26                       /* no. of extents       */
offset = 135 + 2 + esl
sln = c2d(substr(x.i,offset,4))           /*  stat.segment length */

/*------------------------------------------------------------------
/*  Selection filtering by: dsn or job name (sample)                */
/*------------------------------------------------------------------
IF (hlq ¬= "SYS") & (hlq ¬= "DFH") & (hlq ¬= "BET") &,
( hlq ¬= "QM5") & ( hlq ¬= "CIC") &,
( hlq ¬= "CAT") & ( hlq ¬= "BK.") &,
( jbn ¬= "CICSPROD") & ( jbn ¬= "CICSTEST") &,
( jbn ¬= "CICSDEV") & ( comp = 'Data')
/*

/*------------------------------------------------------------------
/*  Figure 1 selection filters used: dsn, job name, close status     */
/*------------------------------------------------------------------
IF hlqt="PROD.HISTFI" &  jbn = "MYJOB   "  & (z1 =1)
Then do
select ;
when sln > 28Ø
then do ;

/*------------------------------------------------------------------
/*  Statistics Section at OPEN Time                                */
/*------------------------------------------------------------------

nil = c2d(substr(x.i,offset+4,4))    /* # of index levels     */
nex = c2d(substr(x.i,offset+8,4))    /* # of extents          */
nlr = c2d(substr(x.i,offset+12,4))   /* # of records          */
nde = c2d(substr(x.i,offset+16,4))   /* # of deletes          */
nin = c2d(substr(x.i,offset+20,4))   /* # of inserts          */
nup = c2d(substr(x.i,offset+24,4))   /* # of updates          */
nre = c2d(substr(x.i,offset+28,4))   /* # of retrieves        */
ncs = c2d(substr(x.i,offset+32,4))   /* # of ci splits        */
nas = c2d(substr(x.i,offset+36,4))   /* # of ca splits        */
nep = c2d(substr(x.i,offset+40,4))   /* # of excp count       */

/*------------------------------------------------------------------
/*  Change in Statistics from OPEN to time of EOV and CLOSE       */
/*------------------------------------------------------------------

dil = c2d(substr(x,i,offset+48,4)) /* # of index levels chg. */
dex = c2d(substr(x,i,offset+52,4)) /* # of extents chg. */
drl = c2d(substr(x,i,offset+56,4)) /* # of records chg. */
nde = c2d(substr(x,i,offset+60,4)) /* # of deleted chg. */
din = c2d(substr(x,i,offset+64,4)) /* # of insert chg. */
dup = c2d(substr(x,i,offset+68,4)) /* # of update chg. */
dre = c2d(substr(x,i,offset+72,4)) /* # of retrieve chg. */
dcs = c2d(substr(x,i,offset+80,4)) /* # of ci splits chg. */
das = c2d(substr(x,i,offset+84,4)) /* # of ca splits chg. */
dep = c2d(substr(x,i,offset+88,4)) /* # of excp chg. */

dbs = c2d(substr(x,i,offset+92,4)) /* physical blocksize */
dci = c2d(substr(x,i,offset+96,4)) /* control interval size */
dls = c2d(substr(x,i,offset+100,4)) /* max. logical rec length */
dkl = c2d(substr(x,i,offset+104,2)) /* key length */
ddn = substr(x,i,offset+106,8) /* dd name */
str = c2d(substr(x,i,offset+114,1)) /* string number */
plh = c2d(substr(x,i,offset+116,2)) /* # of concurrent strings */
plc = c2d(substr(x,i,offset+118,2)) /* # of concurrent strings */
plh = c2d(substr(x,i,offset+116,2)) /* # of concurrent strings */
plc = c2d(substr(x,i,offset+118,2)) /* # of concurrent strings */
bno = c2d(substr(x,i,offset+120,2)) /* buffer space */
bfs = c2d(substr(x,i,offset+122,2)) /* data buffers */
bin = c2d(substr(x,i,offset+124,2)) /* index buffers */

c = x2b(c2x(substr(x,i,offset+170,1)))
acbk = substr(mc1,1,1) /* access data via index? key_access */
acbad = substr(mc1,2,1) /* access without index? rba_access */
acbcnv = substr(mc1,3,1) /* control interval processing? */
acbseq = substr(mc1,4,1) /* sequential processing? */
acbd = substr(mc1,5,1) /* direct processing? */
acbin = substr(mc1,6,1) /* input/get/read? */
acbout = substr(mc1,7,1) /* output/put/write? */
acbuff = substr(mc1,8,1) /* user buffers? */

if (acbout = 1) & (acbin = 1) then open = 'inout'
else if acbout = 1 then open = 'output'
else open = 'input'

mc2 = x2b(c2x(substr(x,i,offset+171,1)))
acbskp = substr(mc2,4,1) /* skip sequential processing */
acblog = substr(mc2,5,1) /* logon indicator */
acbrst = substr(mc2,6,1) /* dataset to empty state */
acbdns = substr(mc2,7,1) /* shared_control_blocks */
acbaix = substr(mc2,8,1) /* path_aix */

if (acbd = 1) & (acbd = 1) then mode = 'mix'
else if acbd = 1 then mode = 'dir'
else if acbskp = 1 then mode = 'skp'
else
  mode = 'seq'

/* Third ACB MACRF flag byte */
/*-------------------------------------------------------------*/
mc3 = x2b(c2x(substr(x.i, offset + 172, 1)))
acblsr = substr(mc3, 2, 1) /* local shared resource */
acbgsr = substr(mc3, 3, 1) /* global shared resource */
acbici = substr(mc3, 4, 1) /* improved ci processing */
acbdfi = substr(mc3, 5, 1) /* deferred write */
acbgsis = substr(mc3, 6, 1) /* sequential insert strategy */
acbnsis = substr(mc3, 7, 1) /* sequential insert strategy */
acbnclis = substr(mc3, 8, 1) /* fixed control blocks */
acbmode = substr(mc3, 8, 1) /* vsam 31 bit addressing */
select
  when acblsr = 1 then shr = 'lsr'
  when acbgsr = 1 then shr = 'gsr'
  otherwise shr = 'nsr'
end
select
  when acbnclis = 1 then fix = 'yes' /* cont. blocks & buffers */
  otherwise fix = 'no' /* fixed in real storage */
end
select
  when acbmode = 1 then bufa = '31' /* buffer addressing mode */
  otherwise bufa = '24'
end
select
  when ACBSIS = '1' then ins = 'SIS' /* insert strategy used */
  otherwise ins = 'nis'
end

/* Fourth ACB MACRF flag byte */
/*-------------------------------------------------------------*/
mc4 = x2b(c2x(substr(x.i, offset + 173, 1)))
acbrls = substr(mc4, 1, 1) /* rls processing */
acbsnp = substr(mc4, 2, 1) /* snp option */
mc43 = substr(mc4, 3, 1) /* reserved */
mc44 = substr(mc4, 4, 1) /* reserved */
mc45 = substr(mc4, 5, 1) /* reserved */
mc46 = substr(mc4, 6, 1) /* reserved */
mc47 = substr(mc4, 7, 1) /* reserved */
mc48 = substr(mc4, 8, 1) /* reserved */

/* SMB Restrictions */
/* MACRF parameters not supported are: */
/* UBF(USER buffering), ICI(Improved Control Interval processing), */
/* GSR(Global Shared Resources), LSR(Local Shared Resources), */
/* RLS(Record Level Sharing), AIX(Alternate Index) - pre z/OS 1.3 rel.* */
/* non-extended format VSAM files. */
if acbubf = 1 then note='USER buffering'
else if acbici = 1 then note='ICI processing'
else if acbaix = 1 then note='Alternate Index'
else if shr ¬= 'nsr' then note='NSR required'
else if acbrls =1 then note='RLS processing'
else if w3 =Ø then note='Extended format required'
else note='none'

/* SMB ACCESS BIAS Information */

smb = x2b(c2x(substr(x.i,offset+174,1)))
s1 = substr(smb,1,1) /* accbias via jcl*/
s2 = substr(smb,2,1) /* accbias via smb*/
s3 = substr(smb,3,1) /* bias=do used */
s4 = substr(smb,4,1) /* bias=so used */
s5 = substr(smb,5,1) /* bias=sw used */
s6 = substr(smb,6,1) /* bias=dw used */
s7 = substr(smb,7,1) /* bias=co used */
s8 = substr(smb,8,1) /* bias=cr used */

/* The way of SMB invocation ? */

select
  when s1 ='1' then smb='jcl'
  when s2 ='1' then smb='sys'
  otherwise smb='none'
end

/* Kind of SMB optimization technique used ? */

select
  when s3 ='1' then bia='do'
  when s4 ='1' then bia='so'
  when s5 ='1' then bia='sw'
  when s6 ='1' then bia='dw'
  when s7 ='1' then bia='co'
  when s8 ='1' then bia='cr'
  otherwise bia='none'
end

/* SMB DO Information */

rsc = x2b(c2x(substr(x.i,offset+175,1)))
vsp = substr(rsc,1,1) /* do with smbvsp */
hwt = substr(rsc,2,1) /* do with smbhwt */
b31 = substr(rsc,3,1) /* remode31=buff used */
cb31 = substr(rsc,4,1) /* remode31=cb used */
ivs = substr(rsc,5,1) /* do: insufficient vs */
rer = substr(rsc,6,1) /* do: reduced resource */
mer = substr(rsc,7,1) /* do: minimum resource */
hyp = substr(rsc,8,1) /* do:some or all hyperspace buffers*/
if comp="Index" then buf =bfi
else if compont="Data" then buf =bfd
elapstm = smf(term-init)
/*------------------------------------------------------------------*/
/* File processing & attribute report */
/*------------------------------------------------------------------*/
fa=right(i,3,'Ø') right(jbn,8) right(date('n',rsd,'j'),11),
left(elapstm,14) left(dnm,30) right(DEP,6),
right(nlr,9) right(dre,8) right(dde,6),
right(din,6) right(dup,6) right(acbkey,3),
right(acbadr,3) right(acbcnv,3) right(acbici,3),
right(dls,5) right(dkl,3) right(DCI,5),
right((nil + dil),4), /* index levels at the end of run */
right((ncs + dcs),10), /* ci splits at the end of run */
right((nas + das),10), /* ca splits at the end of run */
right(ins,5),
right((nex + dex),5), /* extents at the end of run */
left(' ',1') left(addr,8) left(form,8) right(shr,4),
left(' ',2') left(note,24)
PUSH fa
"EXECIO 1 DISKW s64fa"
/*------------------------------------------------------------------*/
/* File buffers & buffering management report */
/*------------------------------------------------------------------*/
bff= right(i,3,'Ø') right(jbn,8) right(date('n',rsd,'j'),11),
left(elapstm,14) left(dnm,30) right(nlr,9),
right(dre,8) right(dep,6) right(mode,4),
left(bia,4) left(smb,4),
right(bno,5) right(acbubf,5) right(bsp,6),
right(bfd,5) right(bfi,5) right(bufa,5),
right(fix,5) right(vsp,5) right(hwt,5),
right(b31,5) right(cb31,5) right(ivs,5),
right(rer,5) right(mer,5) right(hyp,5)
PUSH bff
"EXECIO 1 DISKW s64bf"
end
otherwise do ;
say 'REXX program logic in error !'
exit
end
end
drop x.
/* Close & free all allocated files */
"EXECIO Ø DISKW s64fa(FINIS"
"EXECIO Ø DISKW s64bf(FINIS"
say
CONCLUSION

It is true that SMB may not be the answer to all application program buffering requirements. Its main purpose is to provide a system capability for improving performance buffering options for batch application processing beyond those provided by the standard defaults. However, if you haven’t implemented System Managed Buffering yet, the recommendations in this article can be applied to any VSAM file type, and the performance improvements for batch processing will be remarkable. If your datasets are currently in EXTENDED format, or will be converted to EXTENDED in the near future, you should be able to implement System Managed Buffering, and you will be able to achieve even better performance than is available with NSR buffering. As in all other cases, one shouldn’t make significant changes in a production dataset’s allocation unless they have been thoroughly tested.
REFERENCES

z/OS DFSMS: Using Data Sets (SC26-7410)
z/OS V1R3: DFSMS Technical Guide (SG24-6569-00)
VSAM Demystified (September 2003 edition) (SG24-6105)

Mile Pekic
Systems Programmer (Serbia) © Xephon 2004

JCL tips – part 2

In this article I will concentrate on how we use datasets in our jobs and show which are the useful JCL statements and the parameters and/or options we can use to make our lives easier. Therefore, I will give some general tips for datasets, as well as tips for datasets that reside on tape, and special ones such as Generation Data Group (GDG) datasets.

DD STATEMENTS IN GENERAL

Often we just want to test our jobs without actually processing any data. Sometimes we want to test different scenarios, like what will happen if some of our input files are empty. For that we can use dummy files.

A file can be assigned dummy status by coding either DUMMY as the first parameter in the DD statement or by changing the real DSN to DSN=NULLFILE. Personally, I prefer the first choice, because then I keep the real DSN value in the JCL. After testing, I just need to delete all occurrences of the word ‘dummy’.

Here are some situations where one can use a dummy file.

A dummy file is always accepted from the JCL as an empty file, so to test the program flow to see whether it recognizes empty files, we can use something like this:
By coding DUMMY for INPFILE2, that file becomes empty.

In another situation, you could have several files concatenated in one logical file for your program. You might want to test your program with just some of them. Using DUMMY status you can do something like this:

By coding the third file as a DUMMY file, you not only make that file empty but also create EOF status for INPFILE1, so file 4 and file 5 will not be processed.

So, note that putting DUMMY for a file will not only cause that file to be skipped, as perhaps you expected, but will also terminate the set of files being read in.

DUMMY files can also be used for output files as well as for input files, although the usage is a little different.

Let's say that while we are testing we do not want to save some of the output files, or if our report is too long, and we already
know the content, we do not want to save it; here is what we do:

```jcl
// RUNJ CL  JOB (ACCOUNT), CLASS=B, MSGCLASS=Z, NOTIFY=&SYSUID
// *
// STEP1   EXEC PGM=PROG1
// INPFILE1 DD DSN=APPLID.INPFILE1.TST, DISP=SHR
// INPFILE2 DD DSN=APPLID.INPFILE2.TST, DISP=SHR
// OUTFILE1 DD DUMMY, DSN=APPLID.OUTFILE1.TST,
//            DISP=(NEW,CATLG),
//            SPACE=(CYL,(10,5),RLSE),
//            DCB=(LRECL=80,RECFM=FB,BKSIZE=0)
// REPORT   DD DUMMY, SYSPRINT=A
// SYSPRINT DD SYSPRINT=*          
// *
```

Any output produced for REPORT or OUTFILE1 is suppressed. It means that no file is created and device and space allocations are ignored.

**DISP PARAMETER MOD STATUS FOR FILES**

We all know that by using the MOD value for the status of a file we can add new records to an existing file. Personally, I do not like to use MOD in that way because improper usage can be very difficult to resolve. For instance, the order in which jobs are run is very important, and you must be very careful that new cycles always start with the job that created the file. But there are some situations where I find MOD status very useful.

Let’s say that you have a job with file dependency – your job cannot start if some files are not defined on the system. In other words, you expect some file to come before your job can proceed. It can be from some other application. If that file never comes, your batch stream is stopped; it cannot continue.

Even if your job is time triggered – meaning it must start at some specific time – without checking on the existence of the file, the job itself will abend if the file is not there when the job starts.

We can overcome this problem. Prior to your job, you can have a step using IEFBR14 and a list of the files you expect – something like the following:

```jcl
// RUNJ CL  JOB (ACCOUNT), CLASS=B, MSGCLASS=Z, NOTIFY=&SYSUID
```
Your job will first check whether the expected files are defined. If yes, then nothing happens, but if the files are not defined then using DISP=(MOD,CATLG) will create empty files, so subsequent steps will run correctly.

This is specially important if we process data in cycles based, for example, on dates. If we miss some of the dates this causes problems for other jobs too (eg report jobs that assume all files already exist).

TAPES

There are several reasons to use tapes instead of disks as data storage media. Let's list some of them:

1 Data is coming from outside of your data centre and it is the only possible way for your customers to supply you with data.

2 Your application produces output files that you need to keep from every cycle for some time and you have limited disk space.

3 Files you create are very big, so it is much better to keep
them on tape than on disks, since you do not have enough disk space.

4 Security reasons can also be very important. Maybe your data must be protected, and one way to keep it safe is to keep it out of the system.

There are some others, I am sure, but let’s stop here. To use tapes as your storage medium you must be familiar with some of the parameters that JCL uses.

Number of tapes for output file
If you have an output file and you do not know how big it will be, you can find that your job returns unexpected results and you will be wondering why. Well, the most likely reason is that your output file spreads over more than the default value of five tapes. If you already know that you have a file that needs more than five tapes, or you think that it is possible, you need to use the volume-count subparameter in the VOLUME parameter of the DD statement.

Code a volume count when a new dataset will reside on six or more volumes. If you omit the volume count or if you specify 1 to 5, the system allows up to five volumes; if you specify 6 to 20, the system allows 20 volumes; if you specify a count greater than 20, the system allows 5 plus a multiple of 15 volumes. Here is an example:

```//OUTFILE DD DSNAME=APPL.OUTFILE.TST,DISP=(NEW,CATLG,KEEP), // UNIT=TAPE,VOLUME=(,,,6,SER=(11111,22222))```

The DD statement defines a new file, which will reside on two tapes, serial numbers 11111 and 22222. The VOLUME volume-count subparameter requests six volumes, if required. Thus, if more space is required, the system can assign a third and fourth volume. Without the volume-count value, the system will stop allocating new tapes after finishing with tape 5.

De-allocating files from a job
Once we submit our job, the first thing to be done by the system
is to allocate all the required resources (files, devices, programs, etc). All of these resources have a status of ‘in use’ until the end of the job. Some of them are shareable, some of them are not. Files that reside on tapes cannot be shareable. So it can be very important to make these files available as soon as possible. As an example, if there are several steps in the job and our file on tape is used in just the first one, it is not available to other jobs until the currently-running job ends.

In addition, if we have several files on tapes in our job, there can be a problem with the number of tape drives available to the others. As soon as we de-allocate a tape drive, it is available to the other jobs.

De-allocating can be done using the FREE parameter of the DD statement. By coding FREE=CLOSE, once the file has been used and closed, the system will dynamically de-allocate the dataset.

```
// TAPE1  DD DSNAME=APPL.TAPE1.TST,DISP=OLD,FREE=CLOSE,
//          UNIT=TAPE
```

In this example, the FREE=CLOSE parameter makes JES de-allocate the dataset, de-queue it, and make it available to other jobs as soon as it is closed.

This can be especially useful if the same file is used in an online environment like CICS or IMS.

**Allocating the same tape drives to more than one file**

It is not a rare situation for your application to receive data from several other applications. If input files from them are on tape and you need to process them as one file, you can append all of them into one, or you can process them all together using concatenation of files in the DD statement. If you choose the first approach you will lose both time and as many tapes as all these files occupy. Sometimes you will need as many tape drives as there are files, because the system allocates all the necessary drives at the beginning of the job. In addition, there can be other requests for drives that you have in your system and that can create a problem for all jobs, not just yours.
Well, it doesn’t need to happen if you use the AFF value for the UNIT parameter. What AFF does is request the system to allocate different files residing on different removable volumes on the same device during the execution of the step. Let’s see it in an example:

```
//INFILE DD DSN=APPLID.TAPE1.TST, DISP=SHR
//       DD DSN=APPLID.TAPE2.TST, DISP=SHR, UNIT=AFF=INFILE
//       DD DSN=APPLID.TAPE3.TST, DISP=SHR, UNIT=AFF=INFILE
//       DD DSN=APPLID.TAPE4.TST, DISP=SHR, UNIT=AFF=INFILE
//       DD DSN=APPLID.TAPE5.TST, DISP=SHR, UNIT=AFF=INFILE
```

All five tapes (SMS catalogued) will use the same tape drive unit as the first one, and therefore they will be mounted one at a time. Without using AFF, the system will try to allocate five units.

**Mounting tapes when we need them**

Mounting and de-mounting tapes from tape drives takes some time. If we want to save every single second, we can avoid mounting the tape before we know we need it in our program. So if we know that the logic in our program needs data from tape, but in very specific and rare situations, we should have something like the example below. It will save time for us and tape drives for other jobs.

To delay a tape mount, there is another value for the UNIT parameter. It is DEFER. It asks the system to assign the dataset to a device(s), but it requests that the volume(s) not be mounted until the dataset is opened. To defer mounting, DEFER must be specified or implied for all DD statements that reference the volume.

```
//INFILE DD DSNNAME=APPL.INFILE.TST, DISP=OLD,
//       VOLUME=SER=1095, UNIT=(TAPE,, DEFER)
```

This DD statement defines an existing dataset that resides on a tape volume, and requests that the system assign a tape device. Because DEFER is coded, the volume will not be mounted until the dataset is opened.
GDG FILES

GDG files can be very useful. If we have an application that needs to keep several old versions of files, perhaps for some weekly or monthly processing, it is a good option to use GDG files. But when we use them we need to be careful how we use them. Here are some tips.

Allocating a new generation in a multiple-step job

If you have one job with several steps and during that job you allocate a new generation in your GDG you need to be careful about using that GDG in subsequent job steps. Maybe it does not looks so obvious, but the rules for using relative numbers within GDG are changed in that situation. After we allocate a new generation (by saying (+1) in that step) we need to continue to use (+1) if want to keep processing that generation. This is because it is a new generation if we look from the beginning of the job. In the same way, if we want to process the previous generation to the one just created, we need to use (0) and so on.

If a job fails after the step that creates a new generation and we want to resubmit the job, we need to change all steps after that one before we resubmit. Now (+1) must be changed to (0), (0) to (-1), and so on.

Contention

The next situation we need to think about is when we have a GDG defined in a multi-programming environment. The rule is that as long as one job wants to create a new GDG generation, no other job can use it or create any other because the system does not know what is the current generation. The system does not know the result of the already-running job – will it fail, will it delete the last generation, or something else? – so the system cannot establish which would be the generation to use in some other job.

The solution is to use absolute generation and version numbers, but by doing that we lose all the benefits of GDG files.
IXFP Snapshot performance tips

In an earlier article I have provided a brief summary of gains obtained by implementing Snapshot into batch production (see *The effects of implementing Snapshot copy*, *MVS Update*, issue 183, December 2001). The main benefits of implementing Snapshot were only implicitly noted – Snapshot saves time (by replacing traditional dump and image copy techniques, and cutting hours from your back-up window), saves money (savings in media, CPU, and channel resources), improves business productivity (better application availability, disaster protection), as well as improving quality (more thorough applications testing by allowing you to re-engineer your test processes to achieve greater efficiency and reduce costs).

To get the most out of Snapshot as quickly as possible, I would recommend that a planning session be held in order to analyse your existing data duplication processes to identify those areas where Snapshot will deliver the greatest initial benefit. Use a team of experts from application programming, data management, job scheduling, and systems programming who know the time-critical applications, the possible timing problems with the dataset back-up procedures, the most time-consuming jobs, and whether those jobs are in the critical path of batch processing. There are a number of areas in which Snapshot is being used. Here are some of the most typical examples:

- Deferring back-ups – as all of us have noted, demands for
increased system availability mean that the back-up window is shrinking and consequently the time slot when system back-ups are performed gets shorter. Snapshot is a great tool that helps alleviate this problem by taking a point-in-time copy of all data to be backed up.

- Improving batch processing – Snapshot can be used to replace existing checkpoint processing in batch job streams. The benefit of this is twofold – the batch will run faster because the Snapshot back-ups will be extremely quick; on the other hand, if the batch job fails and backout must be performed, then the original datasets can be restored by snapping back over the original datasets.

- Application testing – when performing testing at application level Snapshot can be used to replicate an entire application within minutes. If testing destroys any data, it can quickly be recreated.

- Data warehousing – Snapshot’s ability to replicate large volumes of data quickly means that data archive copies of production databases can be produced very efficiently.

The *Implementing SnapShot* (SG24-2241) Redbook provides detailed technical and operational guidance for implementing Snapshot in a variety of workloads and environments. It makes recommendations on the different ways you can implement Snapshot and positions Snapshot against current data duplication techniques and tools.

After you introduce Snapshot, you can try to identify Snapshot candidates. You may ask your IBM representative for a copy of the SNAPAID program (available on the IBM MKTTOOLS disk). This program identifies the potential benefits in terms of CPU and elapsed time savings that would accrue from replacing data copying programs with Snapshot. The program has a built-in list of standard copying programs. Other programs can be specified through input parameters. It uses SMF data as input and therefore accurately calculates the data. Parameters are used to limit the time analysed or to limit the analysis to specific jobs.
What makes Snapshot unique and not just another copying tool, such as DFDSS, is that Snapshot is a virtual data duplicator that enables you to make a copy of data by copying the logical pointers to the data. Because no data movement takes place, it is a very fast process and uses no additional disk space to make the copy.

SNAPSHOT PERFORMANCE TIPS

Indeed, Snapshot is extremely quick, as advertised. The performance relates to the number of extents being snapped, and depends very little on the number of tracks involved. If you are snapping an entire volume, it actually does take only a fraction of a second. There are a lot of factors that go into this time, such as the model/speed of the RVA, type of channels, other I/O and CPU activity, so it is difficult to quote any useful numbers. As we did a lot of testing of Snapshot performance when we were building Snapshot support for batch production job streams we realized that Snapshot performance can be improved by following IBM’s recommendations. These recommendations were as follows:

1. Performance of Snapshot can be improved if an alternative SIBSTK.xx PARMLIB member is used (as described in the IXFP Installation for MVS manual, Step 19) and the SET DEST and SET ECAMDEV commands are not used for Snapshot jobs only. IBM’s resolution: Snapshot does not require an ECAM device. If the SET DEST command is specified, it can act as a bottleneck on allocation and slow the Snap process down.

2. When using the Snapshot product to make a copy of a dataset, the copy that results shares data in common with the original. This shared data is pointed to by both the original and the copy. It is reflected in the REPORT SPACEU under the SHARED column. When the original or the copy gets updated, less of the data is ‘shared’ because the new data is rewritten to another location. For VSAM or any pre-formatted dataset, the share will not decrease as we
expected because of the structure of those datasets being pre-defined at allocation time. If release is not specified in the JCL, even though we no longer have access to the data, the space it occupied is still defined as belonging to both the original and the Snap. The SHARED STORAGE column in the REPORT SPACEU is not updated unless RLSE is specified inside the SPACE parameter. The SHARED statistic may also not be correct until an INTERVAL DDSR job has released the space in some cases. Recommendation: use the RLSE parameter.

3 The performance of Snapshot jobs can be improved with the use of the STKPARMS DD card. The STKPARMS DD permits Snapshot to access modules it needs more quickly than by searching for them. Without the STKPARMS DD, Snapshot has to allocate and read SYS1.PARMLIB as part of the Snap process. With it, it is already allocated before the Snap starts. Recommendation: use the STKPARMS DD card.

4 Although it is convenient to use a standard control card for all Snap jobs by using the INDD card, it is also slower than using the SOURCE parameter. Using SOURCE and TARGET will take more effort to program, but it will reduce Snap job time. Recommendation: use the SOURCE and TARGET parameters.

5 Defect support does not handle performance issues, but IBM has noticed the following: if running concurrent Snap jobs, there is an entry in the SIBSTK00 parmlib member for command SET DEST. If this command is used, there can be a performance impact because multiple jobs attempt to update the same logging file as specified in SET DEST. Recommendation: remove the entry in SIBSTK00 for SET DEST.

COLLECTING SNAPSHOT PERFORMANCE DATA AND REPORTING

At user-defined intervals, IXFP requests performance and space utilization statistics from the RVA. This data is written as
SMF records to the host MVS system. Through the use of SAS procedures supplied with IXFP (or user-written routines) this information can be processed to produce trend analysis reports. To use XSA/Reporter, you may need additional software: SAS if you want to use SAS to produce reports or to summarize the data with XSA/Reporter; SAS/GRAPH if you want to produce graphic displays of the reported data. If you do not have SAS installed, you can use other utilities to produce reports from the collected data.

To enable SMF to collect IXFP data, you must identify the SMF record type and subtype for IXFP in the SMFPRMxx member of SYS1.PARMLIB. An example of an SMFPRMxx member, where 250 is the record type selected for IXFP data (default), and XSA/Reporter, DDSR and space utilization are to be written, is as follows:

```
SYS(TYPE(Ø:104, 250))
SUBSYS(STC, TYPE(Ø:55, Ø8:104, 250))
SUBPARM(IXFP(250, 2, 5, 7, 8))
```

The IXFP SMF record can contain eight different subtypes, depending on the event that took place in the RVA subsystem. IXFP Version 2 introduces a new IXFP SMF record subtype 8 and an updated record subtype 7.

Snapshot events are recorded in the original subtype 6. With Snapshot 1.2, the Snapshot events recorded in subtype 6 are left unchanged. However, the new IXFP SMF subtype 8 record, the extended event record, is designed for VSAM dataset support. All information in SMF record subtype 6 is copied to SMF record subtype 8, and additional information is added for VSAM, new keywords, and additional extents. If you are running Snapshot for the first time you should specify only subtype 8 – there is no need to use both 6 and 8.

The subtype 8 record is a segmented SMF record. Each record consists of the SMF record header, the basic segment, and one of four segments attached to it. The four segments are:

- Cluster Definition Segment, which contains the names of
the source and target VSAM clusters, the cluster organization, and the path name.

- Data Set Name Segment, which contains the snapped dataset names, and the DSNTYPE from the catalog if the command is SNAP DATASET.

- Snapped Extents List Segment, which contains information from the Snap event, such as timing, return codes, data mover return codes, device addresses, and extents.

- DDSR Extents List Segment, which contains information about the extent.

I recommend that you start by using the subtype 8 record instead of the original subtype 6 record. To start by using the new subtype 8 record, you must update your SMPPRMxx member in SYS1.PARMLIB so that SMF will record subtype 8 of the IXFP SMF user record. The layout and contents of the record are documented in *IBM RAMAC SnapShot for MVS/ESA: Using SnapShot Version 1 Release 2* (SC26-7173-06).

Based on record descriptions obtained from the above mentioned manual, a sample Snapshot performance report writer was written. The code is a two-part stream. In the first part (COPYSMF), selected SMF records (selection being defined by INCLUDE’s condition) are copied from an SMF dataset to a file, which can be used as a base for archived records. In the second part, RSNAP, the captured records are formatted by invoking SNAPREPT EXEC and three reports are produced.

The Snapshot summary report shows date and time of the Snapshot, jobname and user id of the Snapshot requestor, total time used to Snap (in ms), Snap completion RC, Snapshot options used (defined or defaulted), and dataset name (or VSAM component) being snapped.

The two additional reports deal with Snapshot performance of VSAM and non-VSAM files.
### VSAM FILE SNAPSHOT REPORT

<table>
<thead>
<tr>
<th>Snap date</th>
<th>Snap time</th>
<th>Job name</th>
<th>User ID</th>
<th>time</th>
<th>RC</th>
<th>SnapShot options</th>
</tr>
</thead>
<tbody>
<tr>
<td>14 Dec 2003</td>
<td>00:24:06:34</td>
<td>ACCNTNPG</td>
<td>ITOPER3</td>
<td>6039</td>
<td>0</td>
<td>CAT(YES)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HCPYMODE(SHR) REPL(YES)</td>
</tr>
</tbody>
</table>

Source cluster: ACCNT.HISTORY.CI29
Target cluster: ACCNT.HISTORY.CI29.#COPY

Source Data: ACCNT.HISTORY.CI29.DATA
Target Data: ACCNT.HISTORY.CI29.#COPY.DATA

Source Index: ACCNT.HISTORY.CI29.INDEX
Target Index: ACCNT.HISTORY.CI29.#COPY.INDEX

<table>
<thead>
<tr>
<th>Extent #:</th>
<th>(Data) Ext. Snap:</th>
<th>Source Vol:</th>
<th>Target Vol:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1706</td>
<td>PSR4C0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1374</td>
<td>PSR464</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2157</td>
<td>PSR4C4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1087</td>
<td>PS1558</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>733</td>
<td>PSR468</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>442</td>
<td>PSR488</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1853</td>
<td>PS1561</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>PS1561</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1853</td>
<td>PSR481</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>PSR481</td>
<td></td>
</tr>
</tbody>
</table>

### NON-VSAM FILE SNAPSHOT REPORT

<table>
<thead>
<tr>
<th>Snap date</th>
<th>Snap time</th>
<th>Job name</th>
<th>User ID</th>
<th>time</th>
<th>RC</th>
<th>SnapShot options</th>
</tr>
</thead>
<tbody>
<tr>
<td>13 Dec 2003</td>
<td>14:01:38:31</td>
<td>ACCNPRPN</td>
<td>ITOPER9</td>
<td>916</td>
<td>0</td>
<td>CAT(YES)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HCPYMODE(SHR) REPL(YES)</td>
</tr>
</tbody>
</table>

Source Dsn: ACCNT.FORREV.STAL
Target Dsn: BK.ACCNT.FORREV.STAL.T1401.DPRPD


Source dev.: 1201 Target dev.: 1092
Extent beg.: 1179648 Extent beg.: 2097152
Extent end.: 1179662 Extent end.: 2097166

Job summary:
Snap date   Snap time   Job name  User ID  time   RC   SnapShot options
------------------------------------------------------------------------
HCPYMODE(SHR) REPL(YES)

Source Dsn: ACCNT.REP.STAL
Target Dsn: BK.ACCNT.REP.STAL.T0822.DAV07
Extent #: 1 Ext. Snap: 300 Source Vol : PSPR70 Target Vol : PSR44D
Source dev.: 1210 Target dev.: 1101
Extent beg.: 37486592 Extent beg.: 21954560
Extent end.: 39780366 Extent end.: 24248334

Extent #: 2 Ext. Snap: 0 Source Vol : PSPR70 Target Vol : PSR44D
Source dev.: 1210 Target dev.: 1101
Extent beg.: 69140480 Extent beg.: 24313856
Extent end.: 68493070 Extent end.: 23666446

Extent #: 3 Ext. Snap: 0 Source Vol : PSPR70 Target Vol : PSR44D
Source dev.: 1210 Target dev.: 1101
Extent beg.: 82779904 Extent beg.: 23731968
Extent end.: 18759182 Extent end.: 59711246

JCL

//JOB  JOB ...  
//COPYSMF EXEC PGM=ICETOOL, REGION=0M
//TOOLMSG DD SYSOUT=X
//DFSMSG DD SYSOUT=X
//OUT DD SYSOUT=X
//SMF DD DISP=SHR, DSN=your.smf.vbs.file
//SMFC DD DSN=uid.SMF2508.file, SPACE=(CYL,(15,15)),
//DISP=(NEW,CATLG,KEEP), UNIT=SYSDA,
//DCB=(RECFM=VB, LRECL=32756, BLKSIZE=32760)
//TOOLIN DD *
//COPY FROM(SMF) TO(SMFC) USING(SMFI)
/*
//SMFICNTL DD *
//OPTION COPY, VLSHRT
//INCLUDE COND=(6, 1, BI, EQ, X'FA', AND, 23, 2, BI, EQ, X'0008')
/*
//RSNAP EXEC PGM=IKJEFT01, REGION=0M
//SYSEXEC DD DISP=SHR, DSN=your.rexx.library
//SMF DD DISP=SHR, DSN=uid.SMF2508.file
//SYSTSPTRT DD SYSOUT=* 
//SYSTSIN DD *
PROF NOPREF
%SNAPREPT

SNAPREPT EXEC

/* REXX EXEC to read and format SMF Snapshot 250.8 record */
/* */
/* SIBSMFHD: IXFP Subsystem SMF Record Header */
/* Version 2 Release 1 Modification level 1 */
/* Map the SMF header for the IXFP SMF record subtypes. */
/* see: hlq.SIBMAC(SIBSMFHD) */
/*****************************************************************************/
signal ON ERROR
Address TSO
/*****************************************************************************/
/* Part 1: Handle file allocation & dataset existence */
/*****************************************************************************/
userid=SYSVAR(SYSUID)
vsam   =userid||'.vsam.rep'             /* VSAM file Snap report */
nvsam  =userid||'.nvsam.rep'            /* Non VSAM Snap report */
summ   =userid||'.summ.rep'             /* Summary Snap report */
x = MSG('OFF')
IF SYSDSN(vsam) = 'OK'
    THEN "DELETE "vsam" PURGE"
IF SYSDSN(nvsam) = 'OK'
    THEN "DELETE "nvsam" PURGE"
IF SYSDSN(summ) = 'OK'
    THEN "DELETE "summ" PURGE"
"ALLOC FILE(SSUMM) DA("SUMM")",
   "UNIT(SYSALLDA) NEW TRACKS SPACE(1,1) CATALOG",
   "REUSE RELEASE LRECL(135) RECFM(F B)"
"ALLOC FILE(SVSAM) DA("VSAM")",
   "UNIT(SYSALLDA) NEW TRACKS SPACE(1,1) CATALOG",
   "REUSE RELEASE LRECL(90) RECFM(F B)"
"ALLOC FILE(SNVSAM) DA("NVSAM")",
   "UNIT(SYSALLDA) NEW TRACKS SPACE(1,1) CATALOG",
   "REUSE RELEASE LRECL(90) RECFM(F B)"
/*****************************************************************************/
/* Part 2: Print report header and labels */
/*****************************************************************************/
tit.1 =left(' ',8,' ')||'Snapshot Record Summary Report'
tit.2 =left(' ',2,' ')
tit.3 =left(' ',8,' ')||'Report produced on',
       ||left(' ',1,' ')||left(date(),11),
       ||left(' ',1,' ')||left(' at ',3,' ')||left(time(),10)
tit.4 = left(' ',42)||left('Snap',4)
tit.5 = left('Snap date',12)||left('Snap time',12),
       ||left('Job name',9)||left('User ID',9),
       ||left('time',7)||left('RC',5),
**Part 3: Read and decode SMF Snapshot (250.8) records**

To read and decode SMF Snapshot (250.8) records, you can use the following code snippet:

```plaintext
Do i = 1 to x.0
    SMFRTY = c2d(substr(x.i,2,1)) /* SMF record number */
    RECTYPE = c2d(substr(x.i,19,2)) /* SMF record subtype */
    IF SMFRTY = '250' Then
        Do
            SMFTIME = smf(c2d(substr(x.i,3,4))) /* TOD record written */
            SMFDATE = substr(c2x(substr(x.i,7,4)),3,5) /* Date rec.written */
            SMFSID = substr(x.i,11,4) /* System ID */
            SMFSSI = substr(x.i,15,4) /* Subsystem id for SMF */
        Enddo
    Enddo
Enddo
```

**SMF Record Subtype 08 Basic Segment**

The basic segment of SMF record subtype 08 can be described as follows:

```plaintext
REQTYPE = c2d(substr(x.i,21,1)) /* Snapshot request type */
VERSION = c2d(substr(x.i,22,1)) /* Version of this subtype */
SEGTYPE = c2d(substr(x.i,23,2)) /* Segment type */
RES01 = substr(x.i,25,4) /* Reserved */
RECTOKEN = c2x(substr(x.i,29,8)) /* SMF record 8 token */
COMPCODE = c2d(substr(x.i,37,4)) /* Snap completion RC */
STRTIME = smf(c2d(substr(x.i,41,4))) /* Snap start TOD */
STRDATE = substr(c2x(substr(x.i,45,4)),3,5) /* Date rec.written */
SNAPTIME = c2d(substr(x.i,49,4)) /* Total time to Snap */
FLAG1 = X2B(substr(x.i,53,1)) /* Snap flag one */
FLAG2 = X2B(substr(x.i,54,1)) /* Snap flag two */
```

**First Snapshot flag byte**

The first snapshot flag byte can be obtained from:

```plaintext
f1 = substr(flag1,1,1)
f2 = substr(flag1,2,1)
```
f3 = substr(flag1,3,1)
f4 = substr(flag1,4,1)
f5 = substr(flag1,5,1)
f6 = substr(flag1,6,1)
f7 = substr(flag1,7,1)
f8 = substr(flag1,8,1)
select
  when f1 = 1 then op1="COPYVOLID(YES) ";
  otherwise        op1=" ";
end
select
  when f2 = 1 then op2="BYPASSACS(YES) ";
  otherwise        op2=" ";
end
select
  when f3 = 1 then op3="CAT(YES) ";
  otherwise        op3=" ";
end
select
  when f4 = 1 then op4="TOLENQ(YES) ";
  otherwise        op4=" ";
end
select
  when f5 = 1 then op5="HCPYMODE(SHR) ";
  otherwise         op5=" ";
end
select
  when f6 = 1 then op6="REPL(YES) ";
  otherwise         op6=" ";
end
select
  when f7 = 1 then op7="DELETE(YES) ";
  otherwise         op7=" ";
end
select
  when f8 = 1 then op8="FORCE(YES)";
  otherwise         op8=" ";
END
option =op1||op2||op3||op4||op5||op6||op7||op8

isNaN = 0;   /* Integer format data size

option = substr(flag,1,1)
select
  when flag = '1' then op = "DEBUG(ON)";
otherw ise ot1=' " '; end select when g2='1' then ot2="TRACE(ON)" ; otherwise ot2=' " '; end select when g3='1' then ot3="VOLUME "; otherwise ot3=' " '; end select when g4='1' then ot4="ESOTERIC "; otherwise ot4=' " '; end select when g5='1' then ot5="TRUNCATION "; otherwise ot5=' " '; end select when g6='1' then ot6="CONDVOL(LBL) "; otherwise ot6=' " '; end opt2 =ot1||ot2||ot3||ot4||ot5||ot6 FLAG3 = c2d(substr(x.i,55,1)) /* Snap flag three */ FLAG4 = c2d(substr(x.i,56,1)) /* Snap flag four */ JOBNAME = substr(x.i,57,8) /* Jobname of Snapshot requestor */ USERID = substr(x.i,65,8) /* User ID of Snapshot requestor */ DATAMOVR = substr(x.i,73,8) /* Name of data mover used, or blank */ STORCLAS = substr(x.i,81,8) /* Storage class name, or blank */ MGMTCLAS = substr(x.i,89,8) /* Management class name, or blank */ DATACLAS = substr(x.i,97,8) /* Data class name, or blank */ ESOTERIC = substr(x.i,105,8) /* MVS esoteric name, or blank */ VOLCNT = c2d(substr(x.i,113,4)) /* Volume count */ SEGNUM = c2d(substr(x.i,217,2)) /* Number of segments of this type */ SEGLEN = c2d(substr(x.i,219,2)) /* Length of this type segment */ SEGOFFST = c2d(substr(x.i,221,2)) /* Offset to start of segment */ com.t = right(date('n',STRTDATE,'j'),11), left(STRTIME,11) right(JOBNAME,8) right(USERID,8), right(SNAPTIME,4) right(COMPCODE,4) left(' ',1) ti.1 = left(' ',42)||left('Snap',4) ti.2 = left('Snap date',12)||left('Snap time',12), ||left('Job name ',9)||left('User ID',9), ||left('time',7)||left('RC',5), ||right('Snapshot options',16) ti.3 = left(' '90,' ') if SEGTYPE = 1 & (SEGOFFST <> 0) & (SEGLEN <> 0) Then do sof=SEGOFFST -3 /******************************************* */ /* Cluster Definition Segment (segtype=1) */
"EXECIO * DISKW SVSAM (STEM ti.)"

SCLSTNAM = substr(x.i, sof, 44) /*Source cluster name.*/
SCLSTID = c2d(substr(x.i, sof+44, 2)) /*Source cluster ID number.*/
   /* Used to relate a cluster name*/
   /* to a set of dataset names*/
TCLSTNAM = substr(x.i, sof+48, 44) /*Target cluster name.*/
TCLSTID = c2d(substr(x.i, sof+92, 2)) /*Target cluster ID number.*/
   /* Used to relate a cluster name*/
   /* to a set of dataset names*/

PATHNAME = substr(x.i, sof+96, 44) /*Path name (for future use)*/
CLSTORG = substr(x.i, sof+140, 8) /*VSAM Cluster organization*/
   
clu.1=com.t||option
   clu.2=left(' ', 22)
   clu.3=left('Source cluster:', 16)||SCLSTNAM
   clu.4=left('Target cluster:', 16)||TCLSTNAM

"EXECIO * DISKW SVSAM (STEM clu.)"

if SEGTYPE = 2 & (SEGOFFST <> 0) & (SEGLEN <> 0) Then
   do
      sof=SEGOFFST - 3
   end
if SEGTYPE = 2 & (SEGOFFST <> 0) & (SEGLEN <> 0) Then
   do
      sof=SEGOFFST - 3
   end

SRCDSN = substr(x.i, sof, 44) /* Name of source dataset */
SRCCLSID = c2d(substr(x.i, sof+44, 2)) /* Source cluster ID number */
   /* Used to relate a cluster name*/
   /* to a set of dataset names */
SRCDSNID = c2d(substr(x.i, sof+46, 2)) /* Source dataset ID number */
   /* Used to relate a dataset name */
   /* to a set of extents */
TRGDSN = substr(x.i, sof+48, 44) /* Target dataset name */
TRGCLSID = c2d(substr(x.i, sof+92, 2)) /* Target cluster ID number */
   /* Used to relate a cluster name */
   /* to a set of dataset names */
TRGDSNID = c2d(substr(x.i, sof+94, 2)) /* Target dataset ID number */
   /* Used to relate a dataset name */
   /* to a set of extents */

DSNTYPE = substr(x.i, sof+97, 1) /* Catalog DSNTYPE field */
if DSNTYPE = "D" then do
   clusd.1=left('Source Data :', 16)||SRCDSN
   clusd.2=left('Target Data :', 16)||TRGDSN
   "EXECIO * DISKW SVSAM (STEM clusd.)"
end

if DSNTYPE = "I" then do
   clusi.1=left('Source Index :', 16)||SRCDSN
   clusi.2=left('Target Index :', 16)||TRGDSN
   "EXECIO * DISKW SVSAM (STEM clusi.)"
if DSNTYPE ≠ "D" & DSNTYPE ≠ "I" then do
  "EXECIO * DISK W SNVSAM (STEM ti.)"
  dsnn.1 = com.t||option
  dsnn.2 = left(' ',22)
  dsnn.3 = left('Source Dsn:',12)||SRCDSN
  dsnn.4 = left('Target Dsn:',12)||TRGDSN
  "EXECIO * DISK W SNVSAM (STEM dsnn.)"
end

dsn.1=com.t||option||left(SRCDSN,44)
  "EXECIO * DISK SSSUMM (STEM dsn.)"
end
if SEGTYPE = 3 & (SEGOFFST <> Ø) & (SEGLEN <> Ø) Then
  do
    do j = Ø to segnum -1
      inc = (SEGOFFST + (j*SEGLEN))- 3
    /*--------------------------------------*/
    /* Snapped Extent List Segment (segtype=3)*/
    /*--------------------------------------*/
    EXTnum = j+1
    EXTFLAG = substr(x.i,inc,1) /*Extent flag options*/
    IOSTIME = smf(c2d(substr(x.i,inc+4,4))) /*Start time of I/O for extent*/
    IOSTDATE = substr(c2d(substr(x.i,inc+8,4)),3,5) /*Start date of I/O*/
    SNAPRET = c2d(substr(x.i,inc+12,2)) /*Snap return code*/
    SNAPRES = c2d(substr(x.i,inc+14,2)) /*Snap reason code*/
    IOMRET = c2d(substr(x.i,inc+20,2)) /*I/O interface return code*/
    DMOVERET = c2d(substr(x.i,inc+22,2)) /*Data mover return code*/
    DMOVERES = c2d(substr(x.i,inc+24,2)) /*Data mover reason code*/
    CTNTOKEN = c2d(substr(x.i,inc+26,2)) /*Snap token count*/
    TOKEN = c2d(substr(x.i,inc+28,4)) /*Snap token*/
    SRCFUNC = c2d(substr(x.i,inc+32,2)) /*Source functional extent num*/
    SRCSUB = substr(x.i,inc+34,8) /*RVA subsystem name that*/
    /*owns the source extent, or*/
    /*blank if non-RVA*/
    SRCDEV = c2d(substr(x.i,inc+42,2)) /*Device address of source vol*/
    SRCVOL = substr(x.i,inc+44,6) /*Source volume id*/
    SRCBEO = c2d(substr(x.i,inc+50,4)) /*Begin of source extent addr*/
    SRECBOE = c2d(substr(x.i,inc+54,4)) /*End of source extent addr*/
    TRGFUNC = c2d(substr(x.i,inc+58,2)) /*Target functional extent num*/
    TRGSUB = substr(x.i,inc+60,8) /*RVA subsystem name that*/
    /*owns the target extent, or*/
    /*blank if non-RVA*/
    TRGDEV = c2d(substr(x.i,inc+68,2)) /*Device address of target vol*/
    TRGVOL = substr(x.i,inc+70,6) /*Target volume id*/
    TRGBEO = c2d(substr(x.i,inc+76,4)) /*Begin of target extent addr*/
    TRGEOE = c2d(substr(x.i,inc+80,4)) /*End of target extent addr*/
    SEXTDSID = c2d(substr(x.i,inc+84,2)) /*Source dataset ID number*/
    /*Used to relate to a given*/
/*             dataset name   */
TEXTDSID  = C2D(substr(x,i,inc+86,2))  /* Target dataset ID number: */
/* Used to relate to a given */
/*             dataset name   */

/*             dataset name   */
SEXTCLID  = C2D(substr(x,i,inc+88,2))  /* Source cluster ID number: */
/* Used to relate to a given */
/*             dataset name   */

/*             dataset name   */
TEXTCLID  = C2D(substr(x,i,inc+90,2))  /* Target cluster ID number: */
/* Used to relate to a given */
/*             dataset name   */

if EXTFLAG = 0 then
  extnote = 'No options specified';
else
do;
extnote = '';
ex = left(' ',1)||left('Ext. Snap:',10)||right(IOTIME,4),
||left(' ',1),
||left('Source Vol:',12)||SRCVOL,
||left(' ',3),
||left('Target Vol:',12)||TRGVOL
 /*--------------------------------------------------------------------*/
/* Extent flag options byte                                         */
 /*--------------------------------------------------------------------*/
SELECT
  when EXTFLAG = '10000000'b then extnote = 'Snapshot';
  when EXTFLAG = '01000000'b then extnote = 'Data Mover';
  when EXTFLAG = '00000001'b then extnote = 'Extent not processed';
ootherwise extnote = 'No options specified';
END

select
  when SEXTDSID = 1 then dsorg= 'Non VSAM'
  when SEXTDSID = 2 then dsorg= 'VSAM - Data'
  when SEXTDSID = 3 then dsorg= 'VSAM - Indx'
end
if SEXTDSID = 1 then do
dsne.1=left('Extent #:',10)||right(EXTnum,2)||ext
  dsne.2=left(' ',28)||left('Source dev.:',12)||right(SRCDEV,8),
  ||left(' ',1)||left('Target dev.:',12)||right(TRGDEV,8)
  dsne.3=left(' ',28)||left('Extent beg.:',12)||right(SRCEO,8),
  ||left(' ',1)||left('Extent beg.:',12)||right(TRGEOE,8)
  dsne.4=left(' ',28)||left('Extent end.:',12)||right(SRCEO,8),
  ||left(' ',1)||left('Extent end.:',12)||right(TRGEOE,8)
  "EXECIO * DISKW SNVSAM (STEM dsne.)"
end
if SEXTDSID = 2 then do
dse.1=left('Extent #:',10)||right(EXTnum,2),
  ||left(' ',1)||left('(Data)',7)||ext
  "EXECIO * DISKW SVSAM (STEM dse.)"
end
if SEXTDSID = 3 then do
if SEGTYPE = 4 & (SEGOFFST <> 0) & (SEGLEN <> 0) Then
   do k = 0 to segnum - 1
      incr = (SEGOFFST + (j*SEGLEN)) - 3
      DDSRTYPE = c2d(substr(x.i,incr,1)) /* Type of DDSR extent seg. */
      DDSRTIME = smf(c2d(substr(x.i,incr+1,4))) /* Start time of I/O */
      DDSRDATE = substr(c2x(substr(x.i,incr+5,4)),3,5) /*Start date of I/O */
      DDSRFUNC = c2d(substr(x.i,incr+11,2)) /* Functional extent number*/
      DDSRSUB = substr(x.i,incr+13,8) /* RVA subsystem name */
      DDSRDEV = c2d(substr(x.i,incr+21,2)) /* Device address */
      DDSRVOL = substr(x.i,incr+23,6) /* Volume identifier */
      DDSRBOE = c2d(substr(x.i,incr+29,4)) /* Beginning extent in CCHH.*/
      DDSREOE = c2d(substr(x.i,incr+33,4)) /* Ending extent in CCHH.*/
      TDSRDSID = c2d(substr(x.i,incr+37,2)) /* Target dataset ID number:*/
                 /* used to relate to a given*/
                 /* dataset name */
      TDSRCLID = c2d(substr(x.i,incr+39,2)) /* Target cluster ID number:*/
                 /* used to relate to a given*/
                 /* dataset name */
   end
end
END
END
/* Close & free all allocated files */
say 'Snapshot Record Summary Report. . . : summ
say 'VSAM file Snapshot Report . . . . : vsam
say 'Non VSAM file Snapshot Report . . . : nvsam
say "EXECIO Ø DISKW SVSAM (FINIS"
"EXECIO Ø DISKW SNVSAM (FINIS"
"EXECIO Ø DISKW SSUMM (FINIS"
"FREE FILE(SSUMM SVSAM SNVSAM SMF)"
exit
/* Error exit routine */
ERROR: say 'The following command produced non-zero RC = ' RC
   say SOURCELIN(SIGL)
exit
**SMF: procedure**

/* REXX - convert a 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)

    time = hh||":"||mm||":"||ss||":"||fr
return time

---

**Displaying the virtual storage map from IPCS**

Every monitoring package for OS/390 or z/OS provides an option for displaying the virtual storage address range and the range of addresses in use for the different areas of storage (for example PLPA, read/write nucleus, extended CSA, etc). This information can be very useful when diagnosing system problems or determining the real size of certain storage areas.

Although monitoring packages can provide storage map information from your active system, what happens when you need to review a dump dataset for a problem that occurred prior to the last batch of applied system maintenance or when you need to examine a dump dataset that was sent to you from the Atlanta office when you work in Dallas? Presumably, the storage map on those systems differs from that of your current, running system.

**THE STOREMAP IPCS VERBEXIT ROUTINE**

The STOREMAP program provided with this article is an IPCS VERBEXIT routine that allows for a virtual storage map display.
for the current IPCS default source data. This provides the opportunity to quickly review the storage map from historical dump datasets or from dump datasets that may have been generated at different sites. In its simplest format, from IPCS option 6 you can specify:

**VERBX STOREMAP**

This invocation will display the traditional virtual storage map graphic similar to that shown below:

```
+--------------+ 7FFFFFFF
|              |
|    EPVT      |
|              |
+--------------+ Ø7C00000
|    ECSA      |
|--------------| Ø5E2E000
|    EFLPA     |
|--------------| Ø5E2B000
|    EPLPA     |
|--------------| Ø2690000
|    ESQA      |
|--------------| Ø1904000
|    ER/W NUC  |
|--------------| Ø16AB000
   R/O
16MB --|---        ---|--
|       NUC    |
|--------------| Ø0FDA000
|    R/W NUC   |
|--------------| Ø0FD9117
|    SQA       |
|--------------| Ø0FCB000
|    PLPA      |
|--------------| Ø0E58000
|    CSA       |
|--------------| Ø0C66000
|    PVT       |
+--------------+ Ø0900000
```

Optional blank-separated keywords can also be supplied to the STOREMAP routine if you are interested in determining the starting and ending addresses of only certain areas of storage.
Valid keywords include PVT, CSA, MLPA, FLPA, PLPA, SQA, RWNUC, RONUC, ERWNUC, ESQA, EPLPA, EFLPA, EMLPA, ECSA, and EPVT. For example:

VERBX STOREMAP 'CSA PLPA MLPA RWNUC EMLPA EPVT'

would yield output similar to that shown below:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Start Address</th>
<th>End Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPVT</td>
<td>$07000000</td>
<td>$7FFFFFFF</td>
</tr>
<tr>
<td>R/W NUC</td>
<td>$00FC0000</td>
<td>$00FD9117</td>
</tr>
<tr>
<td>PLPA</td>
<td>$00C66000</td>
<td>$00E57FFF</td>
</tr>
<tr>
<td>CSA</td>
<td>$00900000</td>
<td>$00C65FFF</td>
</tr>
</tbody>
</table>

No storage area available for EMLPA

No storage area available for MLPA

ACTIVATING THE STOREMAP VERBEXIT EXIT

In order to make the STOREMAP VERBEXIT routine available to your IPCS session, linkedit STOREMAP into a load library that resides somewhere in the search order for your active session – the linklist or STEPLIB are two options.

To successfully display all the information about the virtual storage map be sure that SQA is included in the dump data.

SUMMARY

Regardless of the source system, STOREMAP can provide an accurate view of the virtual storage map in use at the time the dump data was created. This can be important background information when assessing dump dataset information in any environment and especially as it relates to the current dump data you are reviewing in IPCS.

STOREMAP

STOREMAP CSECT
STOREMAP AMODE 31
STOREMAP RMODE ANY

*-------------------------------------------------------------*
* STOREMAP is designed to be used as an IPCS VERBX exit routine *
* that can be used to display the virtual storage map information *
* for the current default dump.                                 *
*                                                              *
* To be able to display the entire virtual storage map be sure  *
* the dump data contains SQA.                                   *
*                                                              *
* The simplest format for exit invocation is as follows:        *
*                                                              *
* VERBX STOREMAP                                                 *
*                                                              *
* This will list the virtual storage map existing in the current *
* default dump.                                                 *
*                                                              *
* Specific areas of the virtual storage map can be displayed by *
* providing a blank delimited list of keyword parameters         *
* representing only the areas of storage you are interested in. *
* For example:                                                   *
*                                                              *
* VERBX STOREMAP 'CSA ESQA EMLPA'                               *
*                                                              *
* would display the starting and ending addresses for only CSA,  *
* extended SQA, and extended MLPA. Valid keyword parameters     *
* include: PVT, CSA, MLPA, FLPA, PLPA, SQA, RWNUC, RONUC, ERWNUC, *
* ESQA, EPLPA, EFLPA, EMLPA, ECSA, EPVT.                        *
*                                                              *
* Regardless of the invocation technique (with or without keyword *
* parameters), an IPCS symbol will be created representing the    *
* starting virtual storage address for each area of virtual      *
* storage requested and detected. If no keyword parameters are   *
* specified, a symbol will be created for each area of virtual    *
* storage defined. If keywords are specified, a symbol will be   *
* created for each of the specified areas that are defined in the *
* virtual storage map. The symbol names used for each storage    *
* area match the keyword parameter names described above.        *
*                                                              *
* The following IPCS exit services are demonstrated in this      *
* program:                                                       *
* Storage Access (IPCS service code ADPLSACC)                    *
* Expanded Print (IPCS service code ADPLSPR2)                    *
* Equate Symbol (IPCS service code ADPLSEQS)                     *
*                                                              *
* Chapter 10 in the OS/390 MVS IPCS Customization manual discusses *
* the various IPCS exit services in detail. This exit example    *
offers usage demonstration for only a handful of the available services.

In order to use the STOREMAP VERBX exit ensure that it is linkededit somewhere into the load module search order for your active IPCS session. Linkedit JCL similar to the following can be used:

```plaintext
//IEWL EXEC PGM=HEWLH096,PARM='XREF,LIST,MAP,RENT'
//SYSPRINT DD SYSOUT=*  
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(2,1))
//OBJECT DD DSN=object.code.pds,DISP=SHR
//SYSLMOD DD DSN=load.library,DISP=SHR
//SYSLIN DD *

INCLUDE OBJECT(STOREMAP)
ENTRY STOREMAP
NAME STOREMAP(R)

Register Usage Conventions:

R0       : work register, but generally available for use by calls to system functions
R1       : contains the parameter address on entry; work register, but generally available for use by calls to system functions
R2 - R7  : work registers
R8       : ABDPL base register
R9       : function specific parameter list address
R10      : future base register expansion
R11      : second base register
R12      : first base register
R13      : savearea/temporary storage address
R14 - R15: work registers; return address and return code; but generally available for use by calls to system functions

STM R14,R12,12(R13)       Save incoming registers
LR R12,R15                Copy module address
LA R11,4095(,R12)         Set up second ...
LA R11,1(,R11)            base register
USING STOREMAP,R12,R11    Set module addressability
LR R2,R1                  Copy parameter address
LR R3,R13                 Copy savearea address
STORAGE OBTAIN,LENGTH=WORKLEN,LOC=ANY
LR R0,R1                  Copy working storage address
LR R14,R1                 Again
LR R13,R1                 Again
L R1,=A(WORKLEN)          Get length
XR R15,R15                Set fill byte
MVCL R0,R14               Clear the storage
```
USING WORKAREA, R13          Set addressability
ST   R3, SAVEAREA+4          Save incoming savearea address
LA   R9, WORKPACC           Get ADPLPACC address
USING ADPLPACC, R9          Set addressability
LR   R8, R2                 Get ABDPL address
USING ABDP, R8              Set addressability
MVC  ASID(2), ADPLASID      Save the ASID
MVC  CVTADDR(4), ADPLCVT    Save the CVT address

*---------------------------------------------------------------------*
*   The ADPLEXT contains the address of the extension pointer. If     *
*   you want to process any input parameters passed to the VERBX      *
*   program they can be captured at this point and processed.         *
*                                                                     *
*   +Ø from the ADPLEXT address contains the parameter address.       *
*   +4 from the ADPLEXT address contains the CPPL address.            *
*                                                                     *
*   See comments earlier for the format of valid parameters.          *
*---------------------------------------------------------------------*
L    R7, ADPLEXT            Get extension address
LTR   R7, R7                 An extension?
BZ    NOPARM                No - unusual, but nothing to do
USING ADPLEXTN, R7          Set addressability
L    R15, ADPLOPTR          Get parm buffer address
LTR   R15, R15               A parameter?
BZ    NOPARM                No - nothing to do
ST    R15, PARMADDR         Save parm address
LR    R5, R15                Copy parm buffer address
S     R15, =F' 4'           Point to length
XR    R14, R14               Clear R14
ICM   R14, B'ØØ11', Ø(R15)  Save the length
LR    R6, R14                Copy to R6
S     R6, =F' 4'             Reduce by length word length

PARMLP  DS    ØH
C     R6, =F' 3'           Enough data for a keyword?
BNL    CHKKYWDS             Yes - go through keyword check
BCTR  R6, Ø                  Reduce by one for EX
EX     R6, BLNKCLC         Blanks?
BNE    BADPARN              No - that's an error
B NOPARM                Done the parm check

CHKKYWDS  DS    ØH
C     R6, =F' 6'           Enough for an ERWNUC check?
BL     KYWDLN5              No - check 5 char keywords
CLC    Ø(6, R5), =C' ERWNUC' ERWNUC keyword?
BNE    KYWDLN6              No - check next 6 char keyword
OI    PARMFLG1, PRMERWN    Set the ERWNUC keyword flag
B    KYWDLN6E              Prepare for next parm

KYWDLN6  DS    ØH
B    KYWDLN5              Check 5 char keywords
KYWDLN6E  DS    ØH
LA    R5, 6(, R5)        Point past keyword
S     R6,=F'6'              Reduce length
B     NEXTPARM              Prepare for next parm

KYWDLN5  DS  ØH
C     R6,=F'5'              Enough for 5 char keyword check?
BL    KYWDLN4               No - check 4 char keywords
CLC   Ø(5,R5),=C' EMLPA'
BNE   KYWD2LN5               No - check next 5 char keyword
OI    PARMFLG1,PRMEMLPA     Set the EMLPA keyword flag
B     KYWDLN5E              Prepare for next parm

KYWD2LN5  DS  ØH
CLC   Ø(5,R5),=C' EFLPA'
BNE   KYWD3LN5               No - check next 5 char keyword
OI    PARMFLG1,PRMEFLPA     Set the EFLPA keyword flag
B     KYWDLN5E              Prepare for next parm

KYWD3LN5  DS  ØH
CLC   Ø(5,R5),=C' EPLPA'
BNE   KYWD4LN5               No - check next 5 char keyword
OI    PARMFLG1,PRMEPLPA     Set the EPLPA keyword flag
B     KYWDLN5E              Prepare for next parm

KYWD4LN5  DS  ØH
CLC   Ø(5,R5),=C' RONUC'
BNE   KYWD5LN5               No - check next 5 char keyword
OI    PARMFLG2,PRMRON       Set the RONUC keyword flag
B     KYWDLN5E              Prepare for next parm

KYWD5LN5  DS  ØH
CLC   Ø(5,R5),=C' RWNUC'
BNE   KYWD6LN5               No - check next 5 char keyword
OI    PARMFLG2,PRMRWN       Set the RWNUC keyword flag
B     KYWDLN5E              Prepare for next parm

KYWD6LN5  DS  ØH
B     KYWDLN4               Check 4 char keywords
LA    R5,5,(R5)             Point past keyword
S     R6,=F'5'              Reduce length
B     NEXTPARM              Prepare for next parm

KYWDLN4  DS  ØH
C     R6,=F'4'              Enough for 4 char keyword check?
BL    KYWDLN3               No - check 3 char keywords
CLC   Ø(4,R5),=C' EPVT'
BNE   KYWD2LN4               No - check next 4 char keyword
OI    PARMFLG1,PRMEPVT     Set the EPVT keyword flag
B     KYWDLN4E              Prepare for next parm

KYWD2LN4  DS  ØH
CLC   Ø(4,R5),=C' ECSA'
BNE   KYWD3LN4               No - check next 4 char keyword
OI    PARMFLG1,PRMECSA     Set the ECSA keyword flag
B     KYWDLN4E              Prepare for next parm

KYWD3LN4  DS  ØH
CLC   Ø(4,R5),=C' ESQA'
BNE   KYWD4LN4               No - check next 4 char keyword

OI PARMFLG1, PRMESQA  Set the ESQA keyword flag
B KYWDLN4E Prepare for next parm

KYWD4LN4 DS ØH
CLC Ø(4, R5), =C' PLPA' PLPA keyword?
BNE KYWD5LN4 No - check next 4 char keyword
OI PARMFLG2, PRMPLPA Set the PLPA keyword flag
B KYWDLN4E Prepare for next parm

KYWD5LN4 DS ØH
CLC Ø(4, R5), =C' FLPA' FLPA keyword?
BNE KYWD6LN4 No - check next 4 char keyword
OI PARMFLG2, PRMFLPA Set the FLPA keyword flag
B KYWDLN4E Prepare for next parm

KYWD6LN4 DS ØH
CLC Ø(4, R5), =C' MLPA' MLPA keyword?
BNE KYWD7LN4 No - check next 4 char keyword
OI PARMFLG2, PRMMLPA Set the MLPA keyword flag
B KYWDLN4E Prepare for next parm

KYWD7LN4 DS ØH
B KYWDLN3 Check 3 char keywords

KYWDLN3E DS ØH
LA R5, 4(, R5) Point past keyword
S R6, =F' 4' Reduce length
B NEXTPARM Prepare for next parm

KYWDLN3 DS ØH
CLC Ø(3, R5), =C' SQA' SQA keyword?
BNE KYWD2LN3 No - check next 3 char keyword
OI PARMFLG2, PRMSQA Set the SQA keyword flag
B KYWDLN3E Prepare for next parm

KYWD2LN3 DS ØH
CLC Ø(3, R5), =C' CSA' CSA keyword?
BNE KYWD3LN3 No - check next 3 char keyword
OI PARMFLG2, PRMCSA Set the CSA keyword flag
B KYWDLN3E Prepare for next parm

KYWD3LN3 DS ØH
CLC Ø(3, R5), =C' PVT' PVT keyword?
BNE KYWD4LN3 No - check next 3 char keyword
OI PARMFLG2, PRMPVT Set the PVT keyword flag
B KYWDLN3E Prepare for next parm

KYWD4LN3 DS ØH
B NEXTPARM Prepare for next parm

KYWDLN3E DS ØH
LA R5, 3(, R5) Point past keyword
S R6, =F' 3' Reduce length
B NEXTPARM Prepare for next parm

NEXTPARM DS ØH
LTR R6, R6 End of parameter buffer?
BZ NOPARM Yes - that's fine
CLI Ø(R5), C' A blank?
BNE BADPARM1 No - indicate invalid parm
LA R5, 1(, R5) Point to next data byte

BCTR R6, Ø
B PARMLP

*----------------------------------------------------------*

BADPARM DS ØH
LA R0, PRMMGS1L
LA R1, PARMMGS1
BAL R14, PRINTLN
LA R0, 1
LA R1, =C' ' 
BAL R14, PRINTLN

*----------------------------------------------------------*

NOPARM DS ØH

* Obtain the CVT.                                        *

MVC ADPLPAAD(4), CVTADDR
MVC ADPLDLEN(2), =AL2(CVTOSLVF+1-CVT)
OI ADPLPRDP, ADPLVRT+ADPLSAMK
L R15, ADPLSERV

CALL (15),
((R8),
CODEACC,
(R9)), MF=(E, CALLLST)

MVC CBNAME(4), =C' CVT '
LTR R15, R15
BNZ NOSTORE

* Obtain the Storage Map Extension.                       *

L R1, ADPLPART
USING CVT, R1
MVC CBNAME(4), =C' CVT '
MVC CBADDR(4), ADPLPAAD
CLC CBNAME(3), CVTCVT+1
BNE CBERROR

MVC GDAADDR(4), CVTGDA
MVC ADPLPAAD(4), CVTSMEXT
MVC ADPLDLEN(2), =AL2(CVTEMLPE-CVTVSTGX+4)
NI ADPLPRDP, 255-ADPLSAMK

DROP R1
L R15, ADPLSERV

CALL (15),
((R8),
CODEACC,
(R9)), MF=(E, CALLLST)

MVC CBNAME(4), =C' SMEx'
LTR R15, R15
BNZ NOSTORE

* Extract storage map information

L R1, ADPLPART Get buffer location address
USING CVTVSTGX, R1
MVC MLPAS(72), CVTMLPAS Copy the pertinent information
DROP R1

* Obtain the GDA.

MVC ADPLPAAD(4), GDAADDR Set address to the GDA
MVC ADPLDLLEN(2), =AL2(GDAEND-GDA) Set get length
L R15, ADPLSERV Get service routine address
CALL (15), X ((R8), X CODEACC, X (R9)), MF=(E, CALLLST)
MVC CBNAME(4), =C'GDA ' Indicate control block acronym
LTR R15, R15 Were things ok?
BNZ NOSTORE No - issue storage not found msg
USING GDA, R1
L R1, ADPLPART Get buffer location address
MVC CBNAME(4), =C'GDA ' Copy control block name
MVC CBADDR(4), ADPLPAAD Copy control block address
CLC CBNAME(4), GDAID Correct control block?
BNE CBERROR No - no sense going on

MVC CSA(4), GDACSA Copy CSA starting address
MVC CSASZ(4), GDACSASZ Copy CSA size
MVC ECSA(4), GDAECSA Copy ECSA starting address
MVC ECSAS(4), GDAECSAS Copy ECSA size
MVC SQA(4), GDAQSQA Copy SQA starting address
MVC SQASZ(4), GDAQSASZ Copy SQA size
MVC ESQA(4), GDAESQA Copy ESQA starting address
MVC ESQAS(4), GDAESQAS Copy ESQA size
MVC PVT(4), GDAPVT Copy Private starting address
MVC PVTSZ(4), GDAPVTSZ Copy Private size
MVC EPVT(4), GDAEPVT Copy EPrivate starting address
MVC EPVTS(4), GDAEPVTS Copy EPrivate size
DROP R1

L R14, CSA Get CSA start addr
L R15, CSASZ Get CSA length
AR R14, R15 Calculate ...
BCTR R14, Ø ending addr
ST R14, CSAE Save CSA ending addr
L R14, SQA Get SQA start addr
L R15, SQASZ Get SQA length
AR R14, R15 Calculate ...
BCTR R14, Ø ending addr
ST R14, SQAE Save SQA ending addr
L R14, PVT Get PVT start addr
L    R15, PVTSZ             Get PVT length
AR   R14, R15             Calculate ...
BCTR R14, Ø               ending addr
ST   R14, PVTE             Save PVT ending addr
L    R14, ECSA             Get CSA start addr
L    R15, ECSAS            Get CSA length
AR   R14, R15             Calculate ...
BCTR R14, Ø               ending addr
ST   R14, ECSAE            Save CSA ending addr
L    R14, ESQA             Get SQA start addr
L    R15, ESQAS            Get SQA length
AR   R14, R15             Calculate ...
BCTR R14, Ø               ending addr
ST   R14, ESQAE            Save SQA ending addr
L    R14, EPVT             Get PVT start addr
L    R15, EPVTS            Get PVT length
AR   R14, R15             Calculate ...
BCTR R14, Ø               ending addr
ST   R14, EPVTE            Save PVT ending addr

*--------------------------------------------------------------------*
CLC   PARMFLGS(2), =2X'ØØ'   Display by keyword selection?
BNE   KYWDDSPL              Yes - do keyword display

*--------------------------------------------------------------------*
LA    RØ, 1                  Set message length
LA    R1, =C' '              Get message address
BAL   R14, PRINTLNL          Go print the line

*----------------------------------------------------------------------*
EPVTCHECK DS    ØH
MVC   LINEBUF(L' TOPBOTM), TOPBOTM Get line model
L    R15, EPVTE             Get EPVTE ending address
BAL   R14, HEXCNVT           Make it readable
MVC   LINEBUF+LNOFFST3(8), DBL1 Copy into output buffer
LA    RØ, L' TOPBOTM        Set message length
LA    R1, LINEBUF            Get message address
BAL   R14, PRINTLNL          Go print the line
LA    RØ, L' FILL            Set message length
LA    R1, FILL               Get message address
BAL   R14, PRINTLNL          Go print the line
MVC   LINEBUF(L' FILL), FILL Get line model
MVC   LINEBUF+LNOFFST2(4), =C'EPVT' Set area identifier
LA    RØ, L' FILL            Set message length
LA    R1, LINEBUF            Get message address
BAL   R14, PRINTLNL          Go print the line
LA    RØ, L' FILL            Set message length
LA    R1, FILL               Get message address
BAL   R14, PRINTLNL          Go print the line
MVC   LINEBUF(L' MIDDLE), MIDDL Get line model
L    R15, EPVT              Get EPVT starting address
BAL   R14, HEXCNVT           Make it readable
MVC   LINEBUF+LNOFFST3(8), DBL1 Copy into output buffer
LA R0, L'MIDDLE
LA R1, LINEBUF
BAL R14, PRINTLN
MVC PREVAREA(4), EPVT
MVC AREASTRT(4), EPVT
MVC AREAEND(4), EPVTE
MVC AREAID(8), =CL8'EPVT'
BAL R14, SYMDEF

ECSACHK DS ØH
CLC ECSA(4), =F'Ø'
BE EMLPACHK
MVC AREASTRT(4), ECSA
MVC AREAEND(4), ECSAE
MVC AREAID(8), =CL8'ECSA'
BAL R14, AREACHK

EMLPACHK DS ØH
CLC EMLPAS(4), =F'Ø'
BE EFLPACHK
MVC AREASTRT(4), EMLPAS
MVC AREAEND(4), EMLPAE
MVC AREAID(8), =CL8'EMLPA'
BAL R14, AREACHK

EFLPACHK DS ØH
CLC EFLPAS(4), =F'Ø'
BE ESQACHK
MVC AREASTRT(4), EFLPAS
MVC AREAEND(4), EFLPAE
MVC AREAID(8), =CL8'EFLPA'
BAL R14, AREACHK

ESQACHK DS ØH
CLC ESQA(4), =F'Ø'
BE ERWNCHK
MVC AREASTRT(4), ESQA
MVC AREAEND(4), ESQAE
MVC AREAID(8), =CL8'ESQA'
BAL R14, AREACHK

ERWNCHK DS ØH
CLC    ERWNS(4), = F' Ø' Any data?
BE    ERONCHK No - go check next area
MVC    AREAESTRT(4), ERWNS Copy start address
MVC    AREAEEND(4), ERWNE Copy end address
MVC    AREAILD(8), = CL8' ERW NUC' Copy identifier
BAL    R14, AREACHK Go process

*--------------------------------------------------------------------*
ERONCHK DS ØH
CLC    RONS(4), = F' Ø' Any data?
BE    RWNCCHK No - go check next area
L    R15, PREVAREA Get start addr of previous area
BCTR    R15, Ø Reduce by one
C    R15, RONS End is previous start?
BE    NORONE Yes - no extra info required
MVC    LINEBUF(L' MIDDLE), MIDDLE Get line model
L    R15, RONE Get RON ending addr
BAL    R14, HEXCNVT Make it readable
MVC    LINEBUF+LNOFFST3(8), DBL1 Copy into output buffer
LA    RØ, L' MIDDLE Set message length
LA    R1, LINEBUF Get message address
BAL    R14, PRINTLN Go print the line
NORONE DS ØH
MVC    LINEBUF(L' FILL), FILL Get line model
MVC    LINEBUF+LNOFFST2(3), = C' R/O' Set area identifier
LA    RØ, L' FILL Set message length
LA    R1, LINEBUF Get message address
BAL    R14, PRINTLN Go print the line
MVC    LINEBUF(L' MB16), MB16 Get line model
LA    RØ, L' MB16 Set message length
LA    R1, MB16 Get message address
BAL    R14, PRINTLN Go print the line
MVC    LINEBUF(L' MIDDLE), MIDDLE Get line model
L    R15, RONS Get RON starting address
BAL    R14, HEXCNVT Make it readable
MVC    LINEBUF+LNOFFST3(8), DBL1 Copy into output buffer
LA    RØ, L' MIDDLE Set message length
LA    R1, LINEBUF Get message address
BAL    R14, PRINTLN Go print the line
MVC    PREVAREA(4), RONS Save RON as previous area addr
MVC    AREAESTRT(4), RONS Copy start address
MVC    AREAEEND(4), RONE Copy end address
MVC    AREAILD(8), = CL8' R/O NUC' Copy identifier
BAL    R14, SYMDEF Go define a symbol

*--------------------------------------------------------------------*
RWNCHK DS ØH
CLC    RWNS(4), = F' Ø' Any data?
BE   SQACHK    No - go check next area
MVC   AREASTRT(4), RWNS    Copy start address
MVC   AREAEND(4), RWNE     Copy end address
MVC   AREAID(8), =CL8'R/W NUC'    Copy identifier
BAL   R14, AREACHK    Go process

*--------------------------------------------------------------------*
SQACHK    DS    ØH
CLC   SQA(4), =F'Ø'    Any data?
BE   PLPACHK    No - go check next area
MVC   AREASTRT(4), SQA    Copy start address
MVC   AREAEND(4), SQAE    Copy end address
MVC   AREAID(8), =CL8'SQA'    Copy identifier
BAL   R14, AREACHK    Go process

*--------------------------------------------------------------------*
PLPACHK    DS    ØH
CLC   PLPAS(4), =F'Ø'    Any data?
BE   FLPACHK    No - go check next area
MVC   AREASTRT(4), PLPAS    Copy start address
MVC   AREAEND(4), PLPAE    Copy end address
MVC   AREAID(8), =CL8'PLPA'    Copy identifier
BAL   R14, AREACHK    Go process

*--------------------------------------------------------------------*
FLPACHK    DS    ØH
CLC   FLPAS(4), =F'Ø'    Any data?
BE   MLPACHK    No - go check next area
MVC   AREASTRT(4), FLPAS    Copy start address
MVC   AREAEND(4), FLPAE    Copy end address
MVC   AREAID(8), =CL8'FLPA'    Copy identifier
BAL   R14, AREACHK    Go process

*--------------------------------------------------------------------*
MLPACHK    DS    ØH
CLC   MLPAS(4), =F'Ø'    Any data?
BE   CSACHK    No - go check next area
MVC   AREASTRT(4), MLPAS    Copy start address
MVC   AREAEND(4), MLPAE    Copy end address
MVC   AREAID(8), =CL8'MLPA'    Copy identifier
BAL   R14, AREACHK    Go process

*--------------------------------------------------------------------*
CSACHK    DS    ØH
CLC   CSA(4), =F'Ø'    Any data?
BE   PVTCHK    No - go check next area
MVC   AREASTRT(4), CSA    Copy start address
MVC   AREAEND(4), CSAE    Copy end address
MVC   AREAID(8), =CL8'CSA'    Copy identifier
BAL   R14, AREACHK    Go process

*--------------------------------------------------------------------*
PVTCHK    DS    ØH
L   R15, PREVAREA    Get start addr of previous area
BCTR  R15, Ø    Reduce by one
C   R15, PVTE    End is previous start?
BE  NOPVTE  Yes - no extra info required
MVC  LINEBUF(L'MIDDLE),MIDDLE Get line model
L  R15,PVTE  Get PVT ending addr
BAL  R14,HEXCNVT Make it readable
MVC  LINEBUF+LNOFFST3(8),DBL1 Copy into output buffer
LA  R0,L'MIDDLE Set message length
LA  R1,LINEBUF Get message address
BAL  R14,PRINTLN Go print the line

NOPVTE  DS  ØH
LA  R0,L'FILL Set message length
LA  R1,FILL Get message address
BAL  R14,PRINTLN Go print the line
MVC  LINEBUF(L'FILL),FILL Get line model
MVC  LINEBUF+LNOFFST2(3),=C'PVT' Set area identifier
LA  R0,L'FILL Set message length
LA  R1,LINEBUF Get message address
BAL  R14,PRINTLN Go print the line
LA  R0,L'FILL Set message length
LA  R1,LINEBUF Get message address
BAL  R14,PRINTLN Go print the line
MVC  LINEBUF(L'MIDDLE),MIDDLE Get line model
CLC  PVT(4),=F'Ø' At the start?
BNE  NOSTART No - don't change line format
MVC  LINEBUF(L'TOPBOTM),TOPBOTM Get line model

NOSTART  DS  ØH
L  R15,PVT Get PVT starting address
BAL  R14,HEXCNVT Make it readable
MVC  LINEBUF+LNOFFST3(8),DBL1 Copy into output buffer
LA  R0,L'MIDDLE Set message length
LA  R1,LINEBUF Get message address
BAL  R14,PRINTLN Go print the line
MVC  PREVAREA(4),PVT Save PVT as previous area addr
MVC  AREAEND(4),PVTE Copy end address
MVC  AREAID(8),=CL8'PVT' Copy identifier
BAL  R14,SYMDEF Go define a symbol

*--------------------------------------------------------------------*
DONECHK  DS  ØH
CLC  PREVAREA(4),=F'Ø' All done?
BE  LASTLINE Yes - pack it in
MVC  LINEBUF(L'TOPBOTM),TOPBOTM Get line model
XR  R15,R15 Set to Ø
BAL  R14,HEXCNVT Make it readable
MVC  LINEBUF+LNOFFST3(8),DBL1 Copy into output buffer
LA  R0,L'TOPBOTM Set message length
LA  R1,LINEBUF Get message address
BAL  R14,PRINTLN Go print the line

*--------------------------------------------------------------------*
LASTLINE  DS  ØH
LA  R0,1 Set message length
LA R1, =C' ' Get message address
BAL R14, PRINTLN Go print the line
B RETURN Pack it in for now

*--------------------------------------------------------------------*

KYWDDSPL DS ØH
TM PARMFLG1, PRMEPVT Display extended private?
BNO CHKKEYWD2 No - check next keyword
MVC AREASTRT(4), EPVT Copy start address
MVC AREAEND(4), EPVTE Copy end address
MVC AREAID D(8), =CL8'EPVT' Copy identifier
BAL R14, AREADSPL Output area information

CHKKEYWD2 DS ØH
TM PARMFLG1, PRMECSA Display extended CSA?
BNO CHKKEYWD3 No - check next keyword
MVC AREASTRT(4), ECSA Copy start address
MVC AREAEND(4), ECSAE Copy end address
MVC AREAID D(8), =CL8'ECSA' Copy identifier
BAL R14, AREADSPL Output area information

CHKKEYWD3 DS ØH
TM PARMFLG1, PRMEMLPA Display extended MLPA?
BNO CHKKEYWD4 No - check next keyword
MVC AREASTRT(4), EMLPA Copy start address
MVC AREAEND(4), EMLPAE Copy end address
MVC AREAID D(8), =CL8'EMLPA' Copy identifier
BAL R14, AREADSPL Output area information

CHKKEYWD4 DS ØH
TM PARMFLG1, PRMEFLPA Display extended FLPA?
BNO CHKKEYWD5 No - check next keyword
MVC AREASTRT(4), EFLPA Copy start address
MVC AREAEND(4), EFLPAE Copy end address
MVC AREAID D(8), =CL8'EFLPA' Copy identifier
BAL R14, AREADSPL Output area information

CHKKEYWD5 DS ØH
TM PARMFLG1, PRMEPLPA Display extended PLPA?
BNO CHKKEYWD6 No - check next keyword
MVC AREASTRT(4), EPLPA Copy start address
MVC AREAEND(4), EPLPAE Copy end address
MVC AREAID D(8), =CL8'EPLPA' Copy identifier
BAL R14, AREADSPL Output area information

CHKKEYWD6 DS ØH
TM PARMFLG1, PRMESQA Display extended SQA?
BNO CHKKEYWD7 No - check next keyword
MVC AREASTRT(4), ESQA Copy start address
MVC AREAEND(4), ESQAE Copy end address
MVC AREAID D(8), =CL8'ESQA' Copy identifier
BAL R14, AREADSPL Output area information

CHKKEYWD7 DS ØH
TM PARMFLG1, PRMERWN Display extended R/W NUC?
BNO CHKKEYWD8 No - check next keyword
MVC AREASTRT(4), ERWNS Copy start address
MVC AREA END(4), ERWNE  Copy end address
MVC AREAID(8), =CL8'ER/W NUC' Copy identifier
BAL R14, AREADSPL  Output area information

CHKKYWD8 DS ØH
TM PARM FLG2, PRMRON  Display R/O NUC?
BNO CHKKYWD9  No - check next keyword
MVC AREASTRT(4), RONS  Copy start address
MVC AREAEND(4), RONE  Copy end address
MVC AREAID(8), =CL8'R/O NUC' Copy identifier
BAL R14, AREADSPL  Output area information

CHKKYWD9 DS ØH
TM PARM FLG2, PRMRWN  Display R/W NUC?
BNO CHKKYWDA  No - check next keyword
MVC AREASTRT(4), RWNS  Copy start address
MVC AREAEND(4), RWNE  Copy end address
MVC AREAID(8), =CL8'R/W NUC' Copy identifier
BAL R14, AREADSPL  Output area information

CHKKYWDA DS ØH
TM PARM FLG2, PRMSQA  Display SQA?
BNO CHKKYWDB  No - check next keyword
MVC AREASTRT(4), SQA  Copy start address
MVC AREAEND(4), SQAE  Copy end address
MVC AREAID(8), =CL8'SQA' Copy identifier
BAL R14, AREADSPL  Output area information

CHKKYWDB DS ØH
TM PARM FLG2, PRMPLPA  Display PLPA?
BNO CHKKYWDC  No - check next keyword
MVC AREASTRT(4), PLPAS  Copy start address
MVC AREAEND(4), PLPAE  Copy end address
MVC AREAID(8), =CL8'PLPA' Copy identifier
BAL R14, AREADSPL  Output area information

CHKKYWDC DS ØH
TM PARM FLG2, PRMFLPA  Display FLPA?
BNO CHKKYWDD  No - check next keyword
MVC AREASTRT(4), FLPAS  Copy start address
MVC AREAEND(4), FLPAE  Copy end address
MVC AREAID(8), =CL8'FLPA' Copy identifier
BAL R14, AREADSPL  Output area information

CHKKYWDD DS ØH
TM PARM FLG2, PRMMLPA  Display MLPA?
BNO CHKKYWDE  No - check next keyword
MVC AREASTRT(4), MLPAS  Copy start address
MVC AREAEND(4), MLPAE  Copy end address
MVC AREAID(8), =CL8'MLPA' Copy identifier
BAL R14, AREADSPL  Output area information

CHKKYWDE DS ØH
TM PARM FLG2, PRMCSA  Display CSA?
BNO CHKKYWDF  No - check next keyword
MVC AREASTRT(4), CSA  Copy start address
MVC AREAEND(4), CSAE  Copy end address
MVC AREAID(8), =CL8'CSA'  Copy identifier
BAL R14, AREADSPL  Output area information

CHKKYWDG DS ØH
TM PARMFLG2, PRMPVT  Display private?
BNO CHKKYWDG  No - check next keyword
MVC AREAEXT(4), PVT  Copy start address
MVC AREAEAST(4), PVT  Copy end address
MVC AREAID(8), =CL8'PVT'  Copy identifier
BAL R14, AREADSPL  Output area information

CHKKYWDG DS ØH

RETURN DS ØH
L R3, SAVEAREA+4  Load incoming savearea address
LR R1, R13  Get working storage address
STORAGE RELEASE, LENGTH=WORKLEN, ADDR=(R1)
LR R13, R3  Restore incoming savearea address
LM R14, R12, 12(R13)  Restore incoming registers
XR R15, R15  Set return code
BR R14  Return

*------------------------------------------------------------------------*

RETURN DS ØH

L R3, SAVEAREA+4  Load incoming savearea address
LR R1, R13  Get working storage address
STORAGE RELEASE, LENGTH=WORKLEN, ADDR=(R1)
LR R13, R3  Restore incoming savearea address
LM R14, R12, 12(R13)  Restore incoming registers
XR R15, R15  Set return code
BR R14  Return

*------------------------------------------------------------------------*

RETURN DS ØH

MVI LINEBUF, C' '  Set fill byte
MVC LINEBUF+1(131), LINEBUF  Clear the area
MVC LINEBUF(STMSG), STORMSG  Copy the message
BAL R14, HEXCNVT  Make the rc readable
MVC LINEBUF+51(2), DBL1+6  Copy rc into message
ICM R15, B'1111', ADPLPAAD  Get control block address
BAL R14, HEXCNVT  Make it readable
MVC LINEBUF+37(8), DBL1  Copy c/b address into message
MVC LINEBUF+29(4), CBNAME  Copy c/b name into message
LA R0, STMSGL  Get message length
LA R1, LINEBUF  Get message address
BAL R14, PRINTLN  Go print the line

TERM DS ØH

LA R0, 1  Set message length
LA R1, =C' '  Get message address
BAL R14, PRINTLN  Go print a blank line
LA R0, TRMMSG1L  Get message length
LA R1, TERMMSG1  Get message address
BAL R14, PRINTLN  Go print the line
B RETURN  We're done

*------------------------------------------------------------------------*

CBERROR DS ØH

MVI LINEBUF, C' '  Set fill byte
MVC LINEBUF+1(131), LINEBUF  Clear the area
MVC LINEBUF(CBEMSG1L), CBEMSG1  Copy the message
L R15, CBADDR  Get control block address
BAL R14, HEXCNVT  Make the rc readable
MVC LINEBUF+51(8), DBL1 Copy c/b address into message
MVC LINEBUF+2Ø(4), CBNAME Copy c/b name into message
LA RØ, CBEMSG1L Get message length
LA R1, LINEBUF Get message address
BAL R14, PRINTLN Go print the line
B TERM All done

* Subroutines
*---------------------------------------------------------------------*
PRINTLN DS ØH
* The PRINTLN subroutine generates a line of output using the *
* IPCS print service.
* On entry:  RØ - contains the length of the output line
* R1 - contains the address of the output line
* R8 - contains the address of the ABDPL
* On exit:  R15 - contains the return code from the IPCS print *
* service
*---------------------------------------------------------------------*
STM RØ, R15, REGSAVE Save the registers
LA R7, WORKPPR2 Get BLSUPPR2 address
MVC Ø(PPR2999-PPR2000(R7), PPR2 Copy the PPR2 model
MVC PPR2BUF-PPR2(4, R7), ADPLBUF Copy print buffer address
ST RØ, PPR2BUFL-PPR2(, R7) Save the message length
L R3, PPR2BUFL-PPR2(, R7) Copy the message length
L R15, ADPLBUF Get message buffer address
MVI Ø(R15), C' ' Set fill byte
MVC 1(131, R15), Ø(R15) Clear message buffer area
L R15, ADPLBUF Get message buffer address
BCTR R3, Ø Reduce length by one for EX
EX R3, MSGMVC Copy the message
MVI PPR2PFL1-PPR2(R7), PPR2MSG Indicate buffer contains a msg
L R15, ADPLSERV Get service routine address
CALL (15), (((R8), X
CODEPR2, X
(R7)), MF=(E, CALLLST)
PRINTLN DS ØH
LM RØ, R14, REGSAVE Restore required registers
BR R14 Return

*---------------------------------------------------------------------*
AREACHK DS ØH
* The AREACHK subroutine generates appropriate output lines for *
* the identified area of virtual storage.
* On entry:  AREASTRT - contains the starting address of the *
* area being processed
*---------------------------------------------------------------------*
STM RØ, R15, REGSAVE Save the registers
LA R7, WORKPPR2 Get BLSUPPR2 address
MVC Ø(PPR2999-PPR2000, R7), PPR2 Copy the PPR2 model
MVC PPR2BUF-PPR2(4, R7), ADPLBUF Copy print buffer address
ST RØ, PPR2BUFL-PPR2(, R7) Save the message length
L R3, PPR2BUFL-PPR2(, R7) Copy the message length
L R15, ADPLBUF Get message buffer address
MVI Ø(R15), C' ' Set fill byte
MVC 1(131, R15), Ø(R15) Clear message buffer area
L R15, ADPLBUF Get message buffer address
BCTR R3, Ø Reduce length by one for EX
EX R3, MSGMVC Copy the message
MVI PPR2PFL1-PPR2(R7), PPR2MSG Indicate buffer contains a msg
L R15, ADPLSERV Get service routine address
CALL (15), (((R8), X
CODEPR2, X
(R7)), MF=(E, CALLLST)
PRINTLN DS ØH
LM RØ, R14, REGSAVE Restore required registers
BR R14 Return

*---------------------------------------------------------------------*
* AREAEND - contains the ending address of the area being processed *
* AREAID   - contains the identifier of the area being processed *

STM R0, R15, REGSAVE2  ; Save registers
L R15, PREVAREA      ; Get start addr of previous area
BCTR R15, Ø         ; Reduce by one
C R15, AREAEND      ; End is previous start?
BE NOAREAE          ; Yes - no extra info required
MVC LINEBUF(L' MIDDLE), MIDDLE ; Get line model
L R15, AREAEND      ; Get area ending addr
BAL R14, HEXCNVT    ; Make it readable
MVC LINEBUF+LNOFFST3(8), DBL1 ; Copy into output buffer
LA R0, L' MIDDLE    ; Set message length
LA R1, LINEBUF     ; Get message address
BAL R14, PRINTLN    ; Go print the line

NOAREAE DS ØH
MVC LINEBUF(L' FILL), FILL ; Get line model
MVC LINEBUF+LNOFFST2(8), AREAID ; Set area identifier
LA R0, L' FILL      ; Set message length
LA R1, LINEBUF     ; Get message address
BAL R14, PRINTLN    ; Go print the line
MVC LINEBUF(L' MIDDLE), MIDDLE ; Get line model
L R15, AREASTRT    ; Get area starting address
BAL R14, HEXCNVT    ; Make it readable
MVC LINEBUF+LNOFFST3(8), DBL1 ; Copy into output buffer
LA R0, L' MIDDLE    ; Set message length
LA R1, LINEBUF     ; Get message address
BAL R14, PRINTLN    ; Go print the line
MVC PREVAREA(4), AREASTRT ; Save start as previous area addr
BAL R14, SYMDEF     ; Go define a symbol
LM R0, R15, REGSAVE2 ; Restore registers
BR R14

*---------------------------------------------------------------------*
AREADSPL DS ØH
*---------------------------------------------------------------------*
* The AREADSPL subroutine generates appropriate output lines for the identified area of virtual storage.
* On entry: AREASTRT - contains the starting address of the area being processed
* AREAEND - contains the ending address of the area being processed
* AREAID   - contains the identifier of the area being processed

STM R0, R15, REGSAVE2  ; Save registers
CLC AREASTRT(4), =F' 0'  ; A starting address?
BE AREADS2Ø  ; No - make one more check
AREADS10 DS ØH
MVC LINEBUF(L'AREAMSG2),AREAMSG2 Get line model
MVC LINEBUF+3(8),AREAID Copy the identifier
L R15,AREASTRT Get area starting address
BAL R14,HEXCNVT Make it readable
MVC LINEBUF+L'AREAMSG2(8),DBL1 Copy into output buffer
LA R0,L'AREAMSG2+8 Set message length
LA R1,LINEBUF Get message address
BAL R14,PRINTLN Go print the line
MVC LINEBUF+L'AREAMSG3(8),DBL1 Copy into output buffer
L R0,L'AREAMSG3+8 Set message length
LA R1,LINEBUF Get message address
BAL R14,PRINTLN Go print the line
LA R0,1 Set message length
LA R1,=C' ' Get message address
BAL R14,PRINTLN Go print a blank line
BAL R14,SYMDEF Go define a symbol
B AREADSPE We 're done

AREADS20 DS ØH
CLC AREAID(8),=CL8'PVT' Area is PVT?
BE AREADS10 Yes - start address of Ø is ok
MVC LINEBUF(L'AREAMSG1),AREAMSG1 Get line model
MVC LINEBUF+L'AREAMSG1(8),AREAID Copy area identifier
LA R0,L'AREAMSG1+8 Set message length
LA R1,LINEBUF Get message address
BAL R14,PRINTLN Go print the line
LA R0,1 Set message length
LA R1,=C' ' Get message address
BAL R14,PRINTLN Go print a blank line

AREADSPE DS ØH
LM R0,R15,REGSAVE2 Restore registers
BR R14

*---------------------------------------------------------------------*
HEXCNVT DS ØH
*---------------------------------------------------------------------*
* The HEXCNVT subroutine converts the hex contents of R15 to *
* a human-readable format in variable DBL1. *
*---------------------------------------------------------------------*
ST R15,DBL2 Save the value
UNPK DBL1(9),DBL2(5) Unpack it
NC DBL1(8),=8X'OF' Turn off high nibble
TR DBL1(8),=C'0123456789ABCDEF' Make it readable
BR R14 Return

*---------------------------------------------------------------------*
SYMDEF DS ØH
*---------------------------------------------------------------------*
* Create an IPCS symbol for the specified symbol name. On entry *

to this routine:

- AREAID - contains the name of the symbol to be defined
- AREASTRT - contains the starting address of area for symbol definition
- AREAEND - contains the ending address of area for symbol definition

---------------------------------------------------------------------

STM R0, R15, REGSAVE3       Save the registers
NI FLAG1, 255-SYMTRY       Reset the SYMTRY flag
MVC SYMNAME(32), =40C' '   Clear symbol name area
MVC SYMREMRK(40), =40C' '   Clear remark area
MVC SYMLEN(4), =F'8'       Set default length
MVC SYMNAME(8), AREAID     Copy symbol name
CLC SYMNAME(8), =C'R/W NUC' R/W NUC?
BNE CHKSYM2               No - check next special case
MVC SYMNAME(8), =C'RWNUC'   ' Move in altered symbol name
B SYMØ                   Go on

CHKSYM2 DS ØH
CLC SYMNAME(8), =C'R/O NUC' R/O NUC?
BNE CHKSYM3               No - check next special case
MVC SYMNAME(8), =C'RONUC'   ' Move in altered symbol name
B SYMØ                   Go on

CHKSYM3 DS ØH
CLC SYMNAME(8), =C'ER/W NUC' Extended R/W NUC?
BNE CHKSYM4               No - check next special case
MVC SYMNAME(8), =C'ERWNUC' Extended R/W NUC?
B SYMØ                   Go on

CHKSYM4 DS ØH
B SYMØ                   Go on

SYMØ DS ØH
MVC SYMREMRK(21), =C'Starting address for' Remark prefix
MVC SYMREMRK+21(8), AREAID Remark suffix

SYM1 DS ØH
LA R7, WORKESSY            Get ESSY area address
MVC Ø(ESSYLR, R7), ESSY    Initialize the area
MVC ESSYSYM-ESSY(32, R7), SYMNAME Copy symbol name
MVC ESSYAST-ESSY(2, R7), =C'CV' Move in address space type
MVC ESSYLD-ESSY(4, R7), AREASTRT Move in start address
L R1, AREASTRT            Copy starting address
L R15, AREAEND            Copy ending address
SR R15, R1                Calculate length
LA R15, 1(, R15)          Add one
STCM R15, B'1111', ESSYDLE-ESSY(R7) Save area length
MVC ESSYDTY-ESSY(1, R7), =C[U'Indicate type as AREA
MVC ESSYDTD-ESSY(32, R7), ESSYSYM-ESSY(R7) Move in data name
MVC ESSYRL-ESSY(2, R7), =AL2(40) Move in remark length
MVC ESSYRT-ESSY(40, R7), SYMREMRK Copy symbol remark

* Determine proper ASID

*--------------------------------------------------------------------*
MVC  ESSYAS2-ESSY+2(2,R7), ASID Move in default ASID
CLC  SYMNAME(3), =C'PVT' Private?
BE   SYSSYM Yes - default ASID is good
CLC  SYMNAME(4), =C'EPVT' Extended Private?
BE   SYSSYM Yes - default ASID is good
MVC  ESSYAS2-ESSY+2(2,R7), =AL2(1) Set ASID to 1

SYSSYM
DS  ØH
OI  ESSYFC-ESSY(R7), ESSYFCD Set NODROP attribute on symbol
L   R15, ADPLSERV Load addr of exit services router
CALL  (15), ((R8), CODEEQS, (R7)), MF=(E,CALLLST)
C   R15, =F'12' Symbol equate ok?
BL   SYM1E Yes - go on
TM   FLAG1, SYMTRY Is this the second try?
BO   NOSYM1 Yes - issue message
OI   FLAG1, SYMTRY Set flag
B    SYM1 Try a second time

SYM1E
DS  ØH
LM   RØ, R14, REGSAVE3 Restore the registers
BR   R14 Return

NOSYM1
DS  ØH
MVI  LINEBUF, C' ' Set fill byte
MVC  LINEBUF+1(131), LINEBUF Clear the area
MVC  LINEBUF(L' SYMDMSG1), SYMDMSG1 Copy the message
L   R1, LINEBUF+L' SYMDMSG1 Point to target area
L   R14, SYMLEN Get symbol length
MVC  Ø(32, R1), SYMNAME Copy symbol name
LA   R1, Ø(R14, R1) Point to target area
MVC  Ø(9, R1), =C' - RC( )' Move in remainder of message
BAL  R14, HEXCNVT Make the rc readable
MVC  6(2, R1), DBL1+6 Copy rc into message
L   R14, SYMLEN Get symbol length
LA   R14, Ø( R14, R14) Get message length
LA   R1, LINEBUF Get message address
BAL  R14, PRINTLN Go print the line
LA   RØ, 1 Set message length
LA   R1, =C' ' Get message address
BAL  R14, PRINTLN Go print a blank line
B    SYM1E Go back for more

*---------------------------------------------------------------------*
* Executed instructions *
*---------------------------------------------------------------------*

*---------------------------------------------------------------------*
* BLNKCLC CLC Ø(*-*, R5), =3C' ' Remaining parm data blanks?
* MSGMVC MVC Ø(*-*, R15), Ø(R1) Copy the message *
*---------------------------------------------------------------------*
* Constants *
*---------------------------------------------------------------------*

CODEACC DC  A(ADPLSACC) Storage access service code
CODEPR2 DC A(ADPLSPR2) Expanded print service code
CODEEQS DC A(ADPLSEQS) Equate symbol service code

*--------------------------------------------------------------------*
PPR2 BLSUPPR2 DSECT=NO IPCS expanded print parm list
*--------------------------------------------------------------------*
ESSY BLSRESSY DSECT=NO IPCS equate symbol parm list

*--------------------------------------------------------------------*
PARMMSG1 DC C'STRMP11ØI - Invalid parm detected. Valid parms are:
DC C' EPVT ECSA EMLPA EFLPA EPLPA ESQA ERWNUC RONUC RWNUC
DC C' SQA PLPA FLPA MLPA CSA PVT'
PRMMSG1L EQU *-PARMMSG1

*--------------------------------------------------------------------*
STORMSG DC C'STRMP112I - Unable to locate xxxx at xxxxxxxxx -
DC C' RC(xx)'
STMSGL EQU *-STORMSG

*--------------------------------------------------------------------*
CBEMSG1 DC C'STRMP132I - Invalid xxxx control block detected at
DC C' xxxxxxxx'
CBEMSG1L EQU *-CBEMSG1

*--------------------------------------------------------------------*
TERMMSG1 DC C'STRMP189I - Storage map display terminated.
TRMMSG1L EQU *-TERMMSG1

*--------------------------------------------------------------------*
SYMDMSG1 DC C'STRMP131I - Error detected defining symbol
DC C' RC(xx)'
SYDMSG1L EQU *-SYMDMSG1

*--------------------------------------------------------------------*
TOPBOTM DC CL5Ø'+-------------------+
FILL DC CL5Ø' ] ] ]
MIDDLE DC CL5Ø' ] ] ]
MB16 DC CL5Ø' 16MB ] ] ]
LNOFFST1 EQU 3
LNOFFST2 EQU 17
LNOFFST3 EQU 29

*--------------------------------------------------------------------*
AREAMSG1 DC C' No storage area available for
AREAMSG2 DC C' xxxxxxxx start address: 
AREAMSG3 DC C' end address:

*--------------------------------------------------------------------*
LTORG ,

*--------------------------------------------------------------------*
WORKAREA DSECT
SAVEAREA DS 18F
CALLLST CALL ,(,,,,,), MF=L
REGSAVE DS 16F Subroutine register save area
REGSAVE2 DS 16F Subroutine register save area
REGSAVE3 DS 16F Subroutine register save area
ASID DS XL2

*--------------------------------------------------------------------*
AREAatr DS F Area starting address
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AREAEND</td>
<td>DS</td>
<td>Area ending address</td>
</tr>
<tr>
<td>AREAID</td>
<td>DS</td>
<td>Area identifier</td>
</tr>
<tr>
<td>PARMADDR</td>
<td>DS</td>
<td>Incoming parameter address</td>
</tr>
<tr>
<td>GDADDR</td>
<td>DS</td>
<td>Address of the GDA from the CVT</td>
</tr>
<tr>
<td>MLPAS</td>
<td>DS</td>
<td>Starting virtual addr of MLPA</td>
</tr>
<tr>
<td>MLPAE</td>
<td>DS</td>
<td>Ending virtual addr of MLPA</td>
</tr>
<tr>
<td>FLAS</td>
<td>DS</td>
<td>Starting virtual addr of FLPA</td>
</tr>
<tr>
<td>FLPAE</td>
<td>DS</td>
<td>Ending virtual addr of FLPA</td>
</tr>
<tr>
<td>PLAS</td>
<td>DS</td>
<td>Starting virtual addr of PLPA</td>
</tr>
<tr>
<td>PLPAE</td>
<td>DS</td>
<td>Ending virtual addr of PLPA</td>
</tr>
<tr>
<td>RWNS</td>
<td>DS</td>
<td>Starting virtual addr of R/W Nuc</td>
</tr>
<tr>
<td>RWNE</td>
<td>DS</td>
<td>Ending virtual addr of R/W Nuc</td>
</tr>
<tr>
<td>RONS</td>
<td>DS</td>
<td>Starting virtual addr of R/O Nuc</td>
</tr>
<tr>
<td>RONE</td>
<td>DS</td>
<td>Ending virtual addr of R/O Nuc</td>
</tr>
<tr>
<td>ERWNS</td>
<td>DS</td>
<td>Starting virtual addr of ER/W Nuc</td>
</tr>
<tr>
<td>ERWNE</td>
<td>DS</td>
<td>Ending virtual addr of ER/W Nuc</td>
</tr>
<tr>
<td>EPLAS</td>
<td>DS</td>
<td>Starting virtual addr of EPLPA</td>
</tr>
<tr>
<td>EPLPAE</td>
<td>DS</td>
<td>Ending virtual addr of EPLPA</td>
</tr>
<tr>
<td>EFLAS</td>
<td>DS</td>
<td>Starting virtual addr of EFLPA</td>
</tr>
<tr>
<td>EFLPAE</td>
<td>DS</td>
<td>Ending virtual addr of EFLPA</td>
</tr>
<tr>
<td>EMLAS</td>
<td>DS</td>
<td>Starting virtual addr of EMLPA</td>
</tr>
<tr>
<td>EMLPAE</td>
<td>DS</td>
<td>Ending virtual addr of EMLPA</td>
</tr>
<tr>
<td>CSA</td>
<td>DS</td>
<td>Starting virtual addr of CSA</td>
</tr>
<tr>
<td>CSASZ</td>
<td>DS</td>
<td>Size of CSA</td>
</tr>
<tr>
<td>CSAE</td>
<td>DS</td>
<td>Ending virtual addr of CSA</td>
</tr>
<tr>
<td>SQA</td>
<td>DS</td>
<td>Starting virtual addr of SQA</td>
</tr>
<tr>
<td>SQASZ</td>
<td>DS</td>
<td>Size of SQA</td>
</tr>
<tr>
<td>SQAE</td>
<td>DS</td>
<td>Ending virtual addr of SQA</td>
</tr>
<tr>
<td>ECSA</td>
<td>DS</td>
<td>Starting virtual addr of ECSA</td>
</tr>
<tr>
<td>ECSAS</td>
<td>DS</td>
<td>Size of ECSA</td>
</tr>
<tr>
<td>ECSAE</td>
<td>DS</td>
<td>Ending virtual addr of ECSA</td>
</tr>
<tr>
<td>ESQA</td>
<td>DS</td>
<td>Starting virtual addr of ESQA</td>
</tr>
<tr>
<td>ESQAS</td>
<td>DS</td>
<td>Size of ESQA</td>
</tr>
<tr>
<td>ESQAE</td>
<td>DS</td>
<td>Ending virtual addr of ESQA</td>
</tr>
<tr>
<td>PVT</td>
<td>DS</td>
<td>Starting virtual addr of Private</td>
</tr>
<tr>
<td>PVTSZ</td>
<td>DS</td>
<td>Size of Private</td>
</tr>
<tr>
<td>PVTE</td>
<td>DS</td>
<td>Ending virtual addr of PVT</td>
</tr>
<tr>
<td>EPVT</td>
<td>DS</td>
<td>Starting virtual addr of EPrivate</td>
</tr>
<tr>
<td>EPVTS</td>
<td>DS</td>
<td>Size of EPrivate</td>
</tr>
<tr>
<td>EPVTE</td>
<td>DS</td>
<td>Ending virtual addr of PVTE</td>
</tr>
<tr>
<td>PREVAREA</td>
<td>DS</td>
<td>Previous mapped area's start addr</td>
</tr>
<tr>
<td>CVTADDR</td>
<td>DS</td>
<td>Previous mapped area's start addr</td>
</tr>
<tr>
<td>PARMFLGS</td>
<td>DS</td>
<td>Parm flag 1</td>
</tr>
<tr>
<td>PRMVPVT</td>
<td>EQU</td>
<td>X'80'</td>
</tr>
<tr>
<td>PRMCSA</td>
<td>EQU</td>
<td>X'40'</td>
</tr>
</tbody>
</table>

AMDOSEL=NO,
AMDPACC=YES,
AMDPFMT=YES,
AMDPCT=NO,
AMDPSEL=NO

PRINT NOGEN
CVT DSECT=YES
IHAGDA ,
END
MVS news

Compute (Bridgend) has announced Release 2.10 of CBLVCAT for Mainframe, its VSAM file tuning, VSAM catalog display, and VTOC display and modification software.

CBLVCAT has been updated to support 31-bit addressing and is now linked as AMODE=31 on all platforms. This overcomes previous storage restrictions, allowing CBLVCAT to utilize buffers in storage above the 16MB line.

Also new is the CBLVCAT Interactive (VCI) Environment, which is a component of the new CBL Interactive (CBLi) Interface that provides all authorized users with an interactive environment for executing CBLVCAT and SELCOPY.

The VCI component of CBLi allows interactive execution of CBLVCAT control statements sourced from a data set or via a command line. The generated report is stored in internal buffers and presented to the user in a window area with coloured highlighting. The report may be edited and optionally saved to a data set.

Where LISTVCAT option DEFINE is specified, an edit window is automatically opened for the CBLVCAT generated IDCAMS DEFINE job, thus allowing alteration by the user before submission.

For further information contact:
Compute (Bridgend) Ltd, 8 Merthyr Mawr Road, Bridgend, Wales CF31 3NH, UK.
Tel: (01656) 652222.

* * *

Axios has announced Version 4.1.0 of SmartAnalyzer, its back-up reliability analysis software. The new version provides users with more details regarding the reliability of their onsite application data back-ups and migrations.

SmartAnalyzer identifies problems and inefficiencies in the way data is managed by either the HSM or DMS disk management systems. It identifies missing and unusable back-ups, unnecessary back-ups, back-ups taken by non-standard tools, and more. It provides comprehensive auditing capabilities for maintenance of data availability and recoverability.

For further information contact:
Axios, 1373-10 Veterans Highway, Hauppauge, NY 11788, USA.
Tel: (631) 979 0100.

* * *

Open Software Technologies has announced Version 6.1 of REXXTOOLS/MVS. The product adds access method interfaces, plus many REXX language extensions that increase the functionality of the REXX language.

REXXTOOLS/MVS contains three components, which are Basic Services, and Dynamic SQL Services, and Static SQL Services.

6.1 has a new CICS external interface, an external writer facility, extended addressing or VSAM and the QSAM interface now supports data sets with spanned records.

For further information contact:
Open Software Technologies, PO Box 162652, Altamonte Springs, FL 32716-2652, USA.
Tel: (407) 788 7173.
URL: http://www.open-softech.com/mvs2.htm#newbasic.