230

MVS update

November 2005

In this issue

A xephon PUBLICATION

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

Colin Smith
E-mail: info@xephon.com

Subscriptions and back-issues

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

MVS Update on-line

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

Contributions

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

Printed in England.

Deleted datasets at a glance – revisited

We reviewed the article 'Deleted datasets at a glance' (see *MVS Update*, issue 224, May 2005) with much interest. We also like to use ICETOOL to perform simple data manipulation and reporting. There are two features of ICETOOL that we use quite extensively. The first of these is the OUTFIL feature. This allows you to process a single input dataset and produce multiple output datasets. The second feature is the use of symbols to simplify data element reference. We have provided three symbol definitions as well as the JCL utilized to produce three output files. Our sample JCL requires only a single job step to process the data and produce the reports.

In the sample JCL below, the SAVEFILE DD statement can be used to save all the type 17 and 18 records that are processed. In the sample we have it set up as a temporary dataset. Simple JCL changes make this a permanent dataset. You could choose to make this a GDG or a single dataset that could have data appended into it. REPORT1 and REPORT2 are the two report files that this sample produces. You can add OUTFIL control statements if you wish to produce additional reports. Note the use of symbols for all the data elements and how they are used in the INCLUDE statement to ignore DFHSM datasets. You should consult the DFSORT documentation for more information about the ICETOOL control statements.

SAMPLE JCL

```
//jobname  JOB acct,user,CLASS=a,MSGCLASS=x,
//            REGION=ØM
//STEPØØ1Ø EXEC PGM=ICETOOL
//STEPLIB  DD DISP=SHR,DSN=only.if.needed
//SYMNAMES DD DISP=SHR,DSN=our.symbol.library(SMFHEADR)
//         DD DISP=SHR,DSN=our.symbol.library(SMF17)
//         DD DISP=SHR,DSN=our.symbol.library(SMF18)
//INPUT    DD DISP=SHR,DSN=our.smf.input.file
//SAVEFILE DD DISP=(NEW,DELETE),DSN=&&SAVEFILE,
//            UNIT=SYSALLDA,SPACE=(CYL,(1Ø,1Ø)),
//            DSORG=PS,RECFM=VBS,BLKSIZE=Ø
```

```
//TOOLMSG  DD SYSOUT=*   ICETOOL MESSAGES
//DFSMSG   DD SYSOUT=*   DFSORT  MESSAGES
//SYMNOUT  DD SYSOUT=*
//REPORT1  DD SYSOUT=*
//REPORT2  DD SYSOUT=*
//TOOLIN   DD *
  COPY FROM(INPUT) USING(CPY1)
/*
//CPY1CNTL DD *
 OPTION SPANINC=RC4,VLSHRT
 OUTFIL FNAMES=SAVEFILE,
  INCLUDE=(SMFXRTY,EQ,17,OR,SMFXRTY,EQ,18)
 OUTFIL FNAMES=REPORT1,CONVERT,LINES=55,
  INCLUDE=(SMFXRTY,EQ,17,AND,SMF17DSN_8,NE,C'DFHSM.VT',AND,
           SMF17DSN_8,NE,C'DFHSM.HM'),
  OUTREC=(SMFXSID,8:SMFXDTE,DT1,EDIT=(TTTT/TT/TT),20:SMFXTME,TM1,
          EDIT=(TT:TT:TT),33:SMF17JBN,46:SMF17FVL,58:SMF17DSN),
 HEADER2=(1:DATE,35:'OUR COMPANY NAME IF DESIRED',90:TIME,/,
           39:'DELETED DATA SET REPORT',/,/,
           1:'SMFID',11:'DATE',22:'TIME',35:'JOB',46:'VOLSER',
           64:'DATA SET NAME',/,1:101'-')
 OUTFIL FNAMES=REPORT2,CONVERT,LINES=55,
  INCLUDE=(SMFXRTY,EQ,18),
  OUTREC=(SMFXSID,8:SMFXDTE,DT1,EDIT=(TTTT/TT/TT),20:SMFXTME,TM1,
          EDIT=(TT:TT:TT),29:SMF18JBN,37:SMF18FVL,44:SMF18ODS,
          89:SMF18NDS),
 HEADER2=(1:DATE,35:'OUR COMPANY NAME IF DESIRED',90:TIME,/,
           39:'RENAMED DATA SET REPORT',/,/,
           1:'SMFID',11:'DATE',22:'TIME',30:'JOB',37:'VOLSER',
           46:'OLD DATA SET NAME',
           93:'NEW DATA SET NAME',/,1:131'-')
/*
//*
```

SMFHEADR ICETOOL SYMBOL DEFINITION

```
*---+----+----+----+----+----+----+----+----+----+----+----+----+----*
*                                                                     *
* ICETOOL SYMBOL MAP FOR SMF RECORD HEADER                           *
*                                                                     *
* This layout will accommodate both types of SMF records, those with *
* subtypes and those without.  If the individual record layout is for*
* a record without subtypes, the symbol definition for the actual SMF*
* record should begin with a POSITION,19 directive.                  *
*                                                                     *
* Source documentation: SA22-7630-7                                  *
*                                                                     *
*---+----+----+----+----+----+----+----+----+----+----+----+----+----*
SMF_RDW,1,4,BI                        Record descriptor word
```

```
SMFXLEN,=,2,BI                  Record length
SMFXSEG,*,2,BI                  Segment number
SMFXFLG,*,1,BI                  Flag byte
SMFXRTY,*,1,BI                  SMF record type
SMFXTME,*,4,BI                  Time since midnight in
*                               hundredths of a second, since
*                               the record was moved into the
*                               SMF buffer
SMFXDTE,*,4,PD                  Date when the record was moved
*                               into the SMF buffer, in the form
*                               ØcyydddF, where c is Ø for 19xx
*                               and c is 1 for 2Øxx.
SMFXSID,*,4,CH                  System identifier
SMFXSSI,*,4,CH                  Subsystem identifier
SMFXSTY,*,2,BI                  Subtype indicator
```

SMF17 ICETOOL SYMBOL DEFINITION

```
*---+----+----+----+----+----+----+----+----+----+----+----+----+----*
*                                                                     *
* ICETOOL SYMBOL MAP FOR SMF RECORD TYPE 17                          *
*                                                                     *
* Source documentation: SA22-763Ø-7                                  *
*                                                                     *
*---+----+----+----+----+----+----+----+----+----+----+----+----+----*
POSITION,19
SMF17JBN,*,8,CH                 Job name
SMF17RST,*,4,BI                 Time since midnight, in
*                               hundredths of a second since the
*                               reader recognized the job card
*                               for this job
SMF17RSD,*,4,BI                 Date when the reader recognized
*                               the job card for this job, in
*                               the form ØcyydddF.
SMF17UID,*,8,CH                 User-defined identification
*                               field.  Taken from common exit
*                               parameter area, not from USER=
*                               value on the job card
SMF17RIN,*,2,BI                 Reserved
SMF17DSN_1,*,1,CH               First character of DSN
POSITION,SMF17DSN_1             Reposition
SMF17DSN_2,*,2,CH               First 2 characters of DSN
POSITION,SMF17DSN_1             Reposition
SMF17DSN_3,*,3,CH               First 3 characters of DSN
POSITION,SMF17DSN_1             Reposition
SMF17DSN_4,*,4,CH               First 4 characters of DSN
POSITION,SMF17DSN_1             Reposition
SMF17DSN_5,*,5,CH               First 5 characters of DSN
POSITION,SMF17DSN_1             Reposition
```

```
SMF17DSN_6,*,6,CH                       First 6 characters of DSN
POSITION,SMF17DSN_1                     Reposition
SMF17DSN_7,*,7,CH                       First 7 characters of DSN
POSITION,SMF17DSN_1                     Reposition
SMF17DSN_8,*,8,CH                       First 8 characters of DSN
POSITION,SMF17DSN_1                     Reposition
SMF17DSN,*,44,CH                        Data set name, all 44 chars.
SMF17RV1,*,3,BI                         Reserved
SMF17NVL,*,1,BI                         Number of volumes
SMF17RV2,*,2,BI                         Reserved
SMF17FVL,*,6,CH                         First volser
```

SMF18 ICETOOL SYMBOL DEFINITION

```
*---+----+----+----+----+----+----+----+----+----+----+----+----+----*
*                                                                     *
* ICETOOL SYMBOL MAP FOR SMF RECORD TYPE 18                          *
*                                                                     *
* Source documentation: SA22-7630-7                                  *
*                                                                     *
*---+----+----+----+----+----+----+----+----+----+----+----+----+----*
POSITION,19
SMF18JBN,*,8,CH                         Job name
SMF18RST,*,4,BI                         Time since midnight, in
*                                       hundredths of a second since the
*                                       reader recognized the job card
*                                       for this job
SMF18RSD,*,4,PD                         Date when the reader recognized
*                                       the job card for this job, in
*                                       the form 0cyydddF.
SMF18UID,*,8,CH                         User-defined identification
*                                       field.  Taken from common exit
*                                       parameter area, not from USER=
*                                       value on the job card
SMF18RIN,*,2,BI                         Reserved
SMF18ODS_1,*,1,CH                       First character of DSN
POSITION,SMF18ODS_1                     Reposition
SMF18ODS_2,*,2,CH                       First 2 characters of DSN
POSITION,SMF18ODS_1                     Reposition
SMF18ODS_3,*,3,CH                       First 3 characters of DSN
POSITION,SMF18ODS_1                     Reposition
SMF18ODS_4,*,4,CH                       First 4 characters of DSN
POSITION,SMF18ODS_1                     Reposition
SMF18ODS_5,*,5,CH                       First 5 characters of DSN
POSITION,SMF18ODS_1                     Reposition
SMF18ODS_6,*,6,CH                       First 6 characters of DSN
POSITION,SMF18ODS_1                     Reposition
SMF18ODS_7,*,7,CH                       First 7 characters of DSN
POSITION,SMF18ODS_1                     Reposition
```

```
SMF18ODS_8,*,8,CH                First 8 characters of DSN
POSITION,SMF18ODS_1              Reposition
SMF18ODS,*,44,CH                 Old data set name
SMF18NDS,*,44,CH                 New data set name
SMF18RV1,*,3,BI                  Reserved
SMF18NVL,*,1,BI                  Number of volumes
SMF18RV2,*,2,BI                  Reserved
SMF18FVL,*,6,CH                  First volser
```


Restructuring of MVS consoles in z/OS 1.5

When it was designed, the original console component of a sysplex worked best in a small sysplex filled with similar systems. As sysplexes have grown, users have experienced some issues with console processing. In some cases this has severely disrupted services.

IBM has addressed this by enhancing the infrastructure of messages to try to provide greater reliability and availability of the system and sysplex, and to remove and reduce system outages caused in particular by message flooding.

Previously, console facilities existed in each system in the sysplex and had the same message processing. Messages were sent from one system to another using XCF without any regard to whether the receiving system was able to handle the messages. The receiving system had no choice but to attempt to consume the messages that it was forced to receive.

Message production and the processing of the messages can cause problems. Messages are delivered only once to a system that has multiple interested consoles, but messages must still be individually delivered to each system that has consoles receiving the messages. The actual overhead of delivering messages to other systems rises as the number of systems in the sysplex increases. This results in issues if an

application that is creating messages goes into a loop and operators normally see buffer shortages and out-of-storage conditions that can bring down both the system that is the target for the messages and the system that is the source of the messages. All messages would arrive on a single queue and are queued by a single task to any local consoles, to the SYSLOG, to the OPERLOG, and to the three EMCS console queuers. Any disruption to the initial queueing task affects the queueing to all consoles and to the logs. Once a message has been queued to one of the EMCS queuers, queueing to its consoles proceeds independently of queueing to the consoles owned by the other two EMCS queuers. Because a message can be queued to multiple consoles, the storage that the message occupies cannot be re-used until it is processed by the last console for which it is designated. Once displayed, the message is de-queued.

Messages are delivered only once to a system that may have multiple consoles receiving the messages. The more systems that are in a sysplex, the greater the overhead of delivering messages because the number of XCF writes will increase. The restructuring of console processing has attempted to overcome the following issues. If a problem exists on a single local console, message queue storage shortage problems can occur and this has a knock-on effect on the other local consoles, the log, and EMCS consoles. Once a message is queued to an EMCS console queuer, the message is copied one or more times into EMCS message data spaces. Queueing or storage problems in one data space does not affect queueing or storage in other EMCS message data spaces.

The way multiple line messages are constructed can cause further message delivery problems since a multiple line message can be queued, and sent to other systems in the sysplex before the entire multiple line message is complete. Queueing the incomplete message does not immediately cause problems, but if the message is still incomplete when a console attempts to display the message, it causes the console to stop displaying other messages until the message

it has begun to display is complete. The slowest operator console paces all message traffic. The message is then handed over to the console address space, where it must reside in a 24-bit storage buffer for processing. Any disruption to the initial queueing task affects the queueing to all consoles and the logs.

IBM has enhanced the design of message processing during the console restructure to provide, initially, the following:

- Help in eliminating outages caused by a flood of WTOs and DOMs.
- A new message cache data space.
- Reduced dependence for message processing on a single task.
- Message queueing independence.
- Messages delivered to the SYSLOG and OPERLOG from the caller's unit of work.
- Queueing to the SYSLOG and OPERLOG is separated from queueing to the consoles.
- Queueing to the EMCS consoles is performed in parallel and prior to queueing to the local consoles.
- MLWTOs (multi-line WTOs) are not queued for delivery until all messages have been received.

This list of enhancements is now discussed in a little more detail in the following paragraphs.

Any processes generating messages have to place their messages and message deletion requests into a single common pool from which the message consumers can extract their messages and message deletion requests. The pool will decouple at the rate at which messages and message deletion requests are produced plus the rate at which individual message consumers consume messages. At the sysplex level, this means that messages and message deletion

requests are written once to a log stream that is common to the entire sysplex. At a system level, messages and message deletion requests are read from the common log stream and written to a data space that is common to all the consoles on the system.

The console message cache data space has been designed to decouple message reception from other systems from the queueing of messages to consoles. The console message cache data space decouples at the rate at which messages and message deletion requests are read from the common log stream plus the rate at which those messages can be queued and consumed by the console class queuers.

By changing DIDOCS (Device Independent Display Operator Console) queueing to the end of processing rather than the beginning, the dependence on the one main queueing task is eliminated.

Moving message processing into the user address space reduces reliance on a single task.

Console services also make use of enhancements to the XCF protocols to request ordered delivery of messages and message deletion requests. This should ensure that message deletion requests do not arrive before the message they are intended to delete, and it should ensure that the pieces of a multiple line message arrive in the order in which they were created. Remember this is a ‘should’ and not a guarantee.

The dependency on EMCS consoles and hardcopy processing having the ability to queue to MCS consoles has been removed. The hardcopy log has been moved to the front of command processing and now occurs in the WTO issuer’s task. Delivery is no longer dependent on the health or speed of the DIDOCS queuer. Hardcopy queueing sits on the EMCS interface, but receives messages sent directly from the SVC 35 issuance. It doesn’t rely on the message cache mechanism for delivery.

Any queueing for SYSLOG and OPERLOG is now executed as part of the message issuer’s thread after the message has

passed through the Message Processing Facility (MPF) and SubSystem Interface (SSI) processing. Logging performance is no longer dependent on communications task performance, and writing to the log will no longer cause console message buffer shortages. Logging problems will now affect individual message producers, not the console function.

The three EMCS message queuers will queue messages from the console message cache data space to EMCS consoles independently of each other. All the local consoles are represented as a single EMCS console and messages are then queued from the EMCS message data space to the individual MCS and SMCS consoles.

Incomplete multiple line messages are gathered after MPF and SSI processing into a data space where they will wait until complete. When they are complete they are written to the common log stream from the dataspace. This means that everything on the message path will see only completed multiple line messages. Multiple line messages being constructed incrementally are passed to the MPF exit and the SSI as they are constructed to preserve compatibility with current behaviour. Timeout processing is used to force the completion of multiple line messages which have remained incomplete for a certain period of time.

As well as improving the message processing, there have also been changes to how the system console under MVS works. Also known as the Hardware Management Console (HMC), it was previously designed for use as an initialization and emergency console, the intention being that it should be used at times when MCS consoles could not be active, or were all inactive because of an error. When defined, the system console in the previous versions of z/OS did have a certain number of restrictions associated with it. The operator had to use the system console as a NIP (Nucleus Initialization Program) console. Operators would issue the **VARY CN(*),ACTIVATE** command when NIP processing was over, possibly losing a few messages, then issue the **VARY CN(*),**

DEACTIVATE against the system console. IBM has decided that this is not the best way of processing and decided that an automatic method of activating the system console is required. Also it should be possible to deactivate it automatically when deemed no longer required.

You can instigate the new way of operating using the AUTOACT operand of the CONSOLE statement in the CONSOL*xx* member of SYS1.PARMLIB. The system console would be defined as follows:

```
CONSOLE DEVNUM(SYSCONS) AUTOACT(groupname)
```

If *AUTOACT (groupname)* is specified in CONSOL*xx* on the CONSOLE statement, *groupname* is the name of a console group, as defined in CNGRP*xx*.

AUTOACT specifies the 'automatic activate group' for the system console. If an automatic activate group (AUTOACT) is active for the system console, the system automatically issues the **VARY CN(syscons),ACTIVATE** and **VARY CN(syscons),DEACTIVATE** commands for you.

However, while the AUTOACT group is defined and not suspended, the system console is automatically placed into problem determination (PD) mode when all the consoles in AUTOACT are inactive. The system console is automatically removed from PD mode when any console in the AUTOACT group becomes active.

To suspend AUTOACT processing, issue a **VARY CN(*), DEACTIVATE** command from the system console itself. This will remove automatic processing until the opposite command is issued to turn it back on. This means that, if required, you can easily revert to the old method of working. Full details on the required members in SYS1.PARMLIB and relevant statements can be found in the *MVS Initialization and Tuning Guide*. Changes to MVS commands around console processing can be found in the *MVS Commands Reference* manual. If an operator is using the system console during NIP, he can continue using it without interruption or manual intervention.

When a real console defined as a member of the AUTOACT group is activated, the system console is deactivated automatically; the operator does not have to remember to turn it off. If all of the consoles in the group become inactive, the system console comes up automatically.

As with any new features there are a number of things to be considered:

- The ALTGRP keyword has been available since MVS/ESA 4.2.0. It is now the required method to specify back-up consoles. The ALTERNATE keyword on the CONSOLE statement in CONSOL*xx* is no longer accepted.
- All messages will still appear in the log, but the code no longer takes extra effort determining whether a message has been delivered to a console, or in redirecting the message to a console.
- Undeliverable messages (UD) are no longer detected. The UD keyword on the CONSOLE and HARDCOPY statements in the CONSOL*xx* parmlib member are no longer accepted. The UD keyword is no longer supported on the VARY CONSOLE and VARY HARDCPY commands.
- Hardcopy must now be either SYSLOG or OPERLOG. The HCPYGRP keyword on the HARDCOPY statement in CONSOL*xx* is no longer accepted. Printer devices can still be used, but they cannot be specified as the hardcopy medium.
- Messages queued to a console can no longer be redirected to another console. The **CONTROL Q** command can still be used to indicate that they don't need to be seen. They will still appear in hardcopy.
- The NAME keyword on the CONSOLE statement in CONSOL*xx* parmlib member is now required. All consoles must be explicitly named except for the system console, where the code will continue to create a name if none is supplied by the systems programmer. Console definitions will be rejected if no name is specified.

- The MSCOPE keyword on the CONSOLE statement now defaults to * instead of *ALL.

For the new code to be utilized, a number of compatibility fixes are required. All systems in a sysplex must be at one of the following levels of maintenance:

- z/OS V1R5.
- z/OS V1R4 with SDSF APAR PQ73805 installed.
- z/OS V1R4 with APAR OW56244 and SDSF APAR PQ73805 installed.
- Any level between OS/390 V2R10 and z/OS V1R3 with APAR OW56244.

John Bradley
Systems Programmer
Meerkat Computer Services (UK)

Catalog back-up process – revisited yet again

There is virtually no z/OS mainframe installation that does not have a disaster recovery (DR) plan of one kind or another. The aim of making a good DR plan is to help survive an outage of the overall data centre. The DR plan typically has elaborate scenarios and it is quite usual to see that many installations test at least some portions of their DR plan as often as every month. On the other hand, it is also quite common to see that at most z/OS installations, system-supporting teams already have their hands full with daily problems and processes. Since an ICF catalog failure is not an everyday occurrence, planning for a catalog failure is not a priority task. As is commonly known, a z/OS system may have tens, hundreds of thousands, or even millions of datasets stored on tapes or disk volumes. Finding the exact location of these datasets is assisted by

catalogs. The ICF catalog is where all datasets are catalogued, and access to datsets is possible only through a successful catalog search. The loss of a single catalog can result in the loss of access to thousands of datasets. The loss of a master catalog means an IPL of at least one system. In a sysplex environment where the master catalog is shared, a broken master catalog may mean an outage of the entire sysplex. That being so, one must say that the catalogs are essential system datasets. The catalog back-ups exist solely to support recovery requirements and, hopefully, will never be needed. Therefore, the most important attribute of any back-up is that it is readily available and usable when needed.

Many factors are involved in determining how frequently a given catalog should be backed up, but the main one is an assessment of the impact of an outage and how quickly it must be recovered. A recoverability objective for any dataset is a statement of the level of currency to which it must be recovered and how long it can take. In most installations, some catalogs are very volatile and some are relatively static in terms of the amount of allocate and delete activity. (Note: you can use the MODIFY CATALOG command to list information about catalogs currently allocated to the catalog address space as well as to evaluate the cache activity and performance for a specified catalog, or for all catalogs that are currently being cached. See the *DFSMS: Managing Catalogs* (SC26-7409) manual for additional information on the MODIFY CATALOG command). Thus, a daily back-up may be sufficient for some catalogs, while others may need to be backed up every few hours. On the other hand, restoring a catalog can be a relatively trivial task, but recovering it to an acceptable level of currency can be very complex and time consuming. Therefore, the time needed to recover a catalog may depend on how recently the back-up copy was made. There are a number of different techniques to back up catalogs. Usually it is the storage administrator who decides which back-up technique is appropriate. This can be based on speed, utilities available, or other business factors such as auditor requirements. The

most important consideration when deciding on a back-up technique is that you perform some form of checking to ensure that the back-up of the catalog was successful. IBM recommends that you test your back-up and recovery procedures to be sure that the back-up copy you created is usable in the event it is needed. You may also want to try different utilities to see which one best suits your environment and the ease with which post-recovery updates can take place. The 'Backing up catalogs' chapter of an IBM Redbook *ICF Catalog Backup and Recovery: A Practical Guide* (SG24-5644-01) discusses the techniques and utilities that are available for performing a successful back-up. I would recommend that you use at least two different methods to back up your catalogs, then you should be able to recover from at least one of them. In addition to that, master catalogs are so critical that it is best if a point-in-time copy is taken and refreshed after every major update. If you cannot do this, at the very least make sure you have a full volume dump that you can restore using stand-alone DFDSS.

The article 'Catalog back-up process' published in *VSAM Update* (issue 5, April 1992) and its follow up ('Catalog back-up process – revisited', *VSAM Update*, issue 6, July 1992) inspired me to add the following note hoping that it will help the storage administrator in keeping track of catalogs being exported. While the original article is good and quite useful in discussing the catalog back-up process, the reporting procedure supplied is less than useful since the code was written in the SAS language, which is not available at many mainframe sites. It also uses a program found in MXG Software (from Merrill Consultants), which is an excellent tool itself, but, because it is a SAS-based software package, it requires the SAS BASE product to be installed on your system.

What I really needed was a simple, flexible, and yet fast reporting tool that would quickly report on catalogs being exported. In order to provide a simple and effective way to gather the data needed to monitor and analyse Integrated

Catalog Facility export activity, an easy-to-use reporting procedure has been created. Let us remember that the SMF records provide a wealth of performance data, which is often difficult to analyse in its raw form. You need to convert this raw data into processed data that can be used for trend analysis and management reports. If you do not have access to a performance analysis and reporting product (such as MXG Software), you can use the DFSORT editing functions in combination with the reporting capabilities of ICETOOL or OUTFIL to generate performance data reports. The INCLUDE/OMIT, OUTFIL, and INREC/OUTREC control statements are helpful in selecting the particular fields of the particular SMF, RMF, or other data records to be analysed. ICETOOL or OUTFIL can then be used to generate printable reports from the resulting raw data. See the *z/OS DFSORT Application Programming_Guide* (SC26-7523-00) for more information about ICETOOL, OUTFIL, and editing functions. In the case of a catalog export, a record type 36 is partially built when an ICF catalog is exported; it is then compiled and written. The record contains information to identify the catalog being exported, the time of the export, and information necessary to allocate the portable dataset for subsequent import. It identifies the job-by-job log identification and user identification. Please note that the SMF record type 36 is only written upon successful completion of the EXPORT command.

The code is a two-part stream. In the first part (COPYSMF) selected SMF records are copied from the SMF dataset to a file that can be used as a base for the archived records. In the second part of the code (ICFEXP), the captured records are formatted and the ICF catalog export report is produced.

```
//DEL       EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=X
//SYSIN     DD *
     DELETE hlq.SMF36
     SET MAXCC=Ø
/*
//*-------------------------------------------------------------------*
//* UNLOAD SMF 36 RECORDS FROM VSAM OR VBS TO VB                      *
//* Note: change the DUMPIN DSN=your.smfdata to be the name of        *
```

```
//* the dataset where you currently have SMF data being            *
//* recorded. It may be either an SMF weekly dataset or an active  *
//* dump dataset. If you chose the later, then prior to            *
//* executing this job, you need to terminate SMF recording        *
//* the currently active dump dataset for allow the                *
//* unload of SMF records.                                         *
//* Also, change the DCB reference to match the name of your       *
//* weekly SMF dump dataset.                                       *
//*----------------------------------------------------------------*
//COPYSMF  EXEC PGM=IFASMFDP,REGION=ØM
//DUMPIN   DD DSN=hlq.smf.dataset,DISP=SHR
//DUMPOUT  DD DSN=hlq.SMF36,DISP=(NEW,CATLG),UNIT=SYSDA,
//         SPACE=(CYL,(x,y),RLSE),
//         DCB=(hlq.smfweekly.dataset)
//SYSPRINT DD SYSOUT=X
//SYSIN DD *
    INDD(DUMPIN,OPTIONS(DUMP))
    OUTDD(DUMPOUT,TYPE(364))
/*
//*----------------------------------------------------------------*
//* COPY VBS TO VB, DROP HEADER/TRAILER RECORDS, SORT ON DATE/TIME *
//* Note: change the SMF36 DSN=hlq.SMF36 to be the name of         *
//* the dataset you have used in the previous step.                *
//*----------------------------------------------------------------*

//ICFEXP   EXEC PGM=ICETOOL
//DFSMSG   DD SYSOUT=*
//SRT1IN   DD DSN=hlf.SMF36,DISP=SHR
//SHOW     DD DSN=&&SMF36OUT,DISP=(,PASS),SPACE=(CYL,(1,1))
//ICFRPT   DD SYSOUT=*
//SRT1CNTL DD *
* Include SMF 36 records only
* Sort by Export date and time
  OPTION DYNALLOC,VLSHRT,SPANINC=RC4
  SORT FIELDS=(143,8,CH,A,151,8,CH,A)
  INCLUDE COND=(6,1,BI,EQ,36)
  OUTFIL FNAMES=SHOW,CONVERT, * Build a new output record
  OUTREC=(213,2,CHANGE=(25,   * Export indicator
             C'AE',C'Aliases were exported',
             C'NE',C'No aliases exported'),
             57,8,   * Job
             143,8,  * Export date
             151,8,  * Export time
             81,8,   * Program
             89,44,  * Catalog name
             133,6,  * Catalog volser
             2Ø3,6,  * Export  volser
             159,44) * Export  ds name
/*
//TOOLMSG DD SYSOUT=*
```

```
//REPORT  DD SYSOUT=*
//TOOLIN  DD *
* Print ICF catalog export report
* Get first 20 out of 44 Catalog name chars.
* Also, print only first 20 chars. of Export dsn
  COPY FROM(SRT1IN) TO(SHOW) USING(SRT1)
  DISPLAY FROM(SHOW) LIST(ICFRPT)           -
  TITLE('ICF Catalog Export report')        -
  PAGE DATE TIME BLANK                      -
  HEADER('Job')                ON(26,8,CH)  -
  HEADER('Export date')        ON(34,8,CH)  -
  HEADER('Export time')        ON(42,8,CH)  -
  HEADER('Program')            ON(50,8,CH)  -
  HEADER('Catalog')            ON(58,20,CH) -
  HEADER('Cat.volser')         ON(102,6,CH) -
  HEADER('Export volser')      ON(108,6,CH) -
  HEADER('Export dsn')         ON(114,20,CH)-
  HEADER('Export Indicator')   ON(1,25,CH)
/*
//*
```

It is to be hoped that DASD/storage management personnel now have the ability to easily keep track of the catalog being exported.

Mile Pekic
Systems Programmer (Serbia and Montenegro)

Finding the volume/dataset association

INTRODUCTION

Without a tape management product (TLMS/OS, RMM), it isn't possible to find associated datasets with only the name of a tape volume.

For this reason, I've written the JCL/REXX ICFSCAN, which allows you to examine the contents of any catalog to find the datasets catalogued on a specific volume (DASD or tape).

This procedure is useful in those cases where it is necessary to:

- Obtain a list of any catalogued files on any volume type (online or offline).
- Compare the catalog's entries with the list of the tape's labels.
- Verify the information of your tape management product.

OPERATIONAL ENVIRONMENT

The sample JCL should be self-explanatory – everything can be modified, should one so desire, including the structure of the final printout (see Step/JCL ICETOOL).

The first Step/JCL is an IDCAMS/LISTCat where you should write the catalog name(s) that you wish to examine.

In the Step/JCL REXXBTC, you must enter (in the keyword PARM) the VOLSER for which you intend to make the search.

A return code not equal to zero shows that the catalog(s) has no entries on the specified VOLSER.

SAMPLE JCL TO RUN ICFSCAN

```
//........ JOB  ......................
//** ------------------------------------------------- **
//IDCAMS   EXEC PGM=IDCAMS
//SYSPRINT DD   DUMMY
//IDCOUT   DD   DSN=&&IDCTMP,SPACE=(CYL,(3,1)),UNIT=VIO,
//  DCB=(LRECL=129,BLKSIZE=6250,RECFM=VB),DISP=(MOD,PASS)
//SYSIN    DD   *
  LISTC CAT(CATALOG_NAME)  VOL OFILE(IDCOUT)       <-- UPDATE
  LISTC CAT(............)  VOL OFILE(IDCOUT)       <-- UPDATE
   ..........................................................
/*
//** ------------------------------------------------- **
//IEBGENER EXEC PGM=IEBGENER,COND=(0,NE)
//SYSUT1   DD   DSN=&&IDCTMP,DISP=(OLD,PASS)
//SYSUT2   DD   DSN=&&IEBTMP,SPACE=(CYL,(3,1)),UNIT=VIO,
//  DCB=(LRECL=125,BLKSIZE=6250,RECFM=FB),DISP=(,PASS)
//SYSPRINT DD   DUMMY
```

```
//SYSIN    DD   *
        GENERATE MAXFLDS=1
        RECORD FIELD=(125)
/*
//** ------------------------------------------------------ **
//ICEMAN   EXEC PGM=ICEMAN,COND=(Ø,NE)
//SORTIN   DD   DSN=&&IEBTMP,DISP=(OLD,PASS)
//SORTOUT  DD   DSN=&&ICETMP,SPACE=(CYL,(3,1)),UNIT=VIO,
// DCB=*.IEBGENER.SYSUT2,DISP=(,PASS)
//SYSOUT   DD   DUMMY
//SYSIN    DD   *
  OPTION COPY
    OMIT  COND=(1,124,SS,EQ,C'ACCOUNT-INFO--',OR,                    X
                1,124,SS,EQ,C'BWO STATUS----',OR,                    X
                1,124,SS,EQ,C'BWO-----------',OR,                    X
                1,124,SS,EQ,C'CLUSTER ------',OR,                    X
                1,124,SS,EQ,C'DATASET-OWNER-',OR,                    X
                1,124,SS,EQ,C' HISTORY      ',OR,                    X
                1,124,SS,EQ,C'IDCAMS  SYSTEM',OR,                    X
                1,124,SS,EQ,C'RELEASE-------',OR,                    X
                1,124,SS,EQ,C'THE NUMBER OF ',OR,                    X
                1,124,SS,EQ,C'FUNCTION COMPL',OR,                    X
                1,124,SS,EQ,C' VOLUMES      ',OR,                    X
                24,15,SS,EQ,C' ------')
/*
//** ------------------------------------------------------ **
//ICEGENER EXEC PGM=ICEGENER,COND=(Ø,NE)
//SYSUT1   DD   DATA,DLM=§§
/* REXX ---------------------------- REXX */
/* REXX   ICFSCAN    V1RØ 2ØØ5  Z/OS REXX */
/* REXX ---------------------------- REXX */
"PROF NOPREF"
  PARSE UPPER ARG REXXVOL
   IF REXXVOL = '' THEN EXIT(5)       /* ERROR FOR REXXVOL */
    MSG_STATUS = MSG("OFF")
     TRACE OFF
      CODE = 4                        /* FILE REXOUT EMPTY */

  DO FOREVER
   "EXECIO 1 DISKR REXINP "
     PULL REC_REXINP
      IF RC <> Ø THEN LEAVE

 SELECT
    WHEN   INDEX(REC_REXINP,'LISTING FROM CAT') <> Ø
       THEN   DO
              USRCAT = SUBSTR(REC_REXINP,55,44)
              END
    WHEN   INDEX(REC_REXINP,'VOLSER--') <> Ø
       THEN   DO
```

```
              VOLSER = SUBSTR(REC_REXINP,27,Ø6)
              DEVTYP = SUBSTR(REC_REXINP,51,11)
              END
    OTHERWISE DSNAME = SUBSTR(REC_REXINP,18,44)
 END                                 /* END 'SELECT'      */

 IF VOLSER = REXXVOL
       THEN   DO
              REC_REXOUT = DSNAME VOLSER DEVTYP USRCAT
              PUSH REC_REXOUT
              "EXECIO 1 DISKW REXOUT"
              VOLSER = ''
              CODE = Ø
              END

  END                                /* END 'DO FOREVER'  */

     EXIT CODE
$$
//SYSUT2   DD   DSN=&&PDS(ICFSCAN),SPACE=(TRK,(1,1,1)),
//  DISP=(,PASS),DCB=SYS1.SAMPLIB,UNIT=VIO
//SYSPRINT DD   DUMMY
//SYSIN    DD   DUMMY
//** ------------------------------------------------- **
//REXXBTC  EXEC PGM=IKJEFTØ1,
//  COND=(Ø,NE),PARM='%ICFSCAN VOLSER'            <-- UPDATE
//REXINP   DD   DSN=&&ICETMP,DISP=(OLD,PASS)
//REXOUT   DD   DSN=&&REXTMP,SPACE=(CYL,(3,1)),UNIT=VIO,
// DCB=*.IEBGENER.SYSUT2,DISP=(,PASS)
//SYSTSPRT DD   DUMMY
//SYSPROC  DD   DISP=(OLD,PASS),DSN=&&PDS
//SYSTSIN  DD   DUMMY
//** ------------------------------------------------- **
//ICETOOL  EXEC PGM=ICETOOL,COND=(Ø,NE)
//TOOLMSG  DD   DUMMY
//DFSMSG   DD   DUMMY
//ICEIN    DD   DISP=(OLD,PASS),DSN=&&REXTMP
//REPORT   DD   SYSOUT=*
//TOOLIN   DD   *
   DISPLAY FROM(ICEIN) LIST(REPORT) DATE  TIME PAGE      -
   TITLE(' VOLUME/DATASET ASSOCIATION ')                 -
        HEADER('DATASET NAME')      ON(Ø1,44,CH)         -
        HEADER('VOLSER')            ON(46,Ø6,CH)         -
        HEADER('DEVICE TYPE')       ON(53,11,CH)         -
        HEADER('USERCATALOG NAME')  ON(65,44,CH)         -
        PLUS  LINES(65)
/*
```

PRINTOUT OF THE RESULT

```
GGG/MM/AA       HH:MM:SS        - 1 -        VOLUME/DATASET ASSOCIATION

DATASET NAME                    VOLSER   DEVICE TYPE   USERCATALOG NAME
-----------------------------   ------   ----------   ----------------
CATALOG.ZOS                     VOL001   X'3010200F'   CATALOG.ZOS
CATALOG.ZOS.CATINDEX            VOL001   X'3010200F'   CATALOG.ZOS
ZOS.A.JCL                       VOL001   X'3010200F'   CATALOG.ZOS
ZOS.A.LOAD                      VOL001   X'3010200F'   CATALOG.ZOS
ZOS.A.MESS                      VOL001   X'3010200F'   CATALOG.ZOS
ZOS.A.PANEL                     VOL001   X'3010200F'   CATALOG.ZOS
ZOS.A.VSAM                      VOL001   X'3010200F'   CATALOG.ZOS
ZOS.A.HFS                       VOL001   X'3010200F'   CATALOG.ZOS
ZOS.CICSTS.HTML                 VOL001   X'3010200F'   CATALOG.ZOS
ZOS.CICSTS.LOAD                 VOL001   X'3010200F'   CATALOG.ZOS
ZOS.CICSTS.SOURCE               VOL001   X'3010200F'   CATALOG.ZOS
ZOS.CICSTS.TAB                  VOL001   X'3010200F'   CATALOG.ZOS
.............................   VOL001   X'........'   CATALOG....
```

Massimo Ambrosini
Systems Programmer (Italy)

SQL generator

AIM

The aim of the tool is to create simple DB2 SQL statements (other than joins), which can be used as embedded SQL in COBOL programs written for mainframe servers/applications.

The tool creates the SQL with all the columns listed in the same sequence as they exist in the physical table. The column names are properly aligned and host variables are created to depict the column names correctly. The output SQL statement contains the data type and nullability of each column in the table, to enable the user to supply appropriate values for any of the columns they are interested in. This additional information also enables the user to use the SQL in interactive mode

(foreground execution).

The tool requires three inputs as described below:

- A table name with its qualifier (eg sysibm.systables).
- DB2 subsystem name where the table resides.
- A query keyword (ie select, insert, update, delete, etc).

The tool executes in the foreground after the command is invoked and the output is written to a mainframe dataset.

SQLGEN

```
/*rexx*/
Clear
Say 'Enter the name of the table with qualifier '
Pull tab_nm
dot_loc = Pos('.',strip(tab_nm))
if dot_loc = Ø then
   do
     Say 'Either Table name or the qualifier missed out... Returning'
     Exit 8
   end
else
   do
     qual_nm = substr(strip(tab_nm),1,dot_loc-1)
      tab_nm = strip(substr(strip(tab_nm),dot_loc+1))
   end

Say 'Enter the DB2 subsystem name where this table resides '
Pull sub_sys
upper sub_sys
sub_sys = strip(sub_sys)

Say 'Enter ONE of the below Query Keywords to generate the SQL'
Say '   (ie Select/Insert/Update/Delete/Declare/Fetch)'
Pull kword

upper kword
upper qual_nm
upper tab_nm

kword = Strip(kword)
if kword = 'SELECT' | kword = 'INSERT' |,
   kword = 'UPDATE' | kword = 'DELETE' |,
   kword = 'FETCH'  | kword = 'DECLARE' then
```

```
   nop
else
   do
     Say 'Invalid Keyword entered. Returning'
     Exit 8
   End

ADDRESS TSO "SUBCOM DSNREXX"
IF RC THEN
   DO
      SAY ' '
      S_RC = RXSUBCOM('ADD','DSNREXX','DSNREXX')
   END

ADDRESS DSNREXX "CONNECT "sub_sys
IF SQLCODE ¬= Ø THEN
   CALL ERROR_RTN

   SQLSTMT= ,
    " SELECT NAME,NULLS,KEYSEQ,COLTYPE,LENGTH,SCALE "||,
    " FROM SYSIBM.SYSCOLUMNS    "||,
    " WHERE TBNAME = '"||tab_nm||"'" ||,
    " AND TBCREATOR = '"||qual_nm||"'"||,
    " ORDER BY COLNO                                    "

ADDRESS DSNREXX

"EXECSQL DECLARE C1 CURSOR FOR S1"
IF SQLCODE ¬= Ø THEN
   DO
      SAY 'DECLARE ERROR'
      CALL ERROR_RTN
   END

"EXECSQL PREPARE S1 INTO :OUTSQLDA FROM :SQLSTMT"
IF SQLCODE ¬= Ø THEN
   DO
      SAY 'PREPARE ERROR'
      CALL ERROR_RTN
   END

"EXECSQL OPEN C1"
IF SQLCODE ¬= Ø THEN
   DO
      SAY 'OPEN ERROR'
      CALL ERROR_RTN
   END
j = 1
DO UNTIL(SQLCODE ¬= Ø)
 "EXECSQL FETCH C1 USING DESCRIPTOR :OUTSQLDA"
```

```
  IF SQLCODE = Ø THEN
     DO
       col_nm.j = OUTSQLDA.1.SQLDATA
       var_nm.j = translate(col_nm.j,'-','_')
       nul_id.j = OUTSQLDA.2.SQLDATA
       key.j    = OUTSQLDA.3.SQLDATA
       typ.j    = OUTSQLDA.4.SQLDATA
       len      = OUTSQLDA.5.SQLDATA
       scl      = OUTSQLDA.6.SQLDATA
       if scl   = Ø then
          typ.j = strip(typ.j)||'('||strip(len)||')'
       else
          typ.j = strip(typ.j)||'('||strip(len)||','||strip(scl)||')'
       j = j + 1
     END
  Else
     IF SQLCODE = 1ØØ Then
        do
          If j = 1 Then
             Say 'No Rows Found. Check Table name/Qualifier Name'
        end
     ELSE
        Say 'Fetch Error. SQLCODE 'SQLCODE
END

ADDRESS DSNREXX "EXECSQL COMMIT"

ADDRESS DSNREXX "DISCONNECT"

IF SQLCODE ¬= Ø THEN
   CALL ERROR_RTN
Address TSO
if j > 1 then
   do
     Call create_stmt
     xout = outtrap(sup.)
     "Delete "temp.sqlgenr.out
     xout = outtrap(off)
     "Alloc fi(outfil) da("temp.sqlgenr.out") new catalog",
     " recfm (F B) lrecl(12Ø) space(1Ø 1Ø) tracks"
     "Execio "tmp" diskw outfil (stem out. finis"
     "free ddname(outfil) "
     Say '-------------------------------------------------'
     Say 'Review 'userid()||'.Temp.sqlgenr.OUT for the reqd SQL'
     Say '-------------------------------------------------'
   end

RETURN
/* */
Create_stmt:
```

```
Select
     When kword = 'SELECT'  then
          Call Select_sql
     When kword = 'INSERT'  then
          Call Insert_sql
     When kword = 'UPDATE'  then
          Call Update_sql
     When kword = 'DELETE'  then
          Call Delete_sql
     When kword = 'DECLARE' then
          Call declar_sql
     When kword = 'FETCH'   then
          Call fetch_sql
     Otherwise
          Do
            Say 'Keyword Supplied appears to be corrupted.'
             Exit
          End
END
RETURN

select_sql:

out.1 = 'EXEC SQL'
out.2 = '     '||kword
out.3 = '               '||col_nm.1
tmp = 4
Do i = 2 to (j - 1)
   out.tmp = '             , '||col_nm.i
   tmp = tmp + 1
end
out.tmp = '     '||'FROM'||'   'tab_nm
tmp = tmp + 1
out.tmp = '     '||'INTO'
tmp = tmp + 1
out.tmp = '               '||':HV-'||var_nm.1
tmp = tmp + 1
if nul_id.1 = 'Y' then
  do
     out.tmp = '               '||':HV-'||var_nm.1||'-NUL-IND'
     tmp = tmp + 1
  end
Do i = 2 to (j - 1)
   out.tmp = '             , '||':HV-'||var_nm.i
   tmp = tmp + 1
   if nul_id.i = 'Y' then
     do
        out.tmp = '               '||':HV-'||var_nm.i||'-NUL-IND'
        tmp = tmp + 1
     end
```

```
end
out.tmp = '      '||'WHERE'
tmp = tmp + 1
first_sw = 'Y'

Call where_cond_rtn
out.tmp = 'END-EXEC.'

Return
/*********************************************************************/
insert_sql:

out.1 = 'EXEC SQL'
out.2 = '      '||kword||' INTO '||tab_nm
out.3 = '             (  '
out.4 = '               '||col_nm.1
tmp = 5
Do i = 2 to (j - 1)
   out.tmp = '             , '||col_nm.i
   tmp = tmp + 1
end
out.tmp = '             )  '
tmp = tmp + 1
out.tmp = '      '||'VALUES'
tmp = tmp + 1
out.tmp = '             (  '
tmp = tmp + 1
out.tmp = '               '||':HV-'||var_nm.1
if nul_id.1 = 'Y' then
   out.tmp = Overlay('==N:'||typ.1,strip(out.tmp,t),60)
else
   out.tmp = Overlay('== :'||typ.1,strip(out.tmp,t),60)
tmp = tmp + 1
Do i = 2 to (j - 1)
   out.tmp = '             , '||':HV-'||var_nm.i
   if nul_id.i = 'Y' then
      out.tmp = Overlay('==N:'||typ.i,strip(out.tmp,t),60)
   else
      out.tmp = Overlay('== :'||typ.i,strip(out.tmp,t),60)
   tmp = tmp + 1
end
out.tmp = '             )  '
tmp = tmp + 1
out.tmp = 'END-EXEC.'
Return

/*********************************************************************/

update_sql:
```

```
out.1 = 'EXEC SQL'
out.2 = '     '||kword||' '||tab_nm
out.3 = '     SET'
out.4 = '          '||left(col_nm.1,21)||' = '||':HV-'||var_nm.1
if nul_id.1 = 'Y' then
   out.4 = Overlay('==N:'||typ.1,strip(out.4,t),75)
else
   out.4 = Overlay('== :'||typ.1,strip(out.4,t),75)
tmp = 5
Do i = 2 to (j - 1)
   out.tmp = '        , '||,
                      left(col_nm.i,21)||' = '||':HV-'||var_nm.i
   if nul_id.i = 'Y' then
      out.tmp = Overlay('== :'||typ.i,strip(out.tmp,t),75)
   else
      out.tmp = Overlay('== :'||typ.i,strip(out.tmp,t),75)

   tmp = tmp + 1
end
out.tmp = '     '||'WHERE'
tmp = tmp + 1
first_sw = 'Y'
Call where_cond_rtn

out.tmp = 'END-EXEC.'
Return

/*********************************************************************/
delete_sql:

out.1 = 'EXEC SQL'
out.2 = '     '||kword||' FROM '||tab_nm
out.3 = '     '||'WHERE'
tmp = 4
first_sw = 'Y'
Call where_cond_rtn
out.tmp = 'END-EXEC.'
Return

/*********************************************************************/
declar_sql:

out.1 = 'EXEC SQL'
out.2 = '     '||kword||' <cursor> CURSOR FOR'
out.3 = '     '||'SELECT'
out.4 = '              '||col_nm.1
tmp = 5
Do i = 2 to (j - 1)
   out.tmp = '            , '||col_nm.i
   tmp = tmp + 1
```

```
end
out.tmp = '      '||'FROM'||'   'tab_nm
tmp = tmp + 1
out.tmp = '      '||'WHERE'
tmp = tmp + 1
first_sw = 'Y'
Call where_cond_rtn

out.tmp = 'END-EXEC.'
Return

/*********************************************************************/

fetch_sql:

out.1 = 'EXEC SQL'
out.2 = '      '||kword||' <cursor> '
out.3 = '      '||'INTO'
tmp = 4
out.tmp = '               '||':HV-'||var_nm.1
tmp = tmp + 1
if nul_id.1 = 'Y' then
  do
     out.tmp = '              '||':HV-'||var_nm.1||'-NUL-IND'
     tmp = tmp + 1
  end
Do i = 2 to (j - 1)
   out.tmp = '             , '||':HV-'||var_nm.i
   tmp = tmp + 1
   if nul_id.i = 'Y' then
     do
        out.tmp = '              '||':HV-'||var_nm.i||'-NUL-IND'
        tmp = tmp + 1
     end
end
out.tmp = 'END-EXEC.'
Return

/*********************************************************************/
Where_cond_rtn:
do i = 1 to (j - 1)
      if first_sw = 'Y' then
         do
           out.tmp = '          '||,
                           left(col_nm.i,21)||' = '||':HV-'||var_nm.i
           first_sw = 'N'
         end
      else
           out.tmp = '     AND  '||,
                           left(col_nm.i,21)||' = '||':HV-'||var_nm.i
```

```
      if nul_id.i = 'Y' then
         out.tmp = Overlay('==N:'||typ.i,strip(out.tmp,t),75)
      else
         out.tmp = Overlay('== :'||typ.i,strip(out.tmp,t),75)
      tmp = tmp + 1
end
Return
/*********************************************************************/

ERROR_RTN:

SAY 'SQLCODE  = ' SQLCODE
SAY 'SQLERRP  = ' SQLERRP
SAY 'SQLERRMC = ' SQLERRMC
SAY 'SQLSTATE = ' SQLSTATE
SAY '  '
SAY 'SQLWARN0 = ' SQLWARN.0
SAY 'SQLWARN1 = ' SQLWARN.1
SAY 'SQLWARN2 = ' SQLWARN.2
SAY 'SQLWARN3 = ' SQLWARN.3
SAY 'SQLWARN4 = ' SQLWARN.4
SAY 'SQLWARN5 = ' SQLWARN.5
SAY 'SQLWARN6 = ' SQLWARN.6
SAY 'SQLWARN7 = ' SQLWARN.7
SAY 'SQLWARN8 = ' SQLWARN.8
SAY 'SQLWARN9 = ' SQLWARN.9
SAY 'SQLWARN10 = ' SQLWARN.10
SAY '    '
SAY 'SQLERRD1 = ' SQLERRD.1
SAY 'SQLERRD2 = ' SQLERRD.2
SAY 'SQLERRD3 = ' SQLERRD.3
SAY 'SQLERRD4 = ' SQLERRD.4
SAY 'SQLERRD5 = ' SQLERRD.5
SAY 'SQLERRD6 = ' SQLERRD.6

EXIT(16)
```

COMMAND INVOCATION AND EXECUTION

Create a member named SQLGEN and save the source code in it. Make sure that the PDS is allocated to the user's profile while he/she is logged in.

Like most other REXX tools, the command should be invoked by issuing the following at the command prompt:

```
TSO SQLGEN
```

If the library/PDS is not allocated to the user's session, the above will not work. Alternatively he/she can open the PDS in option 3.4 (DSLIST panel) and type 'EX' against the member name 'SQLGEN'.

Upon successful invocation of the above command, the SQLGEN REXX EXEC will execute and prompt the user for the following (users are expected to supply the appropriate values depending on their system configuration and interest):

```
 Enter the name of the table with qualifier
SYSIBM.SYSSTRINGS                  /* user's response*/
 Enter the DB2 subsystem name where this table resides
MYDB                               /*user's response*/
 Enter ONE of the below Query Keywords to generate the SQL
    (ie Select/Insert/Update/Delete/Declare/Fetch)
SELECT                             /* user's response*/
```

After the tool successfully finishes, it will complete as shown below. This indicates where the output is written to.

```
------------------------------------------------
 Review <User-ID/tso-Prefix>.TEMP.SQLGENR.OUT for the reqd SQL
 ------------------------------------------------
 ***
```

ANALYSIS

Now open the dataset and view the output. The output for the above set of input parameters will be:

```
EXEC SQL
     SELECT
               INCCSID
             , OUTCCSID
             , TRANSTYPE
             , ERRORBYTE
             , SUBBYTE
             , TRANSPROC
             , IBMREQD
             , TRANSTAB
     FROM    SYSSTRINGS
     INTO
               :HV-INCCSID
             , :HV-OUTCCSID
             , :HV-TRANSTYPE
             , :HV-ERRORBYTE
```

```
              :HV-ERRORBYTE-NUL-IND
            , :HV-SUBBYTE
              :HV-SUBBYTE-NUL-IND
            , :HV-TRANSPROC
            , :HV-IBMREQD
            , :HV-TRANSTAB
     WHERE
          INCCSID              = :HV-INCCSID       == :INTEGER(4)
     AND  OUTCCSID             = :HV-OUTCCSID      == :INTEGER(4)
     AND  TRANSTYPE            = :HV-TRANSTYPE     == :CHAR(2)
     AND  ERRORBYTE            = :HV-ERRORBYTE     ==N:CHAR(1)
     AND  SUBBYTE              = :HV-SUBBYTE       ==N:CHAR(1)
     AND  TRANSPROC            = :HV-TRANSPROC     == :CHAR(8)
     AND  IBMREQD              = :HV-IBMREQD       == :CHAR(1)
     AND  TRANSTAB             = :HV-TRANSTAB      == :VARCHAR(256)
END-EXEC.
```

If we take a closer look at the SQL, we will be able to see that the NULL indicator variables are handled in the SQL for all nullable columns in the table.

In the WHERE clause, an SQL comment is given to show whether the column can have null values and the data type of each of columns.

For example, the condition:

```
ERRORBYTE            = :HV-ERRORBYTE     ==N:CHAR(1)
```

should be interpreted as described below:

- ERRORBYTE is the column in the table.
- :HV-ERRORBYTE is the name of the host variable for the column ERRORBYTE.
- ==N:CHAR(1) shows that the column can accept nulls and it is defined to accept one character data for each row in the table.

If the column is not nullable, it will look like this:

```
ERRORBYTE            = :HV-ERRORBYTE     == :CHAR(1)
```

Try working with different table names and SQL keywords and analyse the results.

ERRORS

Below are examples of error messages, when the program finds a SQL error:

```
PREPARE ERROR
SQLCODE  =  -805
SQLERRP  =  DSNXEPM
SQLERRMC =  MYDB..DSNREXX.1831FABB05B50409:DSNREXX:03
SQLSTATE =  51002

SQLWARN0 =
SQLWARN1 =
SQLWARN2 =
SQLWARN3 =
SQLWARN4 =
SQLWARN5 =
SQLWARN6 =
SQLWARN7 =
SQLWARN8 =
SQLWARN9 =
SQLWARN10 =

SQLERRD1 =  -250
SQLERRD2 =  0
SQLERRD3 =  0
SQLERRD4 =  -1
SQLERRD5 =  0
SQLERRD6 =  0
```

ASSUMPTIONS AND LIMITATIONS

It is assumed that:

- The system supports REXX and DB2 connectivity.
- DB2 is up and running in the current executing environment/ system.
- The WHERE clause is always constructed with an ‘AND’ relational operator.
- All comparison operators are assumed to use the EQUAL operator (=).
- Only basic DML SQL statements (like SELECT, INSERT, UPADATE AND DELETE) and embedded SQL (like DECLARE, FETCH) are supported.

- Changes will be required in the output SQL to use it in interactive mode.
- Editing is needed after the SQL is copied in COBOL programs.

Suresh Kumar Murugesan
Systems Analyst
Cognizant Technology Solutions (USA)

Quick JCL dataset edit

PROGRAM SCOPE

The program will execute the editing of a dataset inside the JCL job stream using the functions keys and the cursor as a dataset pointer.

If the cursor is not at the beginning of a dataset or if the dataset does not exist, an error message will be shown.

PROGRAM SOURCE

I have called the REXX program EDSN, but you can modify the name to follow your standards.

```
/* REXX */
/*
Setup: modify for instance the F23 (Shift+F11) key with EDSN
Usage: edit a member, place the cursor to the beginning of the dsname
        and press (Shift+F11)
*/
nomsg = msg(off)
ROW = 0
COL = 0
LINEDATA = " "
ADDRESS ISPEXEC
"ISREDIT MACRO"                          /* macro declaration       */
```

```
"ISREDIT (ROW,COL) = CURSOR"           /* get cursor position      */
"ISREDIT (LINEDATA) = LINE "ROW        /* get line from cursor pos */
do i = col to 8Ø
 strg = substr(LINEDATA,i,1)    /* looking for delimiter after dsn */
 if strg = " " ! ,
     strg ="," then
                  do
                   dsn=substr(LINEDATA,col,i-col)
                   ckdsn = SYSDSN("'"dsn"'")
                   cp = COPIES('*',length(dsn)+18+2) /*format err msg */
                   if pos('.',dsn) = Ø then  /*check for valid dataset*/
                       do
                        say " ";say " ";say cp
                        say " "dsn": Is not a dataset"
                        say cp
                        exit
                       end
                   if ckdsn ¬= "OK" then     /*dataset exist ??       */
                       do
                        say " ";say " ";say cp
                        say " Dataset: "dsn" not found"
                        say cp
                        exit
                       end
                   "ISPEXEC EDIT DATASET("dsn")"  /*edit dataset      */
                   leave
                  end
end
EXIT
Installation
```

PROGRAM INSTALLATION

Copy the EDSN REXX program to the user SYSPROC library.

You can get the SYSPROC library list for editing any dataset by typing the following command from the TSO command line:

```
tso isrddn
```

Press the F8 key until you find the DDname SYSPROC.

On the right side of the screen there is a list of the dataset names allocated to the DDname SYSPROC.

Copy the EDSN REXX program into one of these libraries (for instance *User.lib.exec*).

ENVIRONMENT SET-UP

Now it is necessary to associate the REXX program with a function key.

In order to do this, follow these steps:

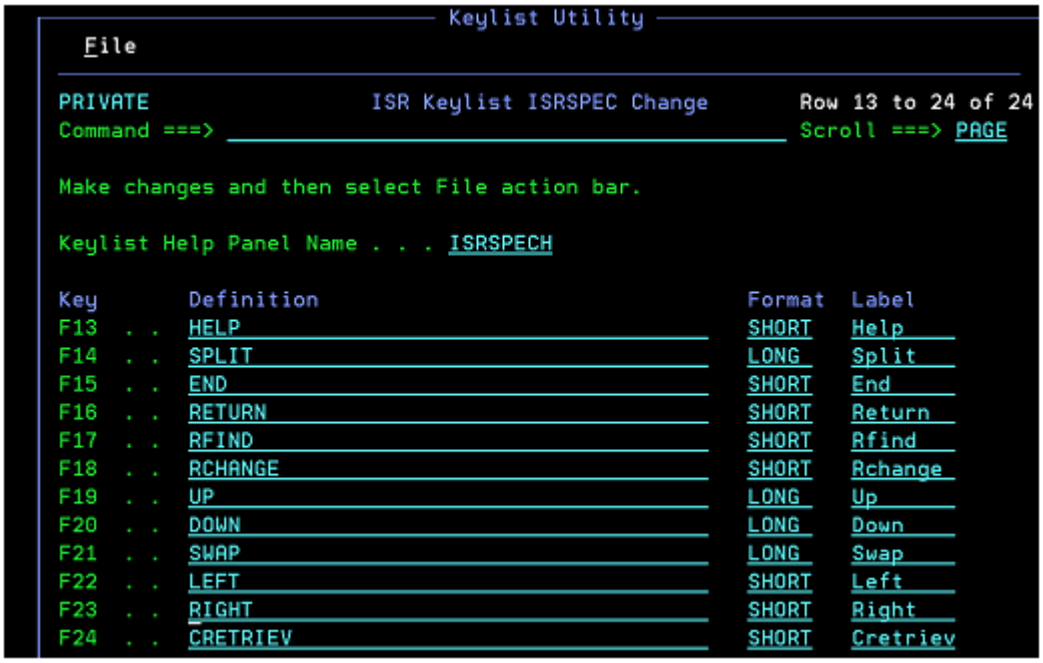

Figure 1: PF key allocation

- Edit any dataset.
- Type KEYS on the command line.
- Press the F8 function key and Figure 1 will be shown:
- For instance to associate the EDSN REXX program with the F23 key, change RIGHT under the *Definition* column to EDSN and press the *Enter* key.
- Now press the F3 key and exit from edit mode by typing Cancel on the command line.

Figure 2: Editing a file from JCL

USAGE

Program EDSN is now associated with the F23 key.

To edit a JCL job stream – see Figure 2.

Put the cursor at the beginning of the dataset, hold Shift and press the F11 key (F23 key); the dataset will be shown in edit mode.

If the cursor is not at the beginning of a dataset, an error message will be shown. If the dataset does not exist, an error message will be shown.

Magni Mauro
Systems Engineer (Italy)

PDSE future direction

IBM has announced a number of future enhancements to PDSE. Some of these may change, but in this article I present

some detail of what has been published as a statement of direction at the moment.

There will be a re-startable PDSE address space enhancement that will implement design changes to further enhance reliability, availability, and serviceability for PDSE processing. The key change in this area is the implementation of a second system address space. This will be named SMSPDSE1. It will be optional and will work alongside the SMSPDSE system address space. This optional address space, SMSPDSE1, will be a re-startable address space. This means that an IPL will no longer be required to restart PDSE processing after a critical error. This enhancement will include new commands to effect termination, activation, and restart. The old ANALYSIS and FREELATCH commands remain, but they can also be used against the new address space when it is utilized and active. Through the implementation of the optional second address space, the need to IPL to recover from a PDSE critical error is eliminated. To implement this change you will need to apply maintenance to all members of a sysplex. APAR OA05075 is related to the fixes that will be required.

SYS1.PARMLIB changes will be required to activate the second system address space, SMSPDSE1. The IGDSMS*xx* member active at IPL must have both PDSESHARING(EXTENDED) and PDSE_RESTARTABLE_AS(YES) specified. The defaults for these parameters are NORMAL and NO, and if either of these defaults values are specified or implied, then only the non-restartable address space, SMSPDSE, will be initiated and activated.

The existing IGDSMS*xx* parameters will be enhanced with synonym parameters for the global PDSE address space (original SMSPDSE address space):

- LRUCYCLES synonym PDSE_LRUCYCLES
- LRUTIME synonym PDSE_LRUTIME
- HSP_SIZE synonym PDSE_HSP_SIZE

- BMFTIME synonym PDSE_BMFTIME.

You do not have to use these synonyms and they are provided only for clarity.

The following IGDSMS*xx* PARMLIB parameters are in support of this function:

- PDSE_RESTARTABLE_AS(YES | NO)
- PDSE1_MONITOR(NO | YES [, interval |60[, duration |15]])

  This specifies whether the SMSPDSE1 address space will be monitored. The default is YES, and optionally *interval* and *duration* values can be specified in seconds. The *interval* and *duration* operands can only be specified with the YES option. *Interval* is the number of seconds between successive scans of the monitor. The default is 60 seconds. *Duration* is the number of seconds an exception must exist before it is treated like an error. The default is 15 seconds.

- PDSE1_LRUCYCLES (nnn | 240)

  This parameter specifies the maximum number of times the BMF LRU routine will allow unused buffers to remain unavailable for reuse. The allowable range is from 5 to 240. BMF will dynamically adjust the actual number of times it allows before inactive buffers are reused. PDSE1_LRUCYCLES is related to PDSE1_LRUTIME. A change to PDSE1_LRUCYCLES will take effect the next time the LRU routine runs.

  Most installations should use the default value of 240. This value can be tuned in the case of very high data rates. SMF type 42 subtype 1 records should be monitored to see the cacheing activity in the BMF dataspace.

- PDSE1_HSP_SIZE (nnn | 256)

  This specifies the size of the hiperspace restartable

SMSPDSE1 address space member cacheing. The unit of measurement is in megabytes and defaults to 256MB. However, if the value specified or defaulted to is greater than half of the expanded storage available, then the hiperspace size will be limited to half of the available expanded storage. On z/Series systems where there is no longer expanded storage, the default will be 256MB. However, if the value specified or defaulted to is greater than a quarter of the central storage available, then the hiperspace size will be limited to a quarter of the available central storage. Note that if the amount of central storage is 64MB or less, the amount of storage used is limited to one eighth of that available. In this instance, however, the system may not complete its IPL. On systems with no expanded storage online and not running in ESAME mode (64-bit addressability), the hiperspace will not be created.

If there are not enough storage resources available to satisfy both the PDSE_HSP_SIZE and PDSE1_HSP_SIZE values, the system will use some portion, up to the full amount. PDSE1 member cacheing will cease if the storage resources in use for the hiperspaces (expanded or central) become fully utilized in the system.

This parameter's value must be specified with care because if it is coded too low degradation of non-global PDSE performance will occur, and if coded too high you could seriously impact whole system-wide performance.

The **DISPLAY SMS,OPTIONS** command can be used to see the current value in use for PDSE1_HSP_SIZE, and its effectiveness may be evaluated by examining SMF type 42 subtype 1 records.

- PDSE1_BMFTIME (nnn | 3600)

  This parameter specifies the interval in seconds between when statistics are recorded and what SMF type 42 records are written for Buffer Management Facility cache used for the SMSPDSE1 address space. You can specify

a value from 1 to 86,399 seconds. If the SMF_TIME keyword is set to YES, it overrides this keyword. The default is 3600 seconds.

Once all systems have the required maintenance applied, an IPL will invoke the change and allow the SMSPDSE1 address space to be started. There will be two system address spaces for PDSEs at that point, SMSPDSE and SMSPDSE1. The SMSPDSE address space will remain as a non-restartable address space for global connections. These connections cannot tolerate interruption and these connections include those made for PDSEs in the LINKLIST. The new SMSPDSE1 address space is restartable. It will provide connections to PDSEs that do not have global connections. These are PDSEs that are not connected through the SMSPDSE address space. While the SMSPDSE1 address space is restartable, it should be restarted only when absolutely necessary. This would be if you had issues with PDSEs under its control. The reason that this is important is because each time the address space is stopped or terminated, a small amount of ECSA is lost. This small amount can be up to a few megabytes. In addition to this, an ASID will be lost too, which is typical for address spaces that provide cross memory services. So while one or two restarts, or maybe a few, will/should not be an issue and will not affect the system resource pool, overall there are limitations that need to be considered because system performance could be impacted in the event of many restarts. Before terminating or restarting the SMSPDSE1 address space you will need to determine what effect the restart may have. For example, a restart could mean that job failures will occur, TSO users might have issues and some connections may not restart. Not all connections are restartable either. Restart might not work correctly if more than one SMSPDSE1 address space is restarted concurrently within the sysplex. The restarting systems will not be of benefit to other sysplex members for the time in seconds specified by the XQUIESCE value. If a PDSE member is deleted while SMSPDSE1 is down, a reconnect to the member will fail.

When a PDSE member is deleted while SMSPDSE1 is down and the quiesce fails or SMSPDSE1 was forced down, a reconnect to the member will fail. Unless standard PDSE interfaces are used, PDSE processing will not be aware of a connection to a PDSE. On another sysplex, if a user opens a PDSE for update when the restartable connection was for a read/write, the restart connection will fail. Dump and restore jobs for PDSEs will fail.

Also, if the SMSPDSE1 address space ended because of an error or the FORCE command, the results of the restart are as expected in such a situation – unpredictable. If all systems in the PDSE extended sharing sysplex do not have PDSE restartable address space support installed, the results of the restart are unpredictable. This latter restriction, however, can be overcome by ensuring that all systems in the sysplex have PDSE restartable address space support installed, either in the form of base code or through a toleration PTF.

There will be some changes to the **DISPLAY SMS,OPTIONS** command output to show what options were either specified or defaulted for the PDSE restartable address space.

There are also new commands that can be used to activate and restart the SMSPDSE1 address space. These are:

- VARY SMS,PDSE1,ACTIVATE[,COMMONPOOLS(NEW | REUSE)]
- VARY SMS,PDSE1,RESTART [,QUIESCE(n|3) [,COMMONPOOLS (NEW | REUSE)]]
- VARY SMS,PDSE[1],ANALYSIS [,DSNAME(dsname) [,VOLSER(volser)]] [,RETRIES(n | 1500)]
- VARY SMS,PDSE[1],FREELATCH(latchaddr,asid,tcbaddr) [,RETRIES(n|1500)]
- VARY SMS,PDSE[1],MONITOR [,ON | OFF | RESTART] [,interval,duration]

None of these commands will work using the PDSE1 option unless the restartable address space, SMSPDSE1, is started at IPL time.

The ACTIVATE command will start up the SMSPDSE1 address space again after a termination. The RESTART command will recycle the SMSPDSE1 address space. When problems are indicated with the address space SMSPDSE1, this command should be used to restart it. The QUIESCE operand specifies an interval in seconds. This interval is the maximum time for which the termination of the address space will be delayed so that operations in progress may finish. New requests are held for this duration. The operand applies to systems on which the SMSPDSE1 address space is running and will default to three seconds. The XQUIESCE operand is similar to the QUIESCE operand, but it applies to other systems within the sysplex. During the interval specified in seconds, certain new PDSE operations are prevented while the SMSPDSE1 address space will terminate. If the SMSPDSE1 address space restarts and reconnects within that interval, there should be no failures for any restartable connections caused by deletion of members or PDSEs opened for updates on other systems. This defaults to five seconds.

Both the ANALYSIS and FREELATCH commands are still available, and can now both be used against the new address space when active. The syntax of these has been changed and the new format is shown below:

- VARY SMS,PDSE[1],ANALYSIS [,DSNAME(*dsname*) [,VOLSER(*volser*)]] [,RETRIES(n | 1500)]
- V A R Y SMS,PDSE[1],FREELATCH(*latchaddr,asid,tcbaddr*) [,RETRIES(n|1500)]

The ASID and TCB address will be displayed in the output of an ANALYSIS command if any contention exists. It is operationally best to always issue an ANALYSIS before a FREELATCH command is issued.

There is also a new command for monitoring purposes:

```
VARY SMS,PDSE[1],MONITOR [,ON [,interval,duration] | OFF | RESTART]
```

This command can be used to start monitoring in either of the PDSE address spaces. The interval and duration operands can be specified only with the ON option. *Interval* is the number of seconds between successive scans of the monitor. It defaults to the interval specified in the SMS initialization parameters, and if it is not specified in that member then it defaults to 60 seconds. *Duration* is the number of seconds an exception must exist for before it is treated like an error. Again, it defaults to the duration specified in the SMS initialization parameters, and if it is not specified in that member the default is 15 seconds.

John Bradley
Systems Programmer
Meerkat Computer Services (UK)

Why not share your expertise and earn money at the same time? *MVS Update* is looking for REXX EXECs, program code, JavaScript, etc, that experienced users of MVS have written to make their life, or the lives of other users, easier. We are also looking for explanatory articles, and hints and tips, from experienced users. Articles can be of any length and should be e-mailed to the editor, Trevor Eddolls, at trevore@xephon.com.

A glimpse at the WLM's predictions

INTRODUCTION

As is commonly known, the Workload Manager (WLM) was first introduced with MVS/ESA 5.1. Since then it has evolved to become one of the most integral parts of the z/OS operating system and of IBM's overall strategy for self-managed, self-healed, and self-tuned systems. Now that most MVS (aka OS/390, z/OS) installations have migrated to WLM goal mode, they are looking into exploiting goal mode features and re-visiting how they set up their service policies, and it was noticed that while many installations are very happy with the implementation of their service definition, some also experienced behaviour different from what they had expected. If one tries to find out the reason for this behaviour it is most likely that an 'it depends' cliché answer will be provided by a performance analyst. Before revisiting the settings (which can be done in order to pre-empt possible WLM goal mode problems) and attempting to re-evaluate them (which is usually done when goals are being missed or workloads are not performing as well as expected – ie it is done to fine tune goals or to resolve problems), be well aware that an important first step towards evaluating your WLM service definition and goals should be to gain an understanding of how WLM works.

There are several IBM manuals that should be studied and consulted if one is keen to understand how Workload Manager works. Of prime importance is a knowledge of Workload Manager algorithm basics; performance index; delays and using indicators, receivers, donors, and resources; receiver value; net value; plots and histories; policy adjustment; and resource adjustment flow control. You also need to understand the basics of WLM algorithms to solve MPL delays, CPU delays, storage delays, I/O delays, etc.

The z/OS WLM in goal mode is a level of management above

SRM as we have known it, and is the component that dynamically allocates or redistributes server resources such as CPU, I/O, and memory across a set of workloads based on user-defined goals and their resource demand within a z/OS image. It dynamically adjusts and directs the settings of the following attributes of a dispatchable unit: dispatching priority, dynamic storage protection, multiprogramming level for swappable work, residency for individual address spaces dispatchability, frequency and duration of dispatching (time slicing/throttling), I/O priority, number of initiators serving job class, and number of dynamic alias UCBs for parallel access volumes. Some new capabilities are added, such as rapid adjustment of dispatching priority to slow down address spaces in service classes that exceed their resource group service rate cap. Also new is the propagation of I/O priority to the device level with the Enterprise Storage Server.

As we already know, every 10 seconds WLM makes an adjustment to address a component of one kind of delay (CPU, storage, etc) in one service class period that is missing its goal. Using the samples of delays, WLM calculates what resource might be able to help that receiver. When the resource is identified, WLM looks for other work that can donate the resource. By analysing the sampled state information and plots and historical data for both the donor and receiver, WLM determines whether the donation is a good idea or not. Making only one change every 10 seconds allows WLM to observe the effects of the change, providing clear feedback to the algorithms. This also allows WLM to reverse a previous change if the intended outcome was not achieved. WLM tries to assist work in the order of its stated (customer policy) importance – from a sysplex perspective first, then from an individual system's perspective.

To summarize, the main features of WLM's philosophy can be seen as this: dynamic workload characterization, which is based on state sampling and measurements; prediction for resource adjustments (rules of thumb are not used since

actual measured results such as plots and histories are being used); receiver–donor loop to adjust resources is run every 10 seconds, and, finally, WLM does the work sufficient for adjustment to make an improvement pertaining to just a single problem at a time since the optimal change at any given point is not required.

Similar to a standard MVS performance monitor, which frequently uses state sampling as a way of determining where work spends its time, the WLM/SRM too samples the current state of every dispatchable unit every 0.25 seconds to learn about delays, and then it tries to use one of its controls to minimize the delay, without compromising more important work. These samples are accumulated for each address space, and then further accumulated into the service class period associated with the address space. The states that WLM cares about are those states that reflect usage of, or delay for, a resource that SRM can allocate, such as using the CPU, waiting for access to the CPU, waiting for a page fault (separate states are sampled, depending on the type of page fault), and waiting for a swap-in to be started/to complete. The collection of state samples is used in several ways. For example, WLM/SRM uses the sampling to determine where work is spending its time. This information allows WLM to determine what resource is the primary bottleneck for the work and allows WLM/SRM to assess the impact of some possible action it might take. WLM is smart enough not to make any drastic tuning decisions based on just a single sample. So WLM keeps enough recent history to have a comprehensive picture of various delays. In addition to gathering data via sampling, WLM maintains information on the response time achieved by work completing in each service class period. Again, WLM needs to avoid making a drastic tuning decision based on just a few completions when managing towards a response time goal. Therefore, WLM maintains some history on the response times for recently completed transactions. When there are a significant number of completed transactions, WLM can use much more recent data for making its tuning decisions. In addition, WLM projects/

predicts the expected response time of work currently 'in-flight'. Comparing the actual or projected/predicted accomplishments for some piece of work with its goal is very straightforward. Frequently, WLM looks at every service class period to compare the actual value with the goal.

In addition to just knowing whether a service class is meeting its goal or not, WLM must assemble some 'insight' before deciding to change resource allocations. This is accomplished with precisely the same tool used by a mathematician or an analyst studying trends – a plot showing relationships between an independent variable, usually on the 'x' axis, and a dependent variable, usually on the 'y' axis. WLM uses resource plots to project a recipient benefit and a donor cost. The net benefit must be positive before a resource adjustment will take place. This idea of plotting data was introduced in MVS/ESA Version 4 Release 2 with the Working Set Management (WSM) component, plotting the address space paging characteristic curve and the system paging curve. In order to derive the net value of the receiver/donor relationship, WLM creates the following plots to track how well work is being processed:

- System paging delay plots
- Period MPL delay plot
- Period ready user average plot
- Period swap delay plot
- Period paging rate plot
- Period proportional aggregate speed plot
- Address space paging plot
- I/O delay plot
- Queue delay plot
- I/O velocity plot.

All plots except the address space paging plot are 'one-curved' plots, where one variable is plotted against another. The address space paging plot is a 'three curved' plot where one variable is plotted against three variables.

UNDERSTANDING THE WLM PLOTS

Before proceeding any further let us remember that the plots WLM makes are used by various WLM components/functions: some are used by policy adjustment function, most of them are being used by resource adjustment functions, while a few plots are being used by various fix routines.

System paging delay plot

The sampling of storage-related conditions is done every 250 milliseconds. After two seconds, resource management collects data about what happened over the last interval and then plots the points. It may happen that the system AUX paging rate is higher than the DASD paging subsystem can handle. The first sign that the paging subsystem is overloaded is that paging-in from AUX starts taking longer to process because of the queue time building up. When the paging subsystem becomes overloaded, this plot will show AUX delay samples growing significantly faster than the system's AUX page-in rate. The first point at which the rate of growth of AUX delay samples becomes more than twice the AUX page-in growth rate is called the 'shed work point'. The last point at which the rate growth in AUX delay samples is close to proportional with the AUX page-in growth rate is called the 'add work point'. On the other hand, if the DASD paging subsystem is performing well, then the plot will show that AUX delay samples are growing proportionately with the system's AUX page-in rate. WLM is using this plot to determine whether the paging configuration is close to capacity or if it can support additional work, and it can show the point at which additional page faults will start requiring a disproportionately longer time because the paging subsystem is becoming overloaded. There is one system paging plot per system. The plot has the

following attributes:

- x axis – the system-wide page fault rate, in page faults per second.
- y axis – the system-wide auxiliary storage delay sample rate, in samples per minute.
- Page faults and delay samples are for all types – private area, common area, and cross memory.

Period MPL delay plot

The period MPL delay plot shows how response times may be improved by increasing MPL slots or how response time may be degraded by reducing MPL slots. In fact this plot is designed to predict the number of ready users when given a particular MPL in_target for a period. Also it can be used to avoid lowering a period's MPL in_target to a point that causes out-and-ready-users to experience large increases.

WLM uses the period MPL delay plot to assess increasing or

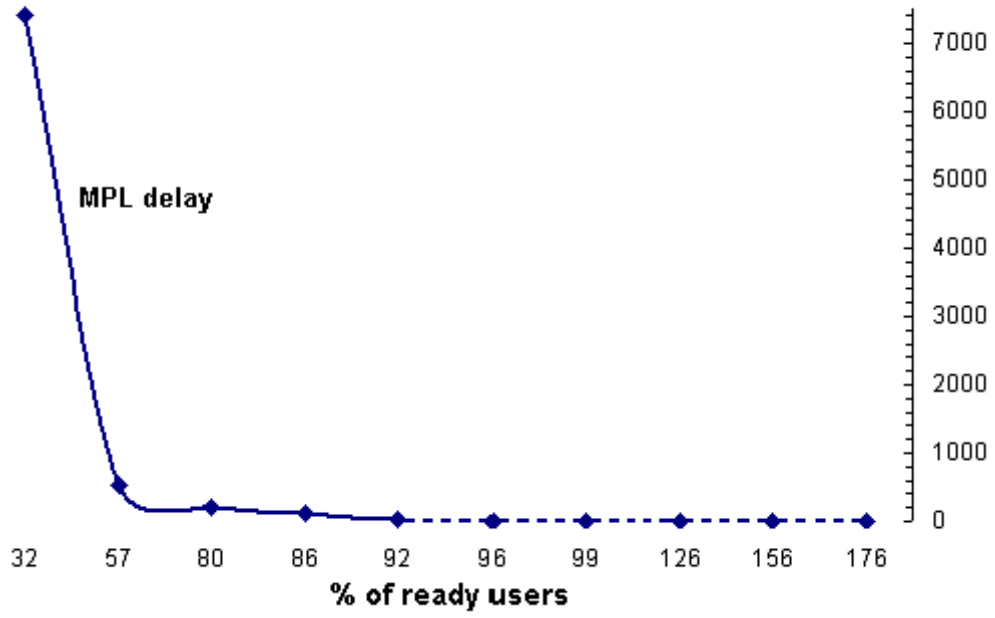

Figure 1: MPL delay plot for TSO service class/first period

decreasing a service class period's MPL targets. Let's remember that the MPL of a period is the number of address spaces in that period that are swapped-in in central storage at any given time. An address space in a period experiences MPL delays when that address space is swapped-out and is ready to be swapped in, but WLM/SRM has not (or cannot) increase the MPL in the period to allow the address space to be swapped in. In order to alleviate a period's MPL delay, WLM invokes the MPL_delay_fix routine, which must allow the period's 'swapped-out and ready' users into central storage. It is done by increasing the period's MPL in_target by 1 and perhaps the period's MPL out_target by 1 as well. On the other hand, WLM must 'find' and 'ensure' that there is enough central storage available to contain another address space typical of that period. This is done through the invocation of the find_storage routine.

A sharp 'knee' is expected in the curve of this plot. The idea here is that if a period were given enough MPL, the number of ready users in that period would remain fairly constant. The 'knee' of the curve indicates the point of MPL sensitivity. That is, the point at which if not enough MPL slots are allocated to the period, the number of out-and-ready users starts to increase. The sharpness of the 'knee' in this plot becomes especially apparent when the ready user average plot is associated with a period for TSO short transactions, which are very sensitive to MPL. See Figure 1 for an example of the plot used to predict the effect of an MPL change.

The plot has the following attributes:

- x axis is the percentage of ready users (RUA) who have an MPL slot available to them.

  This is the ratio of MPL available to the period and the number of ready users in the period. Another way to look at this number is the ratio of how much MPL has been given to how much is needed:

  X = max(in_ target, Current MPL)/max(RUA, long term RUA)

MPL *in_target* is the number of MPL slots representing the number of address spaces in a period that must be swapped-in (ie, in central storage) in order for the period to meet its goals.

Current MPL is the current number of address spaces in a period that are swapped-in-and-ready.

Ready User Average (*RUA*) is the average number of users in the period that are ready to run (not in long wait, not in detected wait, not in terminal wait). These address spaces are either out-and-ready or in-and-ready. It is typical of this number to fluctuate greatly between policy adjustment intervals.

Long term RUA is the long-term average number of users in the period that are ready to run. These address spaces are either out-and-ready or in-and-ready.

- y axis – the MPL delay per completion in milliseconds. This is a measure of time from when a user was out-and-ready.

Period ready user average plot

Closely related to MPL delay plot is the period ready user average plot that SRM uses to predict the number of ready users when assessing an MPL target change. The plot can show the point at which users will start backing up. It also shows the approximate MPL target at which users would be disproportionately delayed because of MPL. Also it can be used to avoid lowering a period's MPL in_target to a point that causes out-and-ready-users to experience large increases.

The plot has the following attributes:

- x axis – the number of MPL slots (times 16) available to the service class period, or the larger of the MPL in_target and the current MPL.
- y axis – the maximum number of ready users, times 16, averaged over a two-second interval.

Period swap delay plot

WLM uses the period swap delay plot to project the change (increasing or decreasing) in the swap delay the service class period will experience if its swap project time is modified. The projected/predicted change in swap delay is used to calculate the change in the swap delay samples that the service class period will experience in the next interval. From these samples the response time or velocity deltas and performance index delta are calculated for the service class period. The performance index delta is then used to determine whether the trade-off being considered is a good one.

A service class period swap delay plot is built by swap_delay_fix routine in order to derive the new value of swap protect time. The routine is invoked if a receiver service class period is experiencing its largest delay because the address spaces within the period are being delayed because of page-in activities from AUX caused by a swap-in process.

The plot shows how response time may be improved or degraded by increasing or decreasing a service class period's swap protect time. The plot decreases, that is the longer that address spaces are allowed to stay swapped-in in processor storage, the smaller the amount of swap delay they have.

The plot has the following attributes:

- x axis – the average time an address space in the service class period is logically swapped or swapped on expanded, in milliseconds.
- y axis – the swap delay time per completion, in milliseconds.

Period paging rate plot

WLM uses period paging rate plot to project the change to the period's AUX page-in rate caused by modifying the period's protective processor storage target (storage isolation). The projected page-in rate change is used to calculate the change in the AUX delay samples that the service class period will experience in the next interval. From these samples the

response time or velocity deltas and performance index delta are calculated for the service class period. The performance index delta is used to determine whether the trade-off being considered is a good one.

For example, for a trivial TSO service class period, it is assumed that most of the paging delays are caused by the trim at swap out (logically or physically). Also, in this type of service class period the transactions do not stay around long enough for SRM to gather sufficient data to manage the working sets of the address spaces individually. Therefore, decisions to change the storage protective processor storage of these address spaces are made on a service class period-wide basis. Every address space in the service class period has the same processor storage protective target.

The plot has the following attributes:

- x axis – the average processor storage size (in frames) of address spaces that left the service class period in the last 10 seconds.

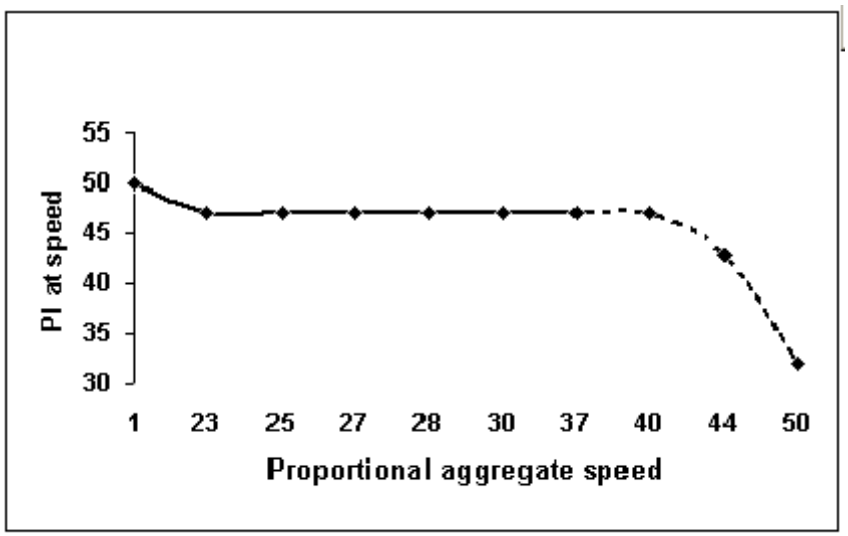

Figure 2: Period proportional aggregate speed plot for CICS service class/first period

- y axis – the page fault rate in tenths of a page fault per departure from the period.

Period proportional aggregate speed plot

WLM uses the proportional aggregate speed plot to assess the effects of processor access or storage allocation changes for served service classes.

HOW WLM DOES IT

When, during policy adjustment looping, WLM finds there is a receiver service class period that is experiencing its largest delay because dispatchable units in this period are waiting for CPU resources, it invokes the processor delay fix routine. The processor delays are addressed by increasing the dispatching priority of the receiver, or decreasing the dispatching priority of one or more donors, or both. Projections are being made for the time the affected service class periods will consume and the new wait-to-using ratios the affected service class periods will experience after the dispatching priority changes. The time is projected based on the total processor time available, the maximum demand of the service class period, and the maximum demand of work at higher and equal dispatch priorities. The wait-to-using ratios are projected using the actual observed data collected for the wait-to-using ratio at each priority, adjusted for the maximum demand moved from one priority to another. In the next step, projected times and wait-to-using ratios are used to calculate the change in processor_using and delay samples that work in the service class period will experience in the next interval.

For served classes, the samples are calculated for all servers of receivers or donors and are then apportioned to the served classes based on the relative number of observations of each server serving each class. A proportional aggregate speed is then calculated for each served receiver or donor and the performance index delta is read off the proportional aggregate speed plot (see Figure 2). For non-served classes, these

samples are used to calculate response time or velocity deltas and performance index deltas for the receiver and donors. In both the served and non-served cases, the performance index deltas are then evaluated to determine whether the trade-off is a good one. If there is a net value to the dispatching priority trade-off, the change is made.

The plot has the following attributes:

- x axis – the proportional aggregate speed of a service class period.
- y axis – the performance index at that speed, times 100.

Address space paging plots

The address space paging plot is used to project the change in the auxiliary and expanded page-in rate for the address space caused by modifying the address space's protective processor storage target. In other words, WLM uses address space paging plots when assessing whether to increase or decrease the central storage or processor storage allocated to an address space.

When a period has a goal (other than a short response time goal which uses period paging rate plot) and experiences private area paging delays, a different approach is used. There are two reasons for this difference:

- Transactions in such a period should stay around long enough for WLM to learn how the transaction is using processor storage.
- Transactions in such a period are probably going to use processor storage very differently, so that a 'one size fits all' approach to solving paging delays may not work. Instead the individual address space's working set is managed to reduce paging delays.

In this case the action taken to address AUX paging delays is to increase the individual protective processor storage target of one address space in the service class period. The address

space chosen to help is the one that can have the biggest impact on the paging overhead that address spaces in the service class period are experiencing. The change in the auxiliary and expanded storage page-in rate is then used to calculate the change in delay samples, which in turn can be used to get the performance index delta in much the same way as the short response time goal case.

The individual address space paging plot is used to project the effect of these changes.

There are two address space paging plots:

- The central storage plot is used by WLM when assessing increasing or decreasing the central storage allocated to an address space. In other words, this plot tracks how an address space performs in relation to the amount of central storage the address space has.

  The plot has the following attributes:

  – x axis – address space processor storage size in frames.

  – y axis – each one of the following:

    i Page-in rate per captured (TCB + SRB) second from auxiliary and expanded storage measured in milliseconds divided by its residency time measured in seconds.

    ii Paging cost in milliseconds per elapsed second from auxiliary and expanded storage, which is in fact the CPU cost of paging from auxiliary and expanded storage measured in terms of the CPU cost of paging to/from auxiliary and expanded storage.

    iii Captured time in milliseconds per elapsed second.

- The processor storage plot is used by WLM when assessing increasing or decreasing the processor storage allocated to an address space:

- x axis – the address space size in frames.
- y axis – each one of the following:
  - i Page-in rate per captured (task+SRB) second from auxiliary storage.
  - ii Paging cost in milliseconds per elapsed second from auxiliary storage.
  - iii Captured time in milliseconds per elapsed second.

I/O delay plot

WLM uses the I/O delay plot while assessing increasing or decreasing a service class period's I/O priority. One should remember that I/O priority queueing is used to control non-paging DASD I/O requests that are queued because the device is busy. The default for I/O priority management is no, which sets I/O priorities equal to dispatching priorities. If you specify yes, workload management sets I/O priorities in the sysplex based on goals. WLM dynamically adjusts the I/O priority based on how well each service class is meeting its goals and whether the device can contribute to meeting the goal. The system does not micro-manage the I/O priorities, and changes a service class period's I/O priority infrequently. If you specify I/O priority management, workload management dynamically sets I/O priorities based on goals and I/O activity. The new DASD I/O using and DASD I/O delay samples are reported even when I/O priority management is turned off. This allows you to calculate the new velocity values to plan for velocity changes.

The plot has the following attributes:

- x axis is the combined maximum I/O demand of service class periods with I/O priorities above a given priority. Maximum I/O demand is the maximum percentage of time that work units in a service class period would use non-paging DASD devices if they were experiencing no I/O delay. Maximum I/O demand is represented as a percentage scaled by 10.

- y axis – the ratio of I/O delay time to I/O using time at that priority scaled by 16.

Queue delay plot

WLM uses the queue delay plot to assess creating or removing server address spaces. The plot shows how queue delay for a service class period may be reduced by adding server address spaces for work running in the service class period or how queue delay may be increased by removing server address spaces for work running in the service class period.

The main reason for using this plot comes from WLM's design. As we know, WLM manages a separate queue for every application environment, and the server address spaces managed by WLM are created to serve work associated with a specific application environment queue. Defining multiple application environments allow you to separate your workload into different address spaces. In addition, within an application environment there is a separate queue for each service class. This means that work run with different goals runs in separate address spaces and can be managed independently. The workload requirements for a given application may determine that multiple server address spaces should be activated. WLM decides to activate a new address space based on whether:

- There is available capacity (CPU and storage) in the system and work in a service class is being delayed waiting for the WLM queue.
- A service class is missing its goals, it is suffering significant queue delay, and other work (donor) in the system can afford to give up resources to the work missing goals (receiver).

There is no mechanism provided for your intervention in this WLM process other than the ability to limit the number of servers to 1. WLM controls the activation and deactivation of an address space and also controls the number of address spaces.

The plot has the following attributes:

- x axis – the ratio of work requests that require a task in a server address space to the number of server tasks. This ratio is scaled by 100.
- y axis – the queue delay per work request in milliseconds.

I/O velocity plot

SRM uses the I/O velocity plot to keep track of the relationship between the number of channel paths connected to a subsystem (along with the channel path's utilization) and the I/O velocity of that subsystem. There is one I/O velocity plot for each I/O subsystem (control unit). If there is a single channel, the contention factor equals the channel utilization. If there are multiple channels, the contention factor is the utilization a single channel would have in order to be equivalent to the channels connected to the control unit, given the number of channels and their average utilization. In this case, 'equivalent' means that an I/O operation would have the same probability of experiencing delay.

The plot has the following attributes:

- x axis – the contention factor.
- y axis – the I/O velocity.

Collecting WLM's prediction data and reporting

When enabled by the SMFPRM*xx* TYPE parameter of your system parameter library, SMF creates record type 99, which provides detailed audit information for work run in workload management goal mode. One can use the type 99 records for analysing the performance characteristics of work. The records contain performance data for each service class period, a trace of WLM actions (which I had dealt with in an earlier article, see 'A peek in WLM's decision making', *MVS Update*, November and December 2004, issues 218 and 219), the data WLM used to decide which actions to take, and the internal

controls WLM uses to manage work. This can help you determine in detail what WLM is doing to meet your work's goals with respect to other work, and the types of delays the work is experiencing. WLM writes type 99 records for each policy interval, or approximately once every 10 seconds. The records contain the data, plots, and tables WLM uses to assess the effects of changes. A detailed description of the layout of SMF type 99 record and its subtypes can be obtained from the *MVS System Management Facilities (SMF)* (SA22-7630) manual. You can also find the subtypes descriptions in macro IRASMF99 in SYS1. AMODGEN.

In order to provide a starting point from which one can begin to ponder over WLM's prediction process I have coded a sample report writer. The code is a four-part stream: the first part (DEL99) is a clean-up step, which deletes the files to be used in later steps. In the second step (EXT99) SMF records 99 are extracted from the SMF dataset to a file, which can be used as a base of archived records. In the next part (SORT99) previously extracted records (selection being defined by INCLUDE's condition) are being sorted and copied to a file, which is the input to WLMPLOT EXEC invoked in WLMREXX step.

```
//DEL99     EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=X
//SYSIN     DD *
    DELETE hlq.R99.DATA
    DELETE hlq.SMFCOPY.OUT
    SET MAXCC=Ø
/*
//EXT99    EXEC PGM=IFASMFDP,REGION=5M
//INDA1 DD DSN=your.smf.dataset,DISP=SHR
//OUTDA DD DSN=hlq.SMFCOPY.OUT,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(CYL,(6Ø,15),RLSE),
//         DCB=(your.smf.dataset)
//SYSPRINT DD SYSOUT=X
//SYSIN DD *
       DATE(yyyyddd,yyyyddd)
       START(Ø9ØØ)
       END(17ØØ)
       INDD(INDA1,OPTIONS(DUMP))
       OUTDD(OUTDA,TYPE(99(1,3,4,5,9)))
/*
```

```
//SORT99   EXEC PGM=ICETOOL
//TOOLMSG  DD SYSOUT=*
//DFSMSG   DD SYSOUT=*
//RAWSMF   DD DSN=hlq.SMFCOPY.OUT,DISP=SHR
//SMF9912  DD DSN=hlq.R99.DATA,
//          SPACE=(CYL,(30,15)),UNIT=SYSDA,DISP=(NEW,KEEP),
//          DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760)
//TOOLIN   DD *
  SORT FROM(RAWSMF) TO(SMF9912) USING(SMF9)
//SMF9CNTL DD *
*
* Eliminate Header and Trailer records
* Sort by date, time and subtype
*
  OPTION SPANINC=RC4,VLSHRT
  INCLUDE COND=(6,1,BI,EQ,99)
  SORT FIELDS=(11,4,PD,A,7,4,BI,A,23,2,BI,A)
/*
//WLMREXX  EXEC PGM=IKJEFT01,REGION=0M
//SYSEXEC  DD DISP=SHR,DSN=your.rexx.lib
//SMF746   DD DISP=SHR,DSN=hlq.R99.DATA
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
prof nopref
%WLMPLOT
/*
```

WLMPLOT EXEC

```
/* REXX EXEC to read and format SMF records pertaining to
   WLM's predictions: Type 99 subtypes 1, 3, 4, 5, and 9         */

/* signal ON ERROR */
ADDRESS TSO
  userid=SYSVAR(SYSUID)
  plot   =userid||'.wlm.plot.rep'     /* Plot report dsn         */
x = MSG('ON')
IF SYSDSN(plot) = 'OK'
THEN "DELETE "plot" PURGE"
"ALLOC FILE(S99P) DA("plot")",
   " UNIT(SYSALLDA) NEW TRACKS SPACE(300,150) CATALOG",
   " REUSE RELEASE LRECL(400) RECFM(F B)"
  mq = 1
  mh = 1
numeric digits 20
 'EXECIO * DISKR SMF ( STEM x. FINIS'
    do i = 1 to x.0
 /* do i = 1 to x.0   */
  SMF99RTY = c2d(substr(x.i,2,1))               /* SMF record type */
```

```
  SMF99TID = c2d(substr(x.i,19,2))              /* Record subtype  */
   /*-------------------------------------------------------------*/
  /* Unpack SMF date & decode SMF time                           */
   /*-------------------------------------------------------------*/
     SMF99DTE = substr(c2x(substr(x.i,7,4)),3,5) /* unpack SMF date */
     SMF99TME = smf(c2d(substr(x.i,3,4)))
     SMF99SID = substr(x.i,11,4)                        /*System id.*/
     SMF99SSID= substr(x.i,15,4)                  /* Sub System Id.*/
   /*-------------------------------------------------------------*/
  /*          Self-Defininig section                             */
   /*-------------------------------------------------------------*/
 POF  = c2d(substr(x.i,25,4))     /* Offset to the PRODUCT section */
 PLN  = c2d(substr(x.i,29,2))     /* Length of the product section */
 PON  = c2d(substr(x.i,31,2))     /* Number of product sections    */
 DOF  = c2d(substr(x.i,33,4))     /* Offset to the DATA section    */
 DLN  = c2d(substr(x.i,37,2))     /* Length of the data section    */
 DON  = c2d(substr(x.i,39,2))     /* Number of data sections       */
   /*-------------------------------------------------------------*/
  /*          PRODUCT information                                */
   /*-------------------------------------------------------------*/
Select
  when  i = 1 then do
 POF = POF - 3
 SMF99VN2   = c2d(substr(x.i,pof,2))       /* Record sub version   */
 SMF99RVN   = c2d(substr(x.i,pof+2,2))     /* Record Version Number*/
 SMF99PNM   =     substr(x.i,pof+4,8)      /* Product Name - SRM   */
 SMF99SLV   =     substr(x.i,pof+12,8)     /* System level         */
 SMF99SNM   =     substr(x.i,pof+2Ø,8)     /* System name          */
  end
  OTHERWISE    nop
End

 IF SMF99TID = '3' Then call SUBTYPE3  DOF
 IF SMF99TID = '4' Then call SUBTYPE4  DOF
 IF SMF99TID = '5' Then call SUBTYPE5  DOF
 IF SMF99TID = '9' Then call SUBTYPE9  DOF
 IF SMF99TID = '1' Then
 Do
   /*-------------------------------------------------------------*/
  /*     Subtype 1: Self-defining section                        */
   /*-------------------------------------------------------------*/
 DOF  = DOF - 3
 TOF  = c2d(substr(x.i,dof,4))     /* Offset to TRACE sec.         */
 TLN  = c2d(substr(x.i,dof+4,2))   /* Length of Trace sec.         */
 TON  = c2d(substr(x.i,dof+6,2))   /* Number of Trace sec.         */
 SSOF = c2d(substr(x.i,dof+8,4))   /* Offset to SYSTAM STATE sec.  */
 SSLN = c2d(substr(x.i,dof+12,2))  /* Length of System state sec.  */
 SSON = c2d(substr(x.i,dof+14,2))  /* Number of system state sec.  */
 PPOF = c2d(substr(x.i,dof+16,4))  /* Offset to PAGING plot sec.   */
 PPLN = c2d(substr(x.i,dof+2Ø,2))  /* Length of Paging plot sec.   */
```

```
 PPON = c2d(substr(x.i,dof+22,2))  /* Number of paging plot sec.   */
 PTOF = c2d(substr(x.i,dof+24,4))  /* Offset to PRIORITY table sec.*/
 PTLN = c2d(substr(x.i,dof+28,2))  /* Length of Priority table sec.*/
 PTON = c2d(substr(x.i,dof+3Ø,2))  /* Number of Priority table sec.*/
 RGOF = c2d(substr(x.i,dof+32,4))  /* Offset to RESOURCE Group sec.*/
 RGLN = c2d(substr(x.i,dof+36,2))  /* Length of resource group sec.*/
 RGON = c2d(substr(x.i,dof+38,2))  /* Number of resource group sec.*/
 GROF = c2d(substr(x.i,dof+4Ø,4))  /* Offset to generic resouce se.*/
 GRLN = c2d(substr(x.i,dof+44,2))  /* Length of generic resource   */
 GRON = c2d(substr(x.i,dof+46,2))  /* Number of generic resource se*/
 SLOF = c2d(substr(x.i,dof+48,4))  /* Offset to SW licensing sec.  */
 SLLN = c2d(substr(x.i,dof+52,2))  /* Length of SW licensing sec.  */
 SLON = c2d(substr(x.i,dof+54,2))  /* Number of SW licensing sec.  */
 SLTOF= c2d(substr(x.i,dof+56,4))  /* Offset to SW licensing       */
                                   /* service table sec.           */
 SLTLN= c2d(substr(x.i,dof+6Ø,2))  /* Length of SW licensing       */
                                   /* service table sec.           */
 SLTON= c2d(substr(x.i,dof+62,2))  /* Number of SW licensing       */
                                   /* service table sec.           */
    /*-------------------------------------------------------------*/
    /*    Subtype 1: Software Licensing Information -              */
    /*    Contains information about license manager.              */
    /*-------------------------------------------------------------*/
if  (SLOF <> Ø) & (SLLN <> Ø) Then
DO
slof = slof - 3
    /*-------------------------------------------------------------*/
    /*           Flags (Subtype 1)                                 */
    /*-------------------------------------------------------------*/
SLCONFFLG=x2b(c2x(substr(x.i,slof,1)))    /* Configuration flags      */
SLSTSI = substr(SLCONFFLG,1,1)          /* Indicates that the machine */
                                        /* supports the store system  */
select                                  /* information instruction.   */
  when SLSTSI = 1 then op1="Store Sys.Info suported/";
  otherwise         op1= "Store Sys.Info not supported ";
end
SLLPAR = substr(SLCONFFLG,2,1)       /* Indicates that MVS is running */
                                        /* in a logical partition   */
select
  when SLLPAR = 1 then op2="Running in LPAR/";
  otherwise         op2= " ";
end
SLVM   = substr(SLCONFFLG,3,1)         /*Indicates that MVS is running*/
                                       /* in a virtual machine        */
select
  when SLVM   = 1 then op3="Running in VM";
  otherwise         op3= "";
end
SLSHLOG = substr(SLCONFFLG,4,1)           /* Indicates that the       */
                                          /* logical CPUs are shared  */
```

```
                                          /* with other partitions    */
select
  when SLSHLOG= 1 then op4="CPU shared/";
  otherwise        op4= "CPU not shared";
end
SLCAPCUST = substr(SLCONFFLG,5,1)       /* Indicates that the logical */
                                      /* partition is configured to be*/
                                         /* capped (as opposed to    */
                                         /* being capped by WLM)     */
select
  when SLCAPCUST = 1 then op5="LPAR to be capped";
  otherwise        op5= "WLM capped";
end
Config =op1||op2||op3||op4||op5
SLSTATFLGS=x2b(c2d(substr(x.i,slof+1,1))) /* State flags.             */
SLCAPPYWLM=substr(SLSTATFLGS,1,1)        /* Indicates that the logical*/
                                         /* partition is capped by WLM*/
select
  when SLCAPPYWLM = 1 then State= "LPAR is actually WLM capped";
  otherwise                State= "LPAR is no actually WLM capped";
end
   /*-----------------------------------------------------------------*/
   /*    The following fields describe the/MVS image                  */
   /*    and CEC capacity. (Subtype 1)                                */
   /*-----------------------------------------------------------------*/
SLIMGCAPACITY=c2d(substr(x.i,slof+4,4))   /* Capacity available to MVS*/
                                  /* image in millions of service    */
                                  /* units per hour,when not running */
                                  /* as VM guest. If running as VM   */
                                  /* guest, capacity available to VM.*/
SLCECCAPACITY=c2d(substr(x.i,slof+8,4))
                                  /* Capacity of CEC in millions     */
                                  /* of service units per hour.      */
   /*-----------------------------------------------------------------*/
   /*    The following fields describe the/CEC and MVS                */
   /*    image configuration. They are used to calculate the CPU      */
   /*    capacity of the CEC and MVS image (Subtype 1)                */
   /*-----------------------------------------------------------------*/
SLCECCPUNUM  =c2d(substr(x.i,slof+12,2))  /* Number of available CPUs */
                                  /* in the CEC. This includes online*/
                                  /* and offline CPUs. It does not   */
                                  /* include reserved CPUs (CPUs     */
                                  /* that can be added via           */
                                  /* Capacity Upgrade on Demand).    */
SLLOGICALCPUNUM=c2d(substr(x.i,slof+14,2))
                                    /* Number of available CPUs in   */
                                  /* the logical partition. This     */
                                  /* includes online and offline     */
                                  /* CPUs. It does not include       */
                                  /* reserved CPUs (CPUs that can    */
```

```
                                    /* be added via Cap.Upgrade on     */
                                    /* Demand).                        */
SLCECSUSECSHARE=c2d(substr(x.i,slof+16,4)) /*The CEC capacity in       */
                                    /* basic-mode service units per    */
                                    /* second that is available for    */
                                    /* sharing among partitions        */
                                    /* using shared logical processors.*/
SLIMGMSUCURRWEIGHT = ,              /* MVS image capacity in millions  */
            c2d(substr(x.i,slof+20,4))    /* of service units per hour*/
                                    /* that is represented by the      */
                                    /* partition's current weight.     */
   /*---------------------------------------------------------------*/
   /*   The following fields describe the logical                   */
   /*   partition's actual CPU usage. (Subtype 1)                   */
   /*---------------------------------------------------------------*/
SLAVGMSU    =c2d(substr(x.i,slof+28,4))   /* Average service rate in  */
                                       /* millions of service units */
                                    /* per hour. This is a long-term*/
                                       /* average.                  */
SLAVGMSUCAP =c2d(substr(x.i,slof+32,4))   /* Average service rate in  */
                                       /* millions of service units*/
                                       /* per hour while the       */
                                    /* partition was capped. This  */
                                       /* is a short-term average. */
SLAVGMSUUNCAP=c2d(substr(x.i,slof+36,4))  /* Average service rate in  */
                                       /* millions of service units*/
                                    /* per hour while the partition*/
                                       /* was uncapped. This is    */
                                       /* a short-term average     */
SLINTERVALSER =c2d(substr(x.i,slof+40,4)) /* Service units over last  */
                                    /* policy adjustment interval.     */
                                    /* NOTE: The service units are     */
                                    /* calculated using the MP         */
                                    /* factor for the number of        */
                                    /* physical CPUs, not the number   */
                                    /* of logical CPUs. This is        */
                                    /* consistent with how capacity    */
                                    /* is measured for software        */
                                    /* licensing. These service        */
                                    /* units cannot be directly        */
                                    /* compared to other service       */
/* units calculated by SRM.         */
SLINTERVALTIME=c2d(substr(x.i,slof+44,4)) /* Elapsed time over last   */
                                    /* policy adjustment interval in   */
                                    /* 1.024 milliseconds              */
SLROLLINTERVAL=c2d(substr(x.i,slof+52,2))    /* Number of policy     */
                               /*adjustment intervals between computation*/
                                    /* of average service rate.        */
SLSERVTABINT  =c2d(substr(x.i,slof+54,2)) /* Number of consecutive    */
                                    /* policy adjustment intervals that*/
```

```
                                    /* have passed since the last      */
                                    /* time that the service table     */
                                    /* was updated.                    */
   /*---------------------------------------------------------------*/
   /*   The following fields describe the cap pattern  (Subtype 1)    */
   /*---------------------------------------------------------------*/
SLINTRCA3=c2d(substr(x.i,slof+56,2))      /* Number of consecutive    */
                                    /* policy adjustment intervals     */
                                    /* to cap the partition.           */
SLINTRUNCAP=c2d(substr(x.i,slof+58,2))    /* Number of consecutive    */
                                    /* policy adjustment intervals     */
                                    /* to uncap the partition.         */
SLPATINTRVNUM=c2d(substr(x.i,slof+60,2))  /* Number of consecutive    */
                                     /*policy adjustment intervals that*/
                                    /* have passed in the current      */
                                    /* cap/uncap state indicated       */
                                    /* by SLCap- pedByWlm.             */
   /*---------------------------------------------------------------*/
   /*    The following fields are response codes (Subtype 1)          */
   /*---------------------------------------------------------------*/
SLQUERYRC =c2d(substr(x.i,slof+64,4))   /* Response code from the last*/
                                    /*'query' for lpar information     */
SLSETCAPRC =c2d(substr(x.i,slof+68,4))  /*Response code from the last */
 select
  when  i = 1 then do
  binfo.1= left('System id   :',13) left(SMF99SID,5)
  binfo.2= left('System level:',13) left(SMF99SLV,8)
  binfo.3= left('System name :',13) left(SMF99SNM,8)
  binfo.4= left('Image  MSU  :',13) left(SLIMGCAPACITY,4),
           left(' CPU num.:',10)      left(SLCECCPUNUM,4)
  binfo.5= left('System configuration FLAGS:',30)
  binfo.6= left(config,90)
  binfo.7= left('System configuration STATE:',30)
  binfo.8= left(state,45)
  binfo.9= left('SU/sec LPAR shared:',20) left(SLCECSUSECSHARE,6)
  binfo.10=left('Policy adj.computational intervals:',36),
           left(SLROLLINTERVAL,4)
      "EXECIO * DISKW s99p (STEM binfo.)"
  end
  OTHERWISE    nop
end
END                              /* of Software Licensing Information*/
   /*---------------------------------------------------------------*/
   /*  Subtype 1:    Software Licensing table                       */
   /*  Information - Contains information about license manager     */
   /*---------------------------------------------------------------*/
 if  (SLTOF <> 0) & (SLTLN <> 0) Then
 DO
 sltof = sltof - 3
   /*---------------------------------------------------------------*/
```

```
   /* The following fields describe the logical partition's actual    */
   /* CPU usage. The data will be filled in from oldest to newest     */
   /* table entries. (Subtype 1)                                      */
    /*---------------------------------------------------------------*/
SLTSERVICEUNCAPPED = ,              /* Basic-mode service units       */
     c2d(substr(x.i,sltof,4))     /* accumulated while the partition  */
                              /* was uncapped. NOTE: The service units */
                              /* are calculated using the MP factor    */
                              /* for the number of physical CPUs,      */
                              /* not the number of logical CPUs.       */
                              /* This is consistent with how capacity  */
                              /* is measured for software licensing.   */
                              /* These service units cannot be directly*/
                              /* compared to other service units       */
                              /* calculated in SRM.                    */
SLTSERVICECAPPED  = ,                 /* Basic-mode service units       */
     c2d(substr(x.i,sltof+4,4))       /*accumulated while the partition */
                                      /* was capped. NOTE: The service  */
                              /* units are calculated using the        */
                              /* MP factor for the number of physical  */
                              /* CPUs, not the number of logical       */
                              /* CPUs. This is consistent with how     */
                              /* capacity is measured for software     */
                              /* licensing. These service units        */
                              /* cannot be directly compared to other  */
                              /* service units calculated in SRM.      */
SLTSERVICEUNCAPPEDCOUNT  = ,          /* Number of seconds that the     */
      c2d(substr(x.i,sltof+8,2))      /* partition was uncapped.        */
SLTSERVICECAPPEDCOUNT     = ,         /* Number of seconds that the     */
     c2d(substr(x.i,sltof+10,2))      /* partition was capped.          */
SLTSERVICELASTUPDATEINTERVAL  = ,     /* Policy adjustment interval id  */
     c2d(substr(x.i,sltof+12,2))     /*when this entry was last updated.*/
                                      /* This field is set in goal mode */
                                      /* only. Since the id is only 1   */
                              /* byte, it will wrap multiple times     */
                              /* over the course of the table.         */
                              /* (That is, the time span of the        */
                              /* table is greater than 255 intervals   */
                              /* so the interval ids will              */
                              /* wrap around.)                         */
END                           /* of  Software Licensing table          */
   /*---------------------------------------------------------------*/
  /* Subtype 1: System State information -                           */
  /* Contains information about the general state of the system      */
   /*---------------------------------------------------------------*/
if  (SSOF <> 0) & (SSLN <> 0) Then
DO
ssof = ssof - 3
CPUA=c2d(substr(x.i,ssof,2))  /*Avg.Processor Utilization scaled by 16*/
UMP =c2d(substr(x.i,ssof+2,2))
```

```
                            /*Recent unmanaged paging and swap cost %  */
UIC1=c2d(substr(x.i,ssof+4,4))    /* Frames in UIC bucket 1 - The     */
                                  /* central and expanded UIC buckets */
                           /* contain a count of frames that have      */
                           /* not been referenced for specified        */
                           /* periods of time.  So, UIC bucket 1       */
                           /* is a count of the frames that have       */
                           /* most recently been referenced, where     */
                           /* bucket 4 contains a count of the         */
                           /* frames that have not been referenced     */
                           /* in a long time. The UIC delimiter values */
                           /* specify the cutoff points for each bucket*/
                           /* For example,EUIC3 will contain the count */
                           /* of all the expanded storage frames whose */
                           /* time since they were last referenced is  */
                           /* between ESTB2 and ESTB3.                 */
UIC2=c2d(substr(x.i,ssof+8,4))    /* Frames in UIC bucket 2           */
UIC3=c2d(substr(x.i,ssof+12,4))   /* Frames in UIC bucket 3           */
UIC4=c2d(substr(x.i,ssof+16,4))   /* Frames in UIC bucket 4           */
EUIC1 =c2d(substr(x.i,ssof+20,4)) /* Frames in Expanded UIC bucket 1  */
EUIC2 =c2d(substr(x.i,ssof+24,4)) /* Frames in Expanded UIC bucket 2  */
EUIC3 =c2d(substr(x.i,ssof+28,4)) /* Frames in Expanded UIC bucket 3  */
EUIC4 =c2d(substr(x.i,ssof+32,4)) /* Frames in Expanded UIC bucket 4  */
FRV1  =c2d(substr(x.i,ssof+36,2)) /* UIC delimiter value 1            */
FRV2  =c2d(substr(x.i,ssof+38,2)) /* UIC delimiter value 2            */
FRV3  =c2d(substr(x.i,ssof+40,2)) /* UIC delimiter value 3            */
ESTB1 =c2d(substr(x.i,ssof+42,2)) /*Expanded storage UIC del. value 1 */
ESTB2 =c2d(substr(x.i,ssof+44,2)) /*Expanded storage UIC del. value 2 */
ESTB3 =c2d(substr(x.i,ssof+46,2)) /*Expanded storage UIC del. value 3 */
W2MIG =c2d(substr(x.i,ssof+48,4)) /* Write to migrate percentage      */
PTAVAIL=smf(c2d(substr(x.i,ssof+52,4))) /* Total CPU time available   */
                           /* incl. captured time plus wait time       */
SHFLAGS = x2b(c2x(substr(x.i,ssof+56,1)))
    CSS=  substr(SHFLAGS,1,1)
    AUF=  substr(SHFLAGS,2,1)
    AUC=  substr(SHFLAGS,3,1)
    SQF=  substr(SHFLAGS,4,1)
    SQC=  substr(SHFLAGS,5,1)
 if CSS = 1      then shortage='Real storage shortage'
 else if AUF = 1 then shortage='First level Aux shortage'
 else if AUC = 1 then shortage='Critical Aux shortage'
 else if SQF = 1 then shortage='First level SQA shortage'
 else if SQC=  1 then shortage='Critical SQA shortage'
 else                 shortage='No Storage shortage'
STFLAGS = x2b(c2x(substr(x.i,ssof+57,1)))
   DCMON   = substr(STFLAGS,1,1)
   DCMGOAL = substr(STFLAGS,2,1)
   COMPAT  = substr(STFLAGS,3,1)
 if DCMON=1        then status='Dynamic chpid active'
 else if DCMGOAL=1 then status='Dynamic chpid goal is active'
```

```
 else if COMPAT=1  then status='in COMPAT mode'
 else                    status='GOAL Mode, no Dynamic chpid'
TOTPAGCOST = c2d(substr(x.i,ssof+58,2))  /* Recent total paging and   */
                                         /* swap cost percentage * 10 */
Select
  when TOTPAGCOST >'0' then TOTPAGCOST=format(TOTPAGCOST/10,3,2)
  otherwise nop
End
CPPS       = c2d(substr(x.i,ssof+60,4))  /*Common protective processor*/
                                         /* storage target in frames  */
ILSUARRAY  = c2x(substr(x.i,ssof+64,32))
                                /* Array of Importance Level SU      */
    k = 1
    j = 0
 do while j < 8
   imp.j  = c2d(substr(ILSUARRAY,k,8))       /* A single entry in the */
/*array of say  imp.j   */       /* Importance Level Service Units    */
   j = j + 1                    /* consumed by work at this          */
   k = k + 8                    /* this Importance Level over the    */
 end                            /* last ten seconds                  */
STWSS    = c2d(substr(x.i,ssof+96,4))   /*Protective processor storage*/
                               /* target for shared area, measured   */
                               /* in frame counts.                   */
NUMEXTSC = c2d(substr(x.i,ssof+100,4))
                               /* No. of external serv. classes      */
DEFIOVEL = c2d(substr(x.i,ssof+104,4))  /* Default I/O velocity.      */
                               /* Calculated by IOS at the beginning */
                               /* of each measurement interval during*/
                               /* data gathering.                    */
SUIFACTOR= c2d(substr(x.i,ssof+108,4))
                               /* Service unit inflation factor.     */
 acpu = format(CPUA/16,3,2)
  sysst.1= left(' ',1)
  sysst.2= left(date('n',SMF99DTE,'j'),11)
  sysst.3= left('Part 1: System State information at interval:',50),
           left(SMF99TME,12) left(' ',28,' ') ,
           left('SU',7)      left('Def.',4)
  sysst.4= left('  CPU%',6) left('CPU available',13),
           left('uic1',5) left('uic2',5) left('uic3',5) left('uic4',4),
           left('Storage shortage',21)      left('UMPS%',5),
           left('Paging%',7) left('CPPS',4) left('STWSS',5),
           left('inflat.',7) left('IO Vel.',7)
  sysst.5= left('-',105,'-')
  sysst.6= left(acpu,6)  left(PTAVAIL,12),
           right(uic1,5)     right(uic2,5)  right(uic3,5) right(uic4,5),
           left(shortage,20) right(UMP,6)   right(TOTPAGCOST,6),
           right(cpps,4)     right(STWSS,4) right(SUIFACTOR,5),
           right(DEFIOVEL,6)
  sysst.7= left('-',105,'-')
    "EXECIO * DISKW s99p (STEM sysst.)"
```

```
END
   /*-----------------------------------------------------------------*/
   /*  Paging plot information - Contains the information plotted   */
   /*  on the system paging responsivness plot. (Subtype 1)         */
   /*-----------------------------------------------------------------*/
 if  (PPOF <> Ø) & (PPLN <> Ø) Then
 DO
 ppof = ppof - 3
 PAGP_BW        = c2d(substr(x.i,ppof,4))        /*  Bucket width     */
 PAGP_LSTX      = c2d(substr(x.i,ppof+4,4))    /*Last plotted x bucket*/
 PAGP_POINTS_OF = c2d(substr(x.i,ppof+12,4)) /*Offset of point entries*/
 PAGP_POINTS_ON = c2d(substr(x.i,ppof+16,4)) /*Number of point entries*/
 PAGP_POINTS_LN = c2d(substr(x.i,ppof+18,4)) /*Length of a point entry*/
Select
 when PAGP_LSTX  > Ø then call XYPROC ,
    PAGP_POINTS_OF  PAGP_POINTS_LN  PAGP_POINTS_ON  PAGP_LSTX
 otherwise nop
 End
 END
   /*-----------------------------------------------------------------*/
   /*  Subtype 1:   Resource Group entry - Contains                   */
   /*  information about resource groups                              */
   /*-----------------------------------------------------------------*/
 if  (RGOF <> Ø) & (RGLN <> Ø) Then
 DO
 rgof = rgof - 3
RGNAME=     substr(x.i,rgof,8)     /* Resource group name             */
MINSR = c2d(substr(x.i,rgof+8,4)) /*Minimum service rate for the group*/
MAXSR = c2d(substr(x.i,rgof+12,4))
                             /* Maximum service rate for the group     */
                            /*(FFFFFFFF if not specified in the policy)*/
ACTSR = c2d(substr(x.i,rgof+16,4)) /* Service rate received in last   */
                             /* interval on the local system    */
SPAS  = c2d(substr(x.i,rgof+2Ø,4)) /* Service per awake slice         */
SLICES= c2d(substr(x.i,rgof+24,2)) /* Number of time slices that      */
                             /* work in this group was capped   */
RHELPØ= c2d(substr(x.i,rgof+26,2)) /* A count of the remote systems   */
                            /* that can help work at importance Ø     */
RHELP1= c2d(substr(x.i,rgof+28,2)) /* A count of the remote systems   */
                              /* that can help work at importance 1*/
RHELP2= c2d(substr(x.i,rgof+3Ø,2)) /* A count of the remote systems   */
                             /* that can help work at importance 2*/
RHELP3= c2d(substr(x.i,rgof+32,2)) /* A count of the remote systems   */
                            /* that can help work at importance 3 */
RHELP4= c2d(substr(x.i,rgof+34,2)) /* A count of the remote systems   */
                            /* that can help work at importance 4     */
RHELP5= c2d(substr(x.i,rgof+36,2)) /* A count of the remote systems   */
                            /* that can help work at importance 5     */
RHELP6= c2d(substr(x.i,rgof+38,2)) /* A count of the remote systems   */
                            /* that can help work at importance 6     */
```

```
LHELPF= x2b(c2d(substr(x.i,rgof+40,1)))  /* Local system can help work*/
                            /* at each importance. 1 - can help       */
                            /* 0 - cannot help                        */
LHELP6 =  substr(LHELPF,2,1)  /*Local system help work at importance 0*/
                                  /* 1 - can help, 0 - cannot help   */
LHELP5 =  substr(LHELPF,3,1)  /*Local system help work at importance 1*/
                                  /* 1 - can help, 0 - cannot help   */
LHELP4 =  substr(LHELPF,4,1)  /*Local system help work at importance 2*/
                                  /* 1 - can help, 0 - cannot help   */
LHELP3 =  substr(LHELPF,5,1)  /*Local system help work at importance 3*/
                                  /* 1 - can help, 0 - cannot help   */
LHELP2 =  substr(LHELPF,6,1)  /*Local system help work at importance 4*/
                                  /* 1 - can help, 0 - cannot help   */
LHELP1 =  substr(LHELPF,7,1)  /*Local system help work at importance 5*/
                                  /* 1 - can help, 0 - cannot help   */
LHELP0 =  substr(LHELPF,8,1)  /*Local system help work at importance 6*/
                                  /* 1 - can help, 0 - cannot help   */
RGFLAGS=x2b(c2d(substr(x.i,rgof+41,1))) /* Resource group flags       */
RGDYNAMIC = substr(RGFLAGS,1,1)    /*Indicates that the resource group*/
                                  /* is a dynamic                    */
select
  when RGDYNAMIC = '1' then rgdyn="Res.Group is dynamic"
  otherwise                 rgdyn="Res.Group is not dynamic"
end
  rgeg.mq= left(date('n',SMF99DTE,'j'),11),
           left(SMF99TME,12),
           left(rgname,8),             /* Resource group name    */
          right(minsr,5),              /* Group min.service rate */
          right(maxsr,5),              /* Group max.service rate */
          right(actsr,5),              /* Service rate received  */
          right(spas,5),               /* Service per awake slice*/
          right(slices,5)              /* No. of time slices     */
 mq = mq + 1
END                                  /* of Resource Group entry  */
/*-----------------------------------------------------------------*/
/*     Subtype 1: Generic Resource entry -                         */
/*     Contains information about Generic Resource Routing         */
/*-----------------------------------------------------------------*/
if  (GROF <> 0) & (GRLN <> 0) Then
DO
grof = grof - 3
GRSYSNAME =    (substr(x.i,grof,8))
                                  /*Name of sys. sessions were routed*/
GRTSO     = c2d(substr(x.i,grof+8,4))
                                  /* Number of TSO sessions routed in*/
                                      /* the last 10 sec.            */
GRNONTSO  = c2d(substr(x.i,grof+12,4))
                                /* Number of non-TSO sessions routed   */
                                      /* in last 10 sec.             */
GRTSOAVG  = c2d(substr(x.i,grof+16,4)) /* Average cost of TSO session */
```

```
                                /* in raw cpu service units on system */
GRTSOPI   = c2d(substr(x.i,grof+2Ø,4))    /* Weighted average PI of   */
                 /* service class periods running TSO work on system  */
GRFLAGS   = x2b(c2x(substr(x.i,grof+24,4))) /*Flags                   */
GRSHORTAGE= c2d(substr(GRFLAGS,1,1))   /*GRSYSNAME had a shortage that*/
                                  /* may have caused sessions not to  */
                                  /* be routed to it                  */
GRSERVIMP = c2x(substr(x.i,grof+28,32))  /*A single entry in the array*/
                                 /* of Importance Level Service Units,*/
                                  /* containing the number of Service  */
                                  /* Units consumed by work at this    */
                                /* Importance Level (or unused) over   */
                                /* the last ten seconds. The entries   */
                                /* are indexed with an origin of zero  */
                                /* so that the index matches the       */
                                /* Importance Level to which the       */
                                /* entry pertains. Index of zero       */
                                /* corresponds to system work and      */
                                /* index of 7 to unused capacity       */
    z = 1
    w = Ø
 do while w < 8
  simp.w  = c2d(substr(GRSERVIMP,z,8))
   w = w + 1
   z = z + 8
 end
  gges.mh= left(date('n',SMF99DTE,'j'),11),
           left(SMF99TME,12),
          right(GRSYSNAME,8),
          right(grtso,5),                /* No.of TSO sess. routed */
          right(grnontso,5),             /* No.of non-TSO sess.    */
          right(grtsoavg,5),             /* Avg. cost of TSO sess. */
          right(grtsopi,5),              /* Weighted average PI    */
          right(GRSHORTAGE,5)
    mh = mh + 1
END                                     /* of Generic Resource entry*/
   /*-----------------------------------------------------------------*/
   /*  Subtype 1:  Priority table entry - Contains                    */
   /*  information pertaining to the demand being put on the          */
   /*  processor by work at different dispatch priorities.            */
   /*-----------------------------------------------------------------*/
  ptt.1  = left(' ',1)
  ptt.2  = left('Part 2: PRIORITY TABLE at interval',5Ø)
  ptt.3  = left(' ',9,' ') left('% CPU demand',16) left('CPU used',12),
           left('Cum. max demand:',16) left('MTTW avg',12),
           left('Cpu samp.',11) left('Sample based',12)
  ptt.4  = left(' DP',5)  left('DPC',3)    left('init',4)
left('proj.',5),
           left('W2UR',5) left('actual',6) left('proj.',5),
           left('ach.',4) left('init',5)   left('proj.',5),
```

```
           left('init',5) left('proj.',6)  left('use',5)  left('dly',5),
           left('use',5)  left('dly',4)
  ptt.5  = left('-',92,'-')
if  (PTOF <> Ø) & (PTLN <> Ø) Then
    "EXECIO * DISKW s99p (STEM ptt. )"
DO
  pty.ext = left(' ',1)
     do r = Ø to PTON -1
       ptff = (PTOF + (r*PTLN))- 3

ext      = r+1
PTPRTY=c2d(substr(x.i,ptff,2))    /* Dispatch priority                */
PTNP  =c2d(substr(x.i,ptff+2,2))
                              /* New dispatch priority (Ø = not changed) */
PTIMDP=c2d(substr(x.i,ptff+4,4))
                              /* Initial max. % of processor demanded    */
                              /* at priority, initial value before       */
                              /* any priority moves or slice changes     */
```

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

Mile Pekic
Systems Programmer (Serbia and Montenegro)

AppWorx has announced AppWorx Version 7. The product automates and integrates application processing, eliminating the manual effort, latency, and human error that impede enterprise application performance. This new version includes a mainframe agent that can control the application processing on a mainframe.

Other new features in AppWorx 7 include enhanced Web services capability with XML support, a SOAP application programming interface, and Java Messaging Service (JMS). AppWorx has also extended its database connectivity via JDBC, so it can connect to DB2, Oracle, and SQL Server.

For further information contact:
URL: www.appworx.com/products/index.cfm.

* * *

BluePhoenix Solutions has added BluePhoenix LogicMiner for COBOL to its Legacy Understanding Solution Suite. The tool provides an analysis of COBOL code, and maps the interaction between various code components, teleprocessing monitors, files systems, databases, and JCL. This information could be used to prepare applications for use in new service-oriented frameworks such as J2EE and .NET.

LogicMiner is built on Eclipse technology and the workbench supports detailed analysis of COBOL programs, COBOL copybooks, CICS and IMS/DC interactions, CICS and IMS maps, VSAM and DL/I file system interactions, as well as database interactions such as SQL for DB2.

For further information contact:
URL: www.bphx.com/index.cfm?headerid=6&catid=6&subcatid=375&subcat=true.

* * *

NEON Enterprise Software has released Lightning DEDB, which is designed to reduce CPU costs and maximize IMS Fast Path DEDB database performance.

Lightning DEDB improves the efficiency of the space search routine in IMS Fast Path DEDB databases enabling all the available IOVF space to be used without incurring performance slow-downs. Automatic alerts can be configured for real-time notification of critical space depletion levels and rates.

For further information contact:
URL: www.neonesoft.com/product_light.html.

* * *

Mainstar has announced Data Set Level Migrate (DSLM), which will move data at the dataset level, independent of DASD manufacturer. DSLM allows non-disruptive migration to occur transparently to critical business applications. DSLM can migrate without application downtime. Business applications can remain online while DSLM migrates the data to larger volumes. DSLM can also migrate an ICF user catalog to a different volume.

For further information contact:
URL: www.mainstar.com/pdf/015-0123_DSM0101_PR.pdf.

* * *