208

MVS update

January 2004

In this issue

MVS Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: trevore@xephon.com

North American office

Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £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 1999 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.

Editor

Trevor Eddolls

Disclaimer

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

Contributions

When Xephon is given copyright, articles published in *MVS Update* are paid for at the rate of £100 ($160) per 1000 words and £50 ($80) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £20 ($32) 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.

What is RSVNONR?

The most recent version of Omegamon exposed an MVS tuning value that was previously not visible to us. We can now get an Omegamon light for the RSVNONR value that is exceeding its threshold. So, what is RSVNONR?

MVS can run many address spaces. The limit is controlled by a value called MAXUSER, which is set in SYS1.PARMLIB(IEASYS00). This establishes the maximum number of address spaces that can be active in an MVS LPAR (batch jobs plus STCs plus INITs plus TSO users). As each task starts, it takes an Address Space ID (ASID). When it ends, it releases the ASID. The ASIDs are reused throughout the lifetime of the IPL. As long as we set MAXUSER high enough to handle our largest workload, there are no problems. If we try to run more than MAXUSER tasks, new work will not start. We have MAXUSER set to 625 on all LPARs of one production sysplex and 473 on alZZZZZZZZl LPARs of another production sysplex. These values are fine and don't present a problem. Our development sysplex, on the other hand, runs many more tasks on a daily basis and is set to 1525. MAXUSER can be changed only with an IPL.

If we use all the 'slots' and more work needs to run, we have to make some decisions. Are we willing to have some TSO users sign off, can we shut down some initiators, or can we shut down some started tasks? This MAXUSER value is set by the system, based on an historical analysis of the workload. The only tool we could find that can monitor this value is CA-Sysview using the ASVT command:

```
SYSVIEW 7.6 SYØ1 ---- Address Space Vector Table ----- Ø6/3Ø/Ø3 11:42:33
Command ====>                                          Scroll *===> PAGE
---------------------------------------- Lvl 2 Row 1-21/753 Col 1-79/173
MAXUSER Size   625    RSVSTRT Size    64    RSVNONR Size    64

        Used   382            Used     Ø            Used     3

        Free   24Ø            Free    64            Free    61
```

```
        Lost     3           Lost     Ø           Lost     Ø

------------------------------------------------------------------------
Cmd       ASID ASCB     Jobname  Stepname Procstep Type  Jobnr Jc Status
          ØØØ1 ØØFC37ØØ *MASTER*                  SYS    4791 $  NS
          ØØØ2 ØØF9128Ø PCAUTH   PCAUTH            SYS            NS
          ØØØ3 ØØF911ØØ RASP     RASP              SYS            NS
          ØØØ4 ØØF94A8Ø TRACE    TRACE             SYS            NS
          ØØØ5 ØØF949ØØ DUMPSRV  DUMPSRV  DUMPSRV  SYS            NS
          ØØØ6 ØØF9478Ø XCFAS    XCFAS    IEFPROC  SYS            NS
          ØØØ7 ØØFADBØØ GRS      GRS               SYS            NS
          ØØØ8 ØØF94E8Ø SMSPDSE  SMSPDSE           SYS            NS
          ØØØ9 ØØF94DØØ CONSOLE  CONSOLE           SYS            NS
          ØØØA ØØF9FØ8Ø WLM      WLM      IEFPROC  SYS            NS
          ØØØB ØØFAD98Ø ANTMAIN  ANTMAIN  IEFPROC  SYS            NS
          ØØØC ØØFAD8ØØ ANTASØØØ ANTASØØØ IEFPROC  STC            NS
          ØØØD ØØFAD68Ø OMVS     OMVS     OMVS     SYS            NS
          ØØØE *LOST***
          ØØØF ØØF4Ø7ØØ IEFSCHAS IEFSCHAS          SYS            NS
          ØØ1Ø ØØF9628Ø JESXCF   JESXCF   IEFPROC  STC            NS
          ØØ11 ØØFB388Ø ALLOCAS  ALLOCAS           SYS            NS
```

Occasionally, an address space can use an ASID incorrectly or the sequence of events can cause an address space to leave an ASID ‘trashed’, and the ASID is ‘lost’ and cannot be reused. This decreases the maximum number of available ASIDs for new work. ASIDs become lost through programs not releasing certain system connections. We have a known history of this with Changeman, DB2, and occasionally the CPSM CMASs. On an historical note, since the FORCE command used to cause this condition, we were always told not to use it. This was fixed in the MVS FORCE command several releases ago. When an ASID is lost, you will see the IEF352I message in the log:

```
IEF352I ADDRESS SPACE UNAVAILABLE
$HASP395 CHGMAN   ENDED
```

The next $HASP395 message will identify the task that trashed the ASID.

When this happens, the systems programmers keep a reserve ‘bucket’ of ASIDs to fill in the holes. This number of reserves (kept in IEASYS00 also) is what RSVNONR represents. As ASIDs are lost, new ones are taken from RSVNONR to keep the MAXUSER at the IEASYS00 value. As long as we are not at

MAXUSER, this really does not matter. Even if all the reserves from RSVNONR are used, it still does not matter if we are not at MAXUSER. The only time we care is when MAXUSER dips to a level that keeps us from starting new work. The screen snapshot above shows the value at 11:42am on Monday 30 June. During the height of the market day, we are at only 61% of MAXUSER. During the night we run more batch, but TSO users drop off, so the number is still way below the MAXUSER value. Omegamon will light up yellow when MAXUSER exceeds 90% and red when MAXUSER exceeds 98%. Since MVS will still continue to process existing work, this is just informational and should be shared with technical support, but does not constitute an emergency situation unless there are a lot of jobs that can't run. This is one of those built-in throttles in MVS. Possibly we are throttling work with Throughput Manager and WLM long before MAXUSER can be exhausted.

So, the debate is whether RSVNONR means anything to us. One suggestion is to just turn off the light. Another suggestion is to predict the worst case scenario based on our known history with 'messy' address spaces. Since we know we have problems when we don't IPL every month, we can calculate the anticipated number of ASIDs we expect to lose during a full month. This number is the number of Change Man bounces (two per day) added to the number of DB2 and CMAS bounces (four per month). This is (30 days * two per day) + (four weeks * one per weekend) + (four weeks * one per weekend) = 68, rounding to 75 for some additional sporadic DB2 address spaces. If we bump RSVNONR to 100 and set the threshold to 75%, we could see when we 'burned' more than the expected number of lost ASIDs in the month. If this happens in less than one month, it suggests we have more bounces of the offenders than normal or a new address space that is contributing to the problem. Either way it becomes valuable information. In the meantime, as long as we are under MAXUSER, the RSVNONR threshold is fairly meaningless.

Robert Zenuk
Systems Programmer (USA)

Using Service Request Blocks (SRBs)

In MVS, the dispatchable units of work can be represented as Task Control Blocks (TCBs) or Service Request Blocks (SRBs). TCBs, representing the tasks executing in an address space, are better understood than SRBs. This article aims to help readers to understand SRBs, and also their restrictions compared with TCBs, so that they can use them more effectively.

WHAT IS AN SRB?

An SRB represents a routine that performs a particular function or service in a specified address space. It is similar to a TCB in the sense that it identifies a Unit Of Work (UOW) to the system.

As the name suggests, SRB routines are meant to provide specialized services. A program can initiate an SRB as a process in the same or a different address space. Unlike TCBs, SRBs cannot own storage areas, although the SRB routines can use areas owned by a TCB.

SCHEDULING AN SRB

Scheduling an SRB is nothing but initiating the process. Being asynchronous in nature, these routines run independently of the scheduling program.

Similarly to the ATTACH macro used for creating a TCB, a program can schedule an SRM process using the SCHEDULE or IEAMSCHD macro. Note that SRBs can be created only by units of work running in the supervisor state with key 0.

Before a program schedules an SRB, it must obtain 44 bytes of storage for the SRB and initialize its fields. This storage has to be freed when the SRB is no longer needed, either by the scheduling program or by the SRB.

The environment in which an SRB routine runs, and its ability to access address spaces and data spaces, can be controlled by

specifying appropriate parameters. For example:

- Specifying IEAMSCHD's ENV=FULLXM implies that the SRB receives control with the scheduling program's current cross-memory environment. This implies that when the SRB routine begins to run, it has the primary, home, and secondary address space of the scheduling program's at the time the IEAMSCHD macro was invoked.
- If the SRB is scheduled with MODE=FULLXM, you can free or reuse the SRB immediately after it has been scheduled.

Although SRB's advantage comes from its asynchronous nature, its processing can be synchronized with the scheduling program by using the WAIT/POST/SUSPEND/RESUME macros.

CHARACTERISTICS OF AN SRB

Being an independent UOW, SRBs have their own dispatching priority. SRBs with local priority take on the dispatching priority of the address space in which they are scheduled to run. SRBs with global priority can have a very high dispatching priority.

The following resources are associated with an SRB:

1 A Dispatchable Unit Access List (DU-AL).

2 A Functional Recovery Routine (FRR) stack.

3 A linkage stack.

4 A CPU timer value.

An SRB must return control to the address supplied in register 14, in supervisor state with no locks held, except the CPU lock.

An SRB in turn can issue a PC instruction and schedule another SRB. The transfer control macro, TCTL, allows an SRB routine to exit from its processing (does a clean-up of the SRB) and to pass control to a task with minimal system overhead. When SRB specifies RESUME RETURN=N, control transfers to the resumed TCB.

COMPARING IEAMSCHD WITH SCHEDULE

Usage of IEAMSCHD is recommended by IBM and it differs from the SCHEDULE macro as follows:

- It obtains storage for the SRB for the caller, and frees the storage when SRB processing is complete.
- The addresses of the Resource Manager Termination Routine (RMTR) and FRR are optional.
- It initializes SRB fields in IHASRB (the macro that maps the structure of an SRB) for the caller.
- In addition to local and global priorities, it also supports current, pre-emptable, client, and enclave priorities.

PURGING AN SRB

The SRB can be reused or freed. When an SRB is no longer required, it could be cleaned up using the PURGEDQ macro. A program such as an ESTAE routine or a resource manager can use the PURGEDQ macro to de-queue SRBs that are scheduled but not yet dispatched.

While purging, when you specify the primary address space, PURGEDQ's behaviour depends on the state of the SRBs:

1 Active – wait for previously-scheduled SRBs to complete processing.

2 Non-dispatched – it de-queues all non-dispatched SRBs. After all the SRBs have been de-queued or completed, the RMTR specified in the SRB is given control to perform the required clean-up for each de-queued SRB.

3 Suspended – it does not pass control to the RMTR of a suspended SRB. Instead, the system abnormally terminates those SRB routines and waits for the termination to complete.

The system automatically issues PURGEDQs at task and address space termination.

If an address space other than the primary address space is

specified, then PURGEDQ will try to purge SRBs that have not been dispatched. But it will not purge suspended SRBs, or wait for active or suspended SRB routines to complete processing.

SRB RESTRICTIONS

SRBs should not be long-running programs and usually represent very short pieces of work that complete much more quickly than TCBs. In line with this expectation, the system enforces the following restrictions on SRBs:

- An SRB routine is generally not pre-empted by I/O interruptions once it is dispatched. SRBs can also run at a pre-emptable priority, allowing work at an equal or higher priority to have access to the processor. Even if they are interrupted, control must be returned to them immediately after the interruption has been processed.
- To prevent them from going into a wait for any avoidable reason, the system restricts SRBs from issuing SVCs (except abends). This implies that they cannot issue some of the system macros like ENQs and data management functions like opening datasets.
- The SRB routine runs in the operating mode known as SRB mode. Code in SRB mode cannot leave supervisor state.
- SRBs can create, use, and delete data spaces, but cannot own a data space.
- An SRB cannot ‘own’ storage areas. SRB routines can obtain, reference, use, and free storage areas, but the areas must be owned by a TCB. Any data that the requesting task and the service share must be placed in common storage.
- SRBs can only GETMAIN storage in subpool 245 (SQA).
- The TCTL macro requires that the SRB requesting the TCTL must not hold any locks and must be in primary ASC mode, where the home and primary address space is the same.

WHY USE SRBS?

The independent and asynchronous nature of an SRB makes it quite useful and was well-exploited by IBM products. For example, CICS features like VTAM high-performance option, VSAM sub-tasking, was achieved by executing parts of the work in parallel under an MVS Service Request Block concurrent with the CICS TCB.

SRBs can be used for the following:

- Scheduling an SRB as an asynchronous process is an approach to making use of multiple virtual address spaces, confining errors (except ones pertaining to common storage) to one address space and improving system reliability.
- Parallel processing in a multi-processor environment:
  - The SRB routine can be dispatched to another processor and can run concurrently with the scheduling program.
  - It can run within the same cross-memory environment as a copy of the scheduling program (DU-AL) as it exists when the SCHEDULE command is issued. This allows the SRB routine and the scheduling program to access the same address and data spaces.
- To avoid serializing, because the scheduling program need not wait for the SRB routine to finish running. By using SRB, the delays like page fault resolution, address space swap-ins, and wait because of lock suspensions can be avoided.
- The priority of a process can be raised by using an SRB with a dispatching priority higher than that of the scheduling program.
- SRBs can be scheduled to carry out certain functions, say requesting a lock, that the running state of the scheduling program prohibits.
- A program running in cross-memory mode can schedule an SRB routine to perform functions that can be performed in non-cross-memory mode only.

- As SRBs can be scheduled to execute in another address space, they can be used as a mechanism by which one address space controls the execution of tasks in another address space.
- An SRB routine can be used to encrypt or decrypt data. Specifying the FEATURE=CRYPTO parameter ensures that it is run on a processor with an Integrated Cryptographic Feature (ICRF).
- As an SRB represents a separate UOW, the resources used can be accounted for separately – at address space or enclave level. This provides options for implementing a better charge-back mechanism.
- An SRB can be used to do post-dump processing for a program. In such cases, MVS can schedule the SRB after the capture phase or the write phase is complete. IBM recommends using the SRB parameter with CAPTURE instead of WRITE to avoid waiting for the write to complete and the dataset to become available.

PITFALLS TO BE AVOIDED

The following are some of the restrictions that need to be taken into account to avoid pitfalls in using SRBs:

- An SRB uses common storage for any data to be passed between SRBs and is therefore limited by the amount of common storage available. Also, placing the data in common storage necessarily makes it less secure than it would be in a private area. In such cases, the use of cross memory services – which is more flexible – is preferred.
- An enabled SRB routine can take page faults. If an SRB routine holds a suspend type lock when a page fault occurs, the suspended SRB routine continues to hold those locks until the system re-dispatches the SRB and the SRB routine explicitly releases the locks.
- As an SRB routine is dispatched after the program actually

issues the IEAMSCHD or the SCHEDULE macro, the conditions that existed in the system at the time the macro was issued can change by the time (eg task terminates) the SRB routine begins to run. If, in this time interval, the environment that the SRB routine needs to run successfully has changed, the results are unpredictable.

- As stated earlier, when you specify an address space other than the primary, PURGEDQ does not guarantee that all SRBs matching the purge parameters will be purged. It is suggested that PURGEDQ users who specify an address space other than primary must construct their own communication mechanism to be sure that an SRB has indeed been purged. You should use an RMTR to inform the issuer of PURGEDQ whether a particular SRB has been purged; otherwise, problems might result.

  One way to be sure that an SRB has been purged is to use an RMTR. When the PURGEDQ successfully finds and purges an SRB, the RMTR associated with that SRB is called. The RMTR can free the SRB, or the RMTR can use the SRB parameter area to indicate to another program (say the issuer of PURGEDQ) that the SRB can be safely freed.

- During address space termination, the recovery programs established by suspended SRB routines in that address space do not get control. Hence, the program that suspends those SRB routines should provide recovery for the SRB routine for the time the SRB routine is suspended. The resuming or purging program can use the RESMGR macro to establish a resource manager that gains control should the address space terminate. The resource manager must free any resources owned by the SRB routine and perform recovery, as needed.

CONCLUSION

Though cross memory services are better in terms of providing synchronous communication across address spaces, SRBs are

still in vogue and being used for asynchronous services in the system (a prelude for the much-hyped Web services!). A better understanding of the restrictions of SRB will help in making appropriate use of it.

Sasirekha Cota
Tata Consultancy Services (India)

DFSMSdss ENQ exit routine

INTRODUCTION

A common problem with shared DASD is managing DFSMSdss full dump with minimal impact on ENQ/RESERVE lockout.

During a standard full dump, DFSMSdss issues an ENQ/RESERVE macro during the entire 'read' of the volume, preventing other systems from accessing the disk for several minutes.

This has a potential impact on shared datasets' activity, which can affect MIM, SLS, RMM, or other system products whose control datasets are shared.

The DFSMSdss enqueue exit routine, ADRUENQ, allows DFSMSdss to enqueue the VTOC for only the read of the VTOC, not the entire read of the volume. In that configuration, the ENQ is held for only a few seconds.

This article explains how we have implemented this exit in our installation.

ADRUENQ INSTALLATION EXIT ROUTINE

To access a volume while it is being dumped, either by another job under the control of a second initiator or by another processor in a shared DASD environment, you can use the ADRUENQ exit routine to enqueue the VTOC only until it is processed.

This exit is called only for physical DUMP, COPY, or PRINT operations.

You can use this exit to prevent DFSMSdss from enqueueing the VTOC for a long period of time. By not enqueueing the VTOC, you can reduce the chances of a deadlock.

However, there is a trade-off – with the reduced chance of a deadlock there is also decreased data integrity.

By default, DFSMSdss enqueues the VTOC for the entire operation of a full or tracks COPY, a full DUMP, or a tracks PRINT.

The sample ADRUENQ routine provided by IBM changes the duration of the ENQ for all DUMP, COPY, and PRINT operations:

```
    ***************************************************
    * ADRUENQ USER EXIT.                              *
    *   SETS RETURN CODE TO 4 INDICATING THAT THE VOLUME *
    *   WILL ONLY BE ENQUEUED FOR THE DURATION OF THE    *
    *   VTOC ACCESS FOR DUMP AND COPY OPERATIONS.        *
    ***************************************************

    ADRUENQ  CSECT
    ADRUENQ  AMODE 31
    ADRUENQ  RMODE 24
              STM   14,12,12(13)          SAVE REGS IN PREVIOUS
SAVEAREA
              USING ADRUENQ,15            SET ADDRESSABILITY TO THE
EXIT
              LM    14,12,12(13)          RESTORE OTHER REGISTERS
              LA    15,4                  SET RETURN CODE TO 4
              BR    14                    RETURN
              END
```

This is a very global and basic approach, hence the reason we decided to write our own exit.

First, we wanted to use different default values for DUMP, COPY, and PRINT operations.

We decided to implement the following default ENQ duration:

- DUMP – short ENQ duration
- COPY – long ENQ duration

- PRINT – short ENQ duration.

In certain circumstances, we wanted to be able to override the default ENQ duration of the DFSMSdss operation. This is why we decided to implement the use of a dummy DD statement to override the default setting:

```
//VTOCENQL DD DUMMY             - to issue a LONG ENQ
//VTOCENQS DD DUMMY             - to issue a short ENQ
```

ADRUENQ installation

ADRUENQ is an installation exit routine: it is a replaceable module that modifies DFSMSdss system functions.

You should use SMP/E to link-edit the ADRDSSU module with your own version of ADRUENQ.

```
//ASSEM    EXEC PGM=ASMA90,PARM=('NODECK,OBJECT,XREF(SHORT)')
//SYSLIB   DD  DISP=SHR,DSN=SYS1.MODGEN
//         DD  DISP=SHR,DSN=SYS1.MACLIB
//SYSUT1   DD  DSN=&SYSUT1,SPACE=(1024,(120,120),,,ROUND),UNIT=SYSDA
//SYSPUNCH DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DISP=SHR,DSN=ZOSR14.FB80(ADRUENQ)
//SYSIN    DD  *
*******************************************************
* ADRUENQ USER EXIT.                                  *
*   SETS RETURN CODE TO 4 INDICATING THAT THE VOLUME *
*   WILL ONLY BE ENQUEUED FOR THE DURATION OF THE    *
*   VTOC ACCESS FOR DUMP AND COPY OPERATIONS.        *
*******************************************************
        TITLE 'DF/DSS EXIT, VTOC ENQ/DEQ FOR DFDSS'
ADRUENQ  CSECT
ADRUENQ  AMODE 31
ADRUENQ  RMODE ANY
*
* DESCRIPTION : THIS EXIT IS CALLED BY DFDSS TO DETERMINE WHETHER
* A VTOC ENQ SHOULD BE HELD FOR THE LIFE OF THE COMMAND (FULL
* VOLUME COPY & DUMP, TRACK PRINTGS, AND PHYSICAL
* DATASET DUMP) OR WHETHER THE VTOC ENQ SHOULD BE HELD ONLY
* WHILE THE VTOC IS BEING PROCESSED
*
* RETURN CODES:
*               0 - HOLD VTOC DURING ENTIRE OPERATION
*               4 - RELEASE VTOC AFTER PROCESSED
*
```

```
         STM R14,R12,12(R13)
         LR R12,R15
         USING ADRUENQ,R12
         B START
         DC CL8'ADRUENQ'
         DC C'&SYSDATE'
         DC C'&SYSTIME'
START    DS ØH
         LR R5,R1
         USING ADRUNQB,R5
         GETMAIN RC,LV=DSECTLEN,SP=Ø,LOC=ANY
         LTR R15,R15
         BNZ GETERROR
         ST R1,8(R13)               SAVE FORWARD CHAIN
         ST R13,4(,R1)              SAVE BACKWARD CHAIN
         LR R13,R1
         USING WORKAREA,R13         ADDRESSIBILITY TO WORK AREA
*
         WTO 'XXXXXXXX ADRUENQ EXIT:'
*
         XC FLAG,FLAG
         LA R2,DSSDEQ               GET DEQ DDNAME
         LA R3,DEVINFO              GET RETURN AREA
*
         DEVTYPE (R2),((R3),8)      CHECK FOR DDNAME
         C R15,=F'Ø'                DEQ FLAG SET?
         BNE NODEQ                  NO
         OI FLAG,X'8Ø'              SET FLAG TO SHORT ENQ
NODEQ    DS ØH
         LA R2,DSSENQ               GET ENQ DDNAME
         LA R3,DEVINFO
         DEVTYPE (R2),((R3),8)      CHECK AGAIN
         C R15,=F'Ø'                FIND IT?
         BNE NOENQ                  YES, CAUSE ENQ
         OI FLAG,X'4Ø'              SET FLAG TO LONG ENQ
*
NOENQ    DS ØH
*
         TM UNFLG1,UNDUMP           DUMP ?
         BZ NODUMP
*
         TM FLAG,X'4Ø'
         BZ DUMPMSG
         WTO 'XXXXXXXX OVERRIDE DUMP DEFAULT - USING LONG VTOC RESERVE'
         B ENDRCØ
*
DUMPMSG  DS ØH
         WTO 'XXXXXXXX USING DUMP DEFAULT- USING SHORT VTOC RESERVE'
         B ENDRC4
```

```
*
NODUMP   DS ØH
         TM UNFLG1,UNCOPY            COPY ?
         BZ NOCOPY
*
         TM FLAG,X'8Ø'
         BZ COPYMSG
         WTO 'XXXXXXXX OVERRIDE COPY DEFAULT - USING SHORT VTOC RESERVE+
               '
         B ENDRC4
*
COPYMSG  DS ØH
         WTO 'XXXXXXXX USING COPY DEFAULT - USING LONG VTOC RESERVE'
         B ENDRCØ
*
NOCOPY   DS ØH
         TM UNFLG1,UNPRINT           PRINT ?
         BZ BADFLAGS
*
         TM FLAG,X'4Ø'
         BZ PRTMSG
         WTO 'XXXXXXXX OVERRIDE PRINT DEFAULT - USING LONG VTOC RESERVE+
               '
         B ENDRCØ
*
PRTMSG DS ØH
         WTO 'XXXXXXXX USING PRINT DEFAULT - USING SHORT VTOC RESERVE'
         B ENDRC4
*
BADFLAGS DS ØH
         WTO 'XXXXXXXX UNKNOWN FUNCTION - USING LONG VTOC RESERVE'
         B ENDRC4
*
ENDRC4   DS ØH
         LA R6,4
         B EOJ
*
ENDRCØ   DS ØH
         LA R6,Ø
         B EOJ
*
EOJ      DS ØH
         LR R1,R13
         L R13,4(,R13)
         ST R15,16(,R13)
         FREEMAIN  RU,LV=DSECTLEN,SP=Ø,A=(1)
         LR R15,R6
         RETURN (14,12),RC=(15)
GETERROR DS ØH
```

```
         WTO 'XXXADRØ4 COULD NOT GETMAIN MEMORY, WILL NOT DO LONG RESERV+
               E'
          RETURN (14,12),RC=4
*
DSSDEQ   DC C'VTOCENQS'
DSSENQ   DC C'VTOCENQL'
*
         LTORG
WORKAREA DSECT
SAVEAREA DS 18F
RC       DS F
DEVINFO  DS 2F
FLAG     DS X
         DS ØD
DSECTLEN EQU *-WORKAREA
         ADRUNQB
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
/*
//SMP     EXEC PGM=GIMSMP,REGION=4M,
//        PARM='DATE=U,CSI=ZOSR14.GLOBAL.CSI'
//*
//SMPHOLD  DD  DUMMY
//SMPCNTL  DD  *
  SET BDY (GLOBAL).
  REJECT  SELECT(EXITØ57) BYPASS(APPLYCHECK) .
  RESETRC .

  RECEIVE SELECT(EXITØ57) .


  SET BDY (MVST1ØØ).
  APPLY SELECT(EXITØ57)
          REDO
          .
```

```
/*
//SMPPTFIN DD  *
++USERMOD(EXITØ57).
++VER(ZØ38) FMID(HDZ11GØ) PRE(UW884Ø3) .
++MOD(ADRUENQ) DISTLIB(AADRLIB) .
//      DD DISP=SHR,DSN=ZOSR14.FB8Ø(ADRUENQ)
```

ADRUENQ usage

Using default DUMP setting

This first example shows the result of a basic DASD full dump using the default ENQ setting:

```
//STEP1   EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//DASD     DD DISP=SHR,UNIT=SYSALLDA,VOL=SER=RES$Ø1
//BACKUP   DD DUMMY
//SYSIN    DD  *
  DUMP FULL INDD(DASD) OUTDD(BACKUP)
/*
```

Full DUMP SYSOUT using default settings:

```
$HASP373 SXSPØØ1D STARTED - WLM INIT  - SRVCLASS JES_3Ø   - SYS SMVS
IEF4Ø3I SXSPØØ1D - STARTED - TIME=1Ø.38.1Ø
TSS7ØØØI SXSPØØ1 Last-Used 18 Aug Ø3 1Ø:37 System=SMVS Facility=BATCH
TSS7ØØ1I Count=56926 Mode=Warn Locktime=None Name=******
XXXXXXXX ADRUENQ EXIT:                                                -
XXXXXXXX USING DUMP DEFAULT- USING SHORT VTOC RESERVE                 -
TSS7ØØØI SXSPØØ1 Last-Used 18 Aug Ø3 1Ø:38 System=SMVS Facility=BATCH
TSS7ØØ1I Count=56927 Mode=Warn Locktime=None Name=******
-                                        --TIMINGS (MINW.)--
-JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  ELAPS   SERV
-SXSPØØ1D STEP1                ØØ  45713    .Ø6    .Ø2    2.6   7Ø8K
IEF4Ø4I SXSPØØ1D - ENDED - TIME=1Ø.4Ø.48
-SXSPØØ1D ENDED.  NAME-WILFORD              TOTAL CPU TIME=   .Ø6
TOTAL
$HASP395 SXSPØØ1D ENDED
```

Overriding DUMP setting

In order to issue a long ENQ during the job, you should code a //VTOCENQL DD card to override the default setting:

```
//STEP1   EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//DASD     DD DISP=SHR,UNIT=SYSALLDA,VOL=SER=RES$Ø1
//BACKUP   DD DUMMY
```

```
//VTOCENQL DD DUMMY                                  - to issue a LONG ENQ
//SYSIN    DD  *
  DUMP FULL INDD(DASD) OUTDD(BACKUP)
/*
```

Full DUMP SYSOUT overriding default setting:

```
$HASP373 SXSP001D STARTED - WLM INIT  - SRVCLASS JES_30   - SYS SMVS
IEF403I SXSP001D - STARTED - TIME=10.38.10
TSS7000I SXSP001 Last-Used 18 Aug 03 10:37 System=SMVS Facility=BATCH
TSS7001I Count=56926 Mode=Warn Locktime=None Name=******
XXXXXXXX ADRUENQ EXIT:
XXXXXXXX OVERRIDE DUMP DEFAULT - USING LONG VTOC RESERVE
TSS7000I SXSP001 Last-Used 18 Aug 03 10:38 System=SMVS Facility=BATCH
TSS7001I Count=56927 Mode=Warn Locktime=None Name=******
-                                        --TIMINGS (MINW.)--
-JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  ELAPS   SERV
-SXSP001D STEP1                00  45713    .06    .02    2.6   708K
IEF404I SXSP001D - ENDED - TIME=10.40.48
-SXSP001D ENDED.  NAME-WILFORD               TOTAL CPU TIME=   .06
TOTAL
$HASP395 SXSP001D ENDED
```

BIBLIOGRAPHY

More information about ADRUENQ user exit can be found in *z/OS DFSMS Installation Exits* (SC26-7396).

Patrick Renard
Sytems Programmer (France)

Analysing HSM dump volumes

This utility is aimed at analysing the HSM LIST DVOL command output and providing the HSM administrator with explanations of the possible anomalies found on dump volumes.

In data centres where HSM plays an important role, ever-growing storage needs mean ever-growing back-up cartridges to cope with the huge back-up demand. HSM uses automatic dump processing to accomplish the back-up policy built for the data centre.

At any time, to display all HSM dump volumes, the command LIST DVOL is used. For big data centres, the output of this command can be very tough to analyse and interpret. The utility that I developed not only analyses this output, but it will also give some explanations on problematic dump volumes and overall statistics on dump volumes.

Here's an example of a problematic case. Let's imagine that the dump volume TTTTT1 has 50 disk back-ups (with the HSM stacking option) and let one of them be disk volume DDDD30. If for any reason this disk volume is removed from the system before the tape TTTTT1 expires, HSM will keep this tape forever because of the last copy feature. This tape will be protected and will cause us to have one less scratch tape. If a data centre has too many dump volumes, sometimes it's not that easy to catch this kind of dump volume by just looking through the original LIST DVOL listing.

From time to time we come across another problematic case where a dump volume (tape) is not associated with any disk volume. This sort of dump volume can be called an orphan dump volume. The utility easily detects dump volumes in this condition. In addition, the utility notifies the user by recommending him/her to use the DELVOL command to get rid of the problematic dump volume.

The utility will work for both 3490 and 3590 Magstar tapes. If a one-disk volume back-up spans into two 3490 dump volume tapes (multi-volume file structure), the utility will detect and report these tapes.

To customize the utility for your environment, all HSM dump classes and ML1 volume naming conventions should be entered in the utility. The utility consists of one JCL, one REXX EXEC, and one edit macro. To run the utility, it's necessary to put all three members into one library and then to make the SYSPROC DD statement refer to this library.

HSMJCL

```
//INSXØ141 JOB  CLASS=C,MSGCLASS=X,MSGLEVEL=(1,1),TIME=4Ø
//*-------------------------------------------------------------------*/
//* JOB-NAME    : Hsmjcl ØØ                                           */
//*-------------------------------------------------------------------*/
//STEP1    EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø,REGION=9ØM
//SYSPROC  DD   DISP=SHR,DSN=EXP.PSM.REXX
//         DD   DISP=SHR,DSN=SIS.EXEC
//ISPPLIB  DD   DISP=SHR,DSN=ISP.SISPPENU
//ISPMLIB  DD   DISP=SHR,DSN=ISP.SISPMENU
//ISPSLIB  DD   DISP=SHR,DSN=ISP.SISPSENU
//ISPTLIB  DD   DISP=SHR,DSN=ISP.SISPTENU
//ISPPROF  DD   DISP=(NEW,DELETE,DELETE),DSN=&&PROF,UNIT=SYSDA,
// DCB=(ISP.SISPTENU),SPACE=(TRK,(1,1,1))
//SYSUDUMP DD   DUMMY
//ISPLOG   DD   SYSOUT=(,),DCB=(LRECL=125,BLKSIZE=129,RECFM=VA)
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
ISPSTART CMD(%HsmRexx) +
BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(99999999)
//*
```

HSMREXX

```
/* REXX ---------------------------------------------------------------+
|  REXX-NAME : HsmRexx   (called by job Hsmjcl.)                      |
|                                                                     |
|  FUNCTION    :                                                      |
|  This REXX analyses HSM Dump tapes usage and reports all kind of    |
|  inconsistencies within them. The result is routed to the SYSTSPRT  |
|  sysout dataset of this job.                                        |
|                                                                     |
|  NOTES ON THE HSM CONFIGURATION:                                    |
|  1 - The library name is R2D2.                                      |
|  2 - The classes SEMANAL, MENSUAL and DIARIML1 are used for weekly, |
|      monthly, and daily dumps respectively.                         |
|  3 - ML1 disks naming convention = xHSyyy.                          |
+--------------------------------------------------------------------*/
"Prof Nopref"
Status = Msg('Off')
Dset_List_Dvol           = "Exp.Psm.Listdvol"
IF Sysdsn(Dset_List_Dvol) ='OK'  Then Delete Dset_List_Dvol
/*-------------------------------------------------------------------*/
/* Issue the "Hsm List Dvol" command  to list all dump tape volumes.  */
/*-------------------------------------------------------------------*/
"Hsend Wait List Dvol Ods("Dset_List_Dvol")"
/*-------------------------------------------------------------------*/
/* FROM THIS POINT ON, WE CAN ANALYSE THE "HSM LIST DVOL" COMMAND     */
```

```
/* OUTPUT.                                                          */
/*------------------------------------------------------------------*/
/*------------------------------------------------------------------*/
/* Remove the lines which include blanks and title information on the */
/* Dset_List_Dvol dataset, which contains details of all HSM Dump     */
/* tapes. This elimination is realized by the edit macro HSMMACRO.    */
/*                                                                  */
/* This will make our job easier when interpreting the vital records  */
/* in this dataset.                                                 */
/*------------------------------------------------------------------*/
"Ispexec Edit Dataset('"Dset_List_Dvol"') Macro(HSMMACRO)"
Rc2 = Rc
If Rc2 <> Ø Then Do
                    Say "An error occurred on editing the dataset,"
                    Say Dset_List_Dvol "Please check the existence "
                    Say "of it. Exiting..."
                    Exit
                End
/*------------------------------------------------------------------*/
/* Read all records in the formatted Dset_List_Dvol dataset and put   */
/* them into the stem variable Deniz.                               */
/*------------------------------------------------------------------*/
"Alloc Fi(Hsmout1)  Da("Dset_List_Dvol") Shr Reuse"
"Execio * Diskr "Hsmout1" (Stem Deniz. Finis"
Delete Dset_List_Dvol
/*------------------------------------------------------------------*/
/*                                                                  */
/* Note : If the "SET OF DUMP VOLSERS" field contains more than one   */
/* tape volser, then only the first one will be counted as the        */
/* unique volser for that group.                                    */
/*------------------------------------------------------------------*/
/*------------------------------------------------------------------*/
/* EXPLANATION ON SOME IMPORTANT VARIABLES:                         */
/* j = Line number of the HSM command output.                       */
/*------------------------------------------------------------------*/
                              /*-------------------------*/
j = 1                         /*                         */
r = Ø                         /*                         */
Dskdmp_num    = Ø             /*                         */
Dmptap_num    = Ø             /* INITIALIZATION OF       */
Dmp_ML1       = Ø             /*         VARIABLES       */
Dmp_Sem       = Ø             /*                         */
Dmp_Men       = Ø             /*                         */
Lib_Cnt       = Ø             /*                         */
NoLib_Cnt     = Ø             /*                         */
                              /*                         */
Dmp_SemL      = Ø             /*                         */
Dmp_MenL      = Ø             /*                         */
Dmp_ML1L      = Ø             /*                         */
                              /*                         */
```

```
Volstat_Avail = Ø              /*                        */
Volstat_Unava = Ø              /*                        */
Volstat_Expir = Ø              /*                        */
Volstat_Unexp = Ø              /*                        */
Volstat_Noret = Ø              /*                        */
                               /*------------------------*/
/*-----------------------------------------------------------------*/
/* Start interpreting the "Hsend List Dvol" HSM command.           */
/*-----------------------------------------------------------------*/
DO WHILE (j <= Deniz.Ø)           /*       BIG DO WHILE            */
      Flag = Ø
      Dmptap_num = Dmptap_num + 1
      Dumptape   = Substr(Deniz.j,2,6)
      Klas       = Substr(Deniz.j,42,8)  /* Dump tape's class name.   */
      Statv      = Substr(Deniz.j,9,5)   /* "VOL STATUS" field.       */
      Libr       = Substr(Deniz.j,85,8)  /* Location of the dmp tape; */
                                         /* in library or out of lib. */
      /*-------------------------------------------------------------*/
      /* Distribution of the HSM Dump Volumes by their               */
      /* "Volume Status", such as AVAIL / UNAVA / EXPIR / UNEXP.     */
      /*-------------------------------------------------------------*/
      Select
        When Statv = 'AVAIL' Then Volstat_Avail = Volstat_Avail + 1
        When Statv = 'UNAVA' Then Volstat_Unava = Volstat_Unava + 1
        When Statv = 'EXPIR' Then Volstat_Expir = Volstat_Expir + 1
        When Statv = 'UNEXP' Then Volstat_Unexp = Volstat_Unexp + 1
        When Statv = 'NORET' Then Volstat_Noret = Volstat_Noret + 1
        Otherwise
      End
      /*-------------------------------------------------------------*/
      /* Distribution of HSM Dump Volumes by their location.         */
      /* ( Are they in the library or out of the library? )          */
      /*-------------------------------------------------------------*/
      Select
        When Libr = 'R2D2    ' Then Lib_Cnt   = Lib_Cnt   + 1
        When Libr = '*NO LIB*' Then Nolib_Cnt = Nolib_Cnt + 1
        Otherwise    Say "Define the ibrary "Libr" in the HsmRexx."
      End
      /*-------------------------------------------------------------*/
      /* Distribution of the HSM Dump Volumes by their Dump Classes. */
      /*-------------------------------------------------------------*/
      Select
        When  Klas = 'SEMANAL ' Then
                   Do
                   Dmp_Sem = Dmp_Sem + 1
                   If Libr = 'R2D2    ' Then Dmp_SemL = Dmp_SemL + 1
                   End
        When  Klas = 'MENSUAL ' Then
                   Do
                   Dmp_Men = Dmp_Men + 1
```

```
                If Libr = 'R2D2    ' Then Dmp_MenL = Dmp_MenL + 1
                End
        When  Klas = 'DIARIML1' Then
                Do
                Dmp_Ml1 = Dmp_Ml1 + 1
                If Libr = 'R2D2    ' Then Dmp_Ml1L = Dmp_Ml1L + 1
                End
        Otherwise    Say "Define Dump class" Klas "in the HsmRexx."
      End
Say "TAPE:" Dmptap_num "-)" Dumptape
             /*---------------------------------------*/
m = j  + 1   /* Next line of the 'Tape Line'.         */
n = j        /* Current line.                         */
J = j  + 1   /* Line number of the HSM command output. */
             /*---------------------------------------*/
/*-------------------------------------------------------------------*/
/* k = Counter in one tape, ie file number in a tape. At the same    */
/* time it is the number of disk dumps in an HSM Dump volume.        */
/*-------------------------------------------------------------------*/
k = 1
/*-------------------------------------------------------------------*/
/* EXTRACTING MULTI-FILE HSM DUMP TAPES (THAT ARE CREATED THROUGH    */
/* STACK OPTION)                                                     */
/* Example:                                                          */
/*  DUMP   VOL    UNIT     FILE  SOURCE              DUMPED          */
/*  VOLSER STATUS TYPE     SEQ   VOLSER SMS CLASS    DATE            */
/*  N5Ø233 UNEXP  359Ø-1                   SEMANAL                  */
/*                        Ø1    DNZGØ1 N            2ØØ1/12/16      */
/*                        Ø2    DNZGØ2 N            2ØØ1/12/16      */
/*                        Ø3    DNZGØ3 N            2ØØ1/12/16      */
/*-------------------------------------------------------------------*/
   Do While (Substr(Deniz.j,42,8) <>  'SEMANAL ' &, /* Do While - XXX*/
             Substr(Deniz.j,42,8) <>  'MENSUAL ' &,
             Substr(Deniz.j,2,23)    = '                      ')
    /*---------------------------------------------------------------*/
    /* Klas = Dump class of the dump volume.                         */
    /*---------------------------------------------------------------*/
    Klas  = Substr(Deniz.n,42,8)
    Pepet = Substr(Deniz.j,31,6)
    /*---------------------------------------------------------------*/
    /* Lab1 = File sequence number. This variable contains the number */
    /* of the file containing the Dump copy.                         */
    /*---------------------------------------------------------------*/
    Lab1  = Substr(Deniz.j,25,2)
    /*---------------------------------------------------------------*/
    /* Volv=First volser that appears on "Set of dump volser field."  */
    /*---------------------------------------------------------------*/
    Volv  = Substr(Deniz.m,1ØØ,6)
    /*---------------------------------------------------------------*/
    /* Sodv = "Set of dump volser" field on the DUMP CONTENTS listing.*/
```

```
    /* This variable holds the list of the volume serial numbers for  */
    /* tape volumes used to dump the source volume.                   */
    /*----------------------------------------------------------------*/
    Sodv = Substr(Deniz.m,100)
     Select
      When  Klas     = DIARIML1        Then Call Msgs1
      When  Dumptape <> Volv           Then Call Msgs3
      When  (Substr(Pepet,2,2) = 'HS') Then Call Msgs2
      Otherwise
       Do
    Say "Comb." k "-)" Pepet Dumptape "combination.-" KLAS "Label= "Lab1
          If (k =  1) & (k <> Lab1) Then Call Msgs4
          If (k <> 1) & (k <> Lab1) & (Flag = 0) Then Call Msgs5
          /*----------------------------------------------------------*/
          /* In case a multi-volume condition. (If a disk dump spans  */
          /*  to 2 or more tape volumes.)                             */
          /*----------------------------------------------------------*/
          If Substr(Sodv,8,1) <> ' ' Then Call Msgs7
         r  = r  + 1
       End
     End
      j = j + 1
      k = k + 1
  End  /* Do While - XXX */
  If (Substr(Deniz.j,42,8) = 'SEMANAL ' |,
      Substr(Deniz.j,42,8) = 'MENSUAL ' |,
      Substr(Deniz.j,2,23) <>  '                        ')
      Then If (K=1  & Substr(Sodv,8,1) = " ") Then Call Msgs6
Say " "
Dskdmp_num = Dskdmp_num + k - 1
END                          /*       END of BIG DO WHILE              */
/*====================================================================*/
   Say " "
   Say "Total number of disk dumps taken by HSM     = " Dskdmp_num
   Say "Total number of ML1        Dump volumes     = " Dmp_ML1
   Say "Total number of Weekly     Dump volumes     = " Dmp_Sem
   Say "Total number of Monthly    Dump volumes     = " Dmp_Men
   Say "-------------------------------------------------------"
   Say "Total number of HSM Dump volumes used       = " Dmptap_num
   Say " "
   Say "Number of Dump volumes in library R2D2      = " Lib_Cnt
   Say "Number of Dump volumes out of library R2D2  = " Nolib_Cnt
   Say " "
   Say "Number of tapes in library R2D2 (Weekly)    = " Dmp_SemL
   Say "Number of tapes in library R2D2 (Monthly)   = " Dmp_MenL
   Say "Number of tapes in library R2D2 (DiariML1)  = " Dmp_Ml1L
   Say " "
   Say "Number of tapes that have a status of AVAIL = " Volstat_Avail
   Say "Number of tapes that have a status of UNAVA = " Volstat_Unava
   Say "Number of tapes that have a status of EXPIR = " Volstat_Expir
```

```
   Say "Number of tapes that have a status of UNEXP = " Volstat_Unexp
   Say "Number of tapes that have a status of NORET = " Volstat_Noret
/*-------------------------------------------------------------------*/
/* Add a null line to indicate the end of the stack.                 */
/*-------------------------------------------------------------------*/
Queue ''
r = r +1
EXIT     /* End-of-HsmRexx   */
/*********************************************************************/
/*-------------------------------------------------------------------*/
MSGS1:
Say "Comb." k "-) "Dumptape"-"Pepet "combination (ML1 dump tape).
Class="Klas
RETURN
/*-------------------------------------------------------------------*/
/*-------------------------------------------------------------------*/
MSGS2:
Say "Comb." k "-) "Dumptape"-"Pepet "combination (ML1 volume) "
RETURN
/*-------------------------------------------------------------------*/
/*-------------------------------------------------------------------*/
MSGS3:
Say "Tape " Dumptape " is part of a multi-volume dump. Class="Klas
Say "But this tape is the second tape of the related dump.   "
k=k-1             /* No need to count tape into Dskdmp_num.  */
RETURN
/*-------------------------------------------------------------------*/
/*-------------------------------------------------------------------*/
MSGS4:
Say "WARNING : The tape "Dumptape" has lastdumpcopy of some disk "
Say "volumes. If the dumps taken on the tape are not to be used, "
Say "delete the dump volume with the DELVOL command.             "
RETURN
/*-------------------------------------------------------------------*/
/*-------------------------------------------------------------------*/
MSGS5:
Say "WARNING : The tape "Dumptape" has some missing labels. There may"
Say "have been some disk volumes that were not dumped on this dump   "
Say "volume.                                                         "
Flag  = 1 /* Do not repeat this warning for the same tape again. */
RETURN
/*-------------------------------------------------------------------*/
/*-------------------------------------------------------------------*/
MSGS6:
Say "WARNING : This dump volume has no any valid dumps associated."
Say " You can delete it with DELVOL command.                      "
RETURN
/*-------------------------------------------------------------------*/
/*-------------------------------------------------------------------*/
MSGS7:
```

```
Say "Note: Set of Dump Volsers for this disk dump is:" Sod
v
RETURN
/*--------------------------------------------------------------------*/
```

HSMMACRO

```
/* REXX */
"Isredit Macro"
/*--------------------------------------------------------------------*/
/* MACRO_NAME  : Hsmmacro    CALLED-BY   : HsmRexx                    */
/*                                                                    */
/* FUNCTION    : Remove unnecessary lines from "Hsend List Dvol"      */
/*               command output.                                      */
/*--------------------------------------------------------------------*/
"Isredit Exclude 'DFSMSHSM CONTROL DATASET'         ALL"
"Isredit Exclude 'DUMP   VOL    UNIT     FILE'      ALL"
"Isredit Exclude 'VOLSER STATUS TYPE     SEQ'       ALL"
"Isredit Exclude 'END OF - DUMP VOLUME - LISTING'   ALL"
"Isredit Exclude '                                          ' ALL"
/*--------------------------------------------------------------------*/
/* Delete all excluded lines and save the dataset.                  */
/*--------------------------------------------------------------------*/
"Isredit Delete ALL X"
"Isredit Save"
"Isredit Cancel"
Return
```

Atalay Gul
Systems Programmer
Azertia SA (Spain)

Java for OS/390 problem determination tips and diagnostic and performance monitoring tools

This is a follow-up to an article entitled *A fresh look at Java for OS/390 – enhancements in functionality, scalability, and performance*, published in *MVS Update*, issue 205, October 2003.

It can be a challenge to debug Java applications on the OS/390 platform, but a number of enhancements have been made in this area and more IBM and third-party products are emerging to assist with Java problem determination.

Some of the most common problems with Java applications is that they have a large footprint, and may over-exhaust virtual and/or real storage allocation. So, let's take a look at issues related to storage management, using WebSphere for z/OS as an example of the Java application running on OS/390.

VIRTUAL STORAGE

Ensure that you do not underestimate the amount of virtual storage needed. Generally, Java applications use significantly more virtual memory than traditional application servers on z/OS or OS/390. Place run-time modules in the Link Pack Area (LPA) whenever possible.

The setting of REGION on the JCL for the start-up procedures should be large and the recommendation is to run with a region size of 0MB to allocate the largest possible region size allowed by the installation. Applications like WebSphere require at least 384MB to run, and much more if high throughput is required.

REAL STORAGE

Since the real storage is needed to back the virtual storage, its usage is also high. Expect a requirement of at least 512MB of real storage for a small configuration running WebSphere. The control regions utilize around 18MB of that real storage. The amount of real storage utilized in the server regions is dependent on the size of the JVM heapsize.

HEAPSIZE

JVM heapsize is controlled by the JVM_HEAPSIZE JVM environment variable set in the current.env file.

If the HEAPSIZE is too large, there will be longer delays when garbage collection is eventually done. It also increases the amount of system resources needed for each of the application servers that can be started by WLM. Since WLM will start multiple application servers as the workload increases, there is

no need to run with an extra large HEAPSIZE. It should be set high enough to support the running of the largest application(s). If the HEAPSIZE is too small, applications may fail because of an out-of-memory condition, or more frequent garbage collection may cause a great deal of CPU overhead.

Set JVM_ENABLE_VERBOSE_GC = 1 for a garbage collection report, which you can use to determine how big the HEAPSIZE should be. The JVM_HEAPSIZE should be set to 512MB for a high-volume WebSphere application and should be equal to the JVM_MINHEAPSIZE so that garbage collection is done only when the heap is full. Make sure that the region size for each application server is large enough to hold the HEAPSIZE.

UNDERSTANDING GARBAGE COLLECTION

Prior to SDK 1.3.1, the OS/390 JVM had its own garbage collector, which ran fully concurrently with the running threads. In SDK 1.3.1, the OS/390 JVM went to a stop-the-world collector and became the first platform with persistent reusable JVM support. The SDK 1.3.1 garbage collector has parallel mark and a serial bitwise sweep, to avoid inspecting individual dead objects for possible collection. It uses compaction avoidance to eliminate, as far as possible, the pause times associated with compaction.

The verbose:gc command, introduced in SDK 1.3.1, prints garbage collection information. Review the output from the last garbage collection, written to the JVM stderr output file, to determine the high-water marks for the heap used.

The most straightforward, and often most useful, way of monitoring the Java heap or debugging the Java out-of-memory condition is by seeing what garbage collection is doing.

The garbage collector is the JVM memory manager and is therefore responsible for allocating memory in addition to collecting garbage. Because the task of memory allocation is small, compared with that of garbage collection, the term ‘garbage collection’ usually also means memory management.

The Java heap becomes exhausted when garbage collection cannot free enough objects to make a new object allocation. Garbage collection can free only objects that are no longer referenced by other objects, or are referenced from the thread stacks. Java heap exhaustion can be identified from the -Xverbosegc output by garbage collection occurring more and more frequently, with less memory being freed. Eventually the JVM will fail, and the 'totally out of heapspace' message can be seen. If the Java heap is being exhausted, and increasing the Java heap size does not solve the problem, the next stage is to examine the objects that are on the heap, and look for suspect data structures that are referencing large numbers of Java objects.

The GC report shows how much live data is left on the heap at the end of a collection cycle. If you have an object leak then, over time, you might see an increase in the amount of live data remaining after the cycle has completed. It's very important that the GC cycle is fast and has no impact on performance. The verbose:gc output includes timing data, so you can check that the whole cycle is not taking much longer than expected.

Here is an example GC report:

```
<AF-54x: Allocation Failure. need 959912 bytes, 331249 ms since last AF>
<AF-54x: managing allocation failure, action=2 (20458672/536803840)>
  <GC(54): mark stack overflow-35x>
<GC(54): GC cycle started Thu Aug 21 09:17:09 2003
<GC(54): freed 427859296 bytes, 83% free (448317968/536803840), in 8364
ms>
  <GC(54): mark: 1746 ms, sweep: 46 ms, compact: 6572 ms>
  <GC(54): refs: soft 0 (age >= 32), weak 1, final 2822, phantom 0>
  <GC(54): moved 1041567 objects, 82299480 bytes, reason=1, used 1576
more bytes
<AFY54x: completed in 8364 ms>
```

The report above shows full allocation failure, where 959,912 bytes were requested. This is the size of the object or array that could not be allocated on the heap that caused the collection cycle to begin. In this case you can see that the heap is allocated at 512MB and over 400MB was freed. This report also shows the times taken for the mark, sweep, and compact, as a part of this

cycle. The total duration for the cycle, including the expansion, was 8,364 ms.

The action fields, in order of severity, are:

1 A pre-emptive garbage collection cycle

2 Full allocation failure

3 A heap expansion takes place

4 All known soft references are cleared

5 Stealing from the transient heap is done

6 Free space is very low.

For normal operation actions 4, 5, and 6 imply that the heap size is not adequate for the application.

RECEIVING OUTOFMEMORY ERRORS

The JVM throws a java.lang.OutOfMemory exception when the heap is full and is unable to find space for object creation. For those from a OS/390 background, this is analogous to the abend 878 rc10 issued for lack of private storage, and, similarly, is not necessarily a problem with the JVM itself. Heap utilization is a consequence of the application design, its use and creation of object populations, and the interaction between this and the garbage collector, which on OS/390 operates concurrently with the application.

Any OutOfMemory condition that occurs could be caused by running out of either Java heap or native heap. In either case it is entirely possible that there is not a memory leak as such, just that the steady state of memory usage required is higher than that available. Therefore the first step is to determine which heap is being exhausted and then increase the size of that heap. If the problem is occurring because of a real memory leak, increasing the heap size will not solve the problem, but will delay the onset of the OutOfMemory conditions.

To help debug OutOfMemory conditions, it may help to set the

envvar **JAVA_DUMP_HEAP=true.** When this envvar is set, records describing the current state of the heap are written to the JAVATRACE file when a java.lang.OutOfMemory exception is thrown.

Here is an example of Out of Memory errors under CICS TS 2.2, written to dfhjvmerr log:

```
JVMDG200: Diagnostics system property ibm.jvm.events.output=event.log
JVMST080: -verbose:gc flag is set
JVMST082: -verbose:gc output will be written to stderr

**Out of memory, aborting**

*** panic: JVMST017: Cannot allocate memory in
initializeMarkAndAllocBits(markbi

<AFY1¨: Allocation Failure. need 16400 bytes, 0 ms since last AF>

**Out of memory, aborting**

*** panic: JVMST018: Cannot allocate memory for
initializeMarkAndAllocBits(alloc

Unable to allocate an initial java heap of 35651584 bytes.
**Out of memory, aborting**

*** panic: JVMST016: Cannot allocate memory for initial java heap
```

MEMORY LEAKS

Frequent OutofMemory conditions, with proper heap size allocation, are most likely caused by memory leaks. This condition occurs when references to unused objects are not released. Sometimes applications cause the heap to grow by retaining references to objects they no longer use. The best way to debug problems like this is with dedicated J2SE or J2EE profiling tools, discussed later in this article. This will allow you to understand the object graphs created during application execution. If you don't have access to these types of tool you can use the following functionality, which is built into the JVM.

The heapdump mechanism for the z/OS JVMs is different from that of the distributed platforms. The libhprof.so profiling library

that is available as part of the JVM has been modified to support a new option. This creates a dump of all the live objects that are on the Java heap when an OutOfMemoryError occurs. You can use the combination of the binary heapdump that is produced by the libhprof.so and the post-processing tool FindRoots to identify objects or collections of objects that are using large amounts of memory on the Java heap. This is useful in cases of Java memory leaks.

FINDROOTS

IBM uses FindRoots for analysing these binary dump files. It is available internally from IBM javaserv at ftp://javaserv.hursley.ibm.com/pmrs/cm131s/FindRoots.java.

FindRoots is supplied in file svcdump.jar. The file can be obtained on request from Java L2 or L3 support. The FindRoots tool was initially made available to help progress issues where application object leaks occurred, causing an OutOfMemoryError to be thrown because the garbage collector was unable to free up enough space on the Java-managed heap to allow an allocation of an object or array on the heap to proceed.

The FindRoots tool has been enhanced to support direct reading of SVC dumps, without the need for intermediate software like IPCS.

CONSOLE DUMP

For z/OS customers it is often easier to take a console dump, which can be taken as a result of poor performance, a hang, or a loop condition.

To take a console dump, perform the following:

1 Use the operating system commands (D OMVS,A=ALL) or SDSF (DA = Display Active) to locate the ASID of interest.

2 Use the DUMP command to take a console-initiated dump:

```
DUMP COMM=(Put a dump title here)
```

```
R xx,JOBNAME=(Webserv,OMVS),DSPNAME=('OMVS '.*),CONT
R yy,(SDATA=(GRSQ,LSQA,RGN,SUM,SWA,TRT,LPA,NUC,SQA)
```

When the console dump has been generated, you can view the systrace in IPCS and look for system trace entries for all threads that are associated with the ASID of interest.

JVM DUMPS

There are also a number of JVM dumps that can be system-generated in response to specific errors or events, depending on the setting of the environment variable JAVA_DUMP_OPTS.

The full syntax for JAVA_DUMP_OPTS on z/OS is:

```
JAVA_DUMP_OPTS="ONcondition(dumptype,dumptype),ONcondition
(dumpytpe,...),...),USERABEND(nnnn),ceedumpoptions"
```

where *dumptype* can be ALL, NONE, JAVADUMP, SYSDUMP, or CEEDUMP.

If USERABEND is set, it must specify an integer in the range 1 to 4094.

These events are grouped as follows:

- EXCEPTION – unexpected synchronous terminating signal; that is, unrecoverable storage violation.
- ERROR – controlled abort because of an internally-detected error; for example, no more memory is available.
- INTERRUPT – asynchronous terminating signal.

The types of dump that can be produced are:

1 SYSDUMP or SYSMDUMP – an unformatted dump that the operating system generated.

2 JAVADUMP – an internally generated and formatted analysis of the JVM. You can read system dumps by using native dump analysis tools like IPCS.

3 CEEDUMP – an LE CEEDUMP is produced for the relevant conditions, after any SYSDUMP processing, but before a

JAVADUMP is produced. A CEEDUMP is a formatted summary system dump that shows stack traces for each thread that is in the JVM process, together with register information and a short dump of storage pertaining to each register. The default options for CEEDUMP are:

```
THREAD(ALL),PAGESIZE(Ø),ENC(CUR),NOENTRY,GENO
```

Additional or overriding options can be appended to the JAVA_DUMP_OPTS string.

These three documents provide the ability to determine the failing function, and therefore decide which product owns the failing code, be it the JVM, application JNI code, or third-party native libraries.

Dumps are produced in the following form:

- SYSMDUMP (also known as TDUMP) is a standard MVS dataset, using the default name in the form:

  ```
  &userid.SYSTDUMP.&date.T&time
  ```

- CEEDUMP – in the current directory, or as determined by the setting of _CEE_DMPTARG as:

  ```
  CEEDUMP.&date.&time.&processid
  ```

- JAVADUMP – in the same directory as CEEDUMP, or standard JAVADUMP directory as:

  ```
  JAVADUMP.&date.&time.&processid.txt
  ```

You can use the following JVM variables to control TDUMP's creation:

- IBM_JAVA_ZOS_TDUMP – this environment variable controls the TDUMP when the JVM detects a problem. Setting this environment variable to NO prevents that from happening. For more complete information, see information APAR II13292.
- IBM_JAVA_ZOS_TDUMP_PATTERN – this environment variable is set to a string, which is a pattern for the dump dataset name in case the default name of the TDUMP

dataset does not conform to your shop standards.

For example:

```
IBM_JAVA_ZOS_TDUMP_PATTERN=DUMP.MVS.&JOBNAME..TDUMP.T&HHMMSS
```

- IBM_JAVA_ZOS_TDUMP_COUNT – this environment variable limits the number of TDUMPs that are requested. For example, if an error occurs during CEEDUMP processing, it could result in two dumps. The default for this value is 2, to allow for the diagnosis of those cases where a problem occurs in CEEDUMP or JAVADUMP processing.

Be aware that some customers have reported problems in analysing the TDUMP using IPCS and were unable to format the system trace using the ip systrace command. IBM has recognized it as a bug and has the following fixes to address it: UW89731 – R608 (OS/390 2.8.0), UW89732 – R703 (z/OS 1.1), UW89733 – R705 (z/OS 1.2), UW89734 – R706 (z/OS 1.3), UW89735 – R707 (z/OS 1.4).

When the TDUMP fails, you'll see a message on the console (msgIEA820I) and the return code and reason code are externalized by the JVM in message JVMHP004.

For example (in syslog):

```
IEA82ØI TRANSACTION DUMP REQUESTED BUT NOT TAKEN
  DUMP DATA SET NAME NOT VALID
```

USING IPCS COMMANDS

The most practical way to find where the exception occurred is to review either the CEEDUMP or the Javadump. Both of these show where the exception occurred and the native stack trace for the failing thread. The same information can be obtained from the transaction dump by using either the IPCS LEDATA VERB exit or the svcdump.jar toolset. These generate a report that is similar to the CEEDUMP.

Here are some sample IPCS commands that you might find useful during your debugging sessions. In the examples below,

the address space of interest is ASID(X'1D').

```
ip verbx ledata 'nthreads(*)'
```

This command formats all the C-stacks (DSAs) for threads in the process that is the default ASID for the dump.

```
ip setd asid(x'001d')
```

This command is to set the default ASID.

```
ip verbx ledata 'all,asid(001d),tcb(tttttt)'
```

In this command, the *all* report formats key LE control blocks such as CAA, PCB, ZMCH, CIB. In particular, the CIB/ZMCH captures the PSW and GPRs at the time the program check occurred.

```
ip verbx ledata 'cee,asid(001d),tcb(tttttt)'
```

This command formats the traceback for one specific thread.

```
ip summ regs asid(x'001d')
```

Next, issue **find 'slip regs sa'**, to locate the GPRs and PSW at the time a SLIP TRAP is matched. This command is useful in the case of setting an SA (Storage Alter) trap to catch an overlay of storage.

```
ip omvsdata process detail asid(x'001d')
```

This command generates a list of the address spaces in the system at the time of the dump, so you can tie up the ASID with the JOBNAME.

```
ip systrace asid(x'001d') time(gmt)
```

This command formats the system trace entries for all threads in this address space. It is useful for diagnosing loops. *time(gmt)* converts the TOD clock entries in the system trace to a human-readable form.

DEBUGGING JAVA PROGRAMS ON OS/390

There are a number of ways to debug your Java programs on OS/390. One way is to use jdb debugger, which allows you to debug

OS/390 Java bytecode from OS/390 USS. Using this option, you need only Java for OS/390. You do not need any other software.

Under the USS itself, you can use the jdb command, which comes with the Java Development Kit as part of Java for OS/390. Type jdb under the shell to start the debugger and then type help for the full list of commands. The jdb command is a command-line debugging tool, analogous to dbx, though not nearly as powerful. If you're compiling from the command line using the javac command, be sure to specify the -g option to generate the debugging information needed for jdb and the other debuggers.

Jdb is the command line debugger distributed with the J2SE SDK V1.3 and later releases.

The jdb sample can be run by executing:

```
java tty.TTY <options>.. <class-name>
```

where <*class-name*> is the name you would normally place on the Java command line. The *-help* option provides information on options.

JAVA PLATFORM DEBUGGING ARCHITECTURE – JPDA

Another way to debug Java applications on OS/390 is by making use of a standard called Java Platform Debugging Architecture (JPDA). JPDA is included in the Java 2 platform, Standard Edition (J2SE) SDK 1.3 and later, on all platforms – and OS/390 and z/OS are no exception.

JPDA consists of the following layered APIs:

- Java Debug Interface (JDI) – this is a Java programming language interface providing support for remote debugging. This is the highest-level interface in the architecture, and can be used to implement a remote debugger user interface without having to write any code that runs in the application JVM or understands the protocol between the debugger and the JVM. Most third-party debuggers that support JPDA currently use this API.

- Java Debug Wire Protocol (JDWP) – this API defines the format of the flows that run between the application JVM and the debugger user interface. This protocol is for use by debuggers that need to exploit the communication at a lower level than the JDI, and for JVM suppliers or more advanced debugger developers who need to support the standard connection architecture from the application JVM side.
- Java Virtual Machine Debug Interface (JVMDI) – this is a low-level native interface within the JVM. It defines the services a Java virtual machine must provide for debugging, and can be used by advanced debugger developers who wish to implement debugger code that runs inside the application JVM.

  When you start the JVM in debug mode, the JVMDI interface is activated and additional threads are started in the JVM. One of these threads handles communication with the remote debugger, the others monitor the application that is running in the JVM.

To make use of JPDA you will need to set the following JVM debugging options:

- Xdebug=YES (NO is the default) – specifies whether or not debugging support is to be enabled in the JVM.
- Xrunjdwp=(*suboption=string*,*suboption=string*...) – this option loads the JPDA reference implementation in-process debugging libraries and passes any -Xrunjdwp sub-options specified. This library resides in the target VM and uses Java Virtual Machine debug interface (JVMDI) and the Java Native Interface (JNI) to interact with it. It uses a transport and the Java Debug Wire Protocol (JDWP) to communicate with a separate debugger application.

For more information on JPDA please see the Java Debugger (JDB) description in the Java 2 specification at http://java.sun.com/products/jpda/doc/.

JVM TRACE

In some cases you may want to choose to use JVM trace for problem determination. JVM trace is an extremely effective tool and can be a key diagnostic tool for JVM debugging.

Tracing is switched off by default. Command-line options enable trace to be turned on or off and the trace level set. There are ten levels of trace. You can also redirect trace output to a file.

Trace can be very useful when debugging a crash, hang, or memory leak problem, as it provides the state of JVM before the problem occurred.

JVM trace is a low-overhead trace facility provided in all IBM-supplied JVMs. In most cases, the trace data is kept in compact binary format, with variable-length trace records from 8 to 64KB. A cross-platform Java formatter is supplied to format the trace. You can enable tracepoints at run-time by using levels, components, group names, or individual tracepoint identifiers.

You can trace entry to and exit from methods for selected classes. Using the ibm.dg.trc.methods property, you can select method trace by class, method name, or both. Wildcards can be used, and a ‘not’ operator is provided to allow for complex selection criteria. Note that this property selects only the methods that are to be traced. The MT trace component must be selected for a given trace destination. For example:

```
-Dibm.dg.trc.methods=*.*,!java/lang/*.*
-Dibm.dg.trc.print=mt
```

This routes method trace to stderr for all methods for all classes except those that start with java/lang.

The trace formatter is a Java program that runs on any platform and can format a trace file from any platform. The formatter, which is shipped with the SDK in rt.jar, also requires a file called TraceFormat.dat, which contains the formatting templates. This file is shipped in jre/lib.

JAVA VIRTUAL MACHINE PROFILER INTERFACE – JVMPI

Another IBM native tool for JVM problem determination is JVM Profiler Interface (JVMPI), which was introduced in Java 2 SDK. It is evolving towards a Java standard that standardizes a way for any profiling tool to talk to any running JVM to collect statistics about it.

JVMPI tools conform to the JVM Profiling Interface that is common across all JVMs. The IBM JVM is fully JVMPI compatible. Any tool conforming to JVMPI can be used to profile the IBM JVM. JVMPI tools help with problems involving leaks and performance, although profile logs might give useful hints to the state of the JVM just before a crash or hang problem. The JVMPI is intended for interested parties to write profilers, but IBM provides a useful agent with the IBM SDK.

The JVMPI is a two-way function call interface between the Java virtual machine and an in-process profiler agent. On one hand, the virtual machine notifies the profiler agent of various events, corresponding to, for example, heap allocation, thread start, etc. On the other hand, the profiler agent issues controls and requests for more information through the JVMPI. For example, the profiler agent can turn on/off a specific event notification, based on the needs of the profiler front-end.

A profiling tool based on JVMPI can obtain a variety of information such as heavy memory allocation sites, CPU usage hot-spots, unnecessary object retention, and monitor contention, for a comprehensive performance analysis.

This allows a tool like WebSphere Studio Application Developer, or any tool that includes some performance analysing software, to set the amount of data to collect, create reports, integrate with monitoring tools, and coordinate activities of multiple programs, possibly on multiple machines, without worrying too much about the target run-time.

The user can specify the name of the profiler agent and the options to the profiler agent through a command line option to the Java virtual machine. For example:

```
java -Xrunmyprofiler:heapdump=on,file=log.txt ToBeProfiledClass
```

JAVA DIAGNOSTIC TOOLS

A number of JVM diagnostic and performance monitoring tools are available on the OS/390 platform that can help you fix the problem quickly or allow you to take proactive action before before the customers feel any effects.

Below you will find an overview of the tools.

IBM WebSphere Studio

WebSphere Studio has a number of cutting-edge technology products that can be invaluable tools for Java problem determination and performance monitoring. Let's take a brief look at what IBM has to offer.

WebSphere Studio Application Developer is the follow-on product for VisualAge for Java, Enterprise Edition. It is designed from the ground up to meet the requirements for all new types of application. These requirements include open standards, Java, XML, Web services, testing, varying levels of integration with other components and ISV products, pluggability, expandability, role-based development, increased usability for all users, enhanced team support, and increased speed to market.

Debugger

Debugger is included with WebSphere Studio Site Developer Advanced and WebSphere Studio Application Developer.

All products based on the WebSphere Studio Workbench include a debugger that enables you to detect and diagnose errors in your programs running either locally or remotely. The debugger lets you control the execution of your program by setting breakpoints, suspending launches, stepping through your code, and examining the contents of variables.

You can debug live server-side code as well as programs running locally on your workstation. The debugger includes a debug view

that shows threads and stack frames, a processes view that shows all currently running and recently terminated processes, and a console view that lets developers interact with running processes. There are also views that display breakpoints and let you inspect variables.

Performance profiling tools

Performance profiling tools are included with WebSphere Studio Application Developer.

WebSphere Studio Application Developer provides tools that enable you to test application performance early in the development cycle. This allows enough time to make architectural changes and resulting implementation changes, reducing risk early in the cycle and avoiding problems in final performance tests. The profiling tools collect data related to a Java program's run-time behaviour, and present this data in graphical and non-graphical views. This helps you visualize your program execution and explore different patterns within the program.

The tools are useful for performance analysis and for gaining a deeper understanding of your Java programs. You can view object creation and garbage collection, execution sequences, thread interaction, and object references. The tools also show you which operations take the most time, and help you find and solve memory leaks. You can easily identify repetitive execution behaviour and eliminate redundancy, while focusing on the highlights of an execution.

The profiling tools feature:

- Information display suited to object-oriented programs
- Pattern extraction capabilities
- Features to find and solve memory leaks
- Distributed process monitoring
- Colour coding for classes.

IBM announced on 14 August 2003 availability of WebSphere

Studio V5.1, which is a new software development tool that automates many time-consuming tasks that bog down the development process. These new development tools will make creating applications and Web sites faster and easier for developers, including those without Java skills.

IBM has developed the industry's first tools that automate much of the process, so developers can automatically update cross-site information, shift entire groups of links *en masse* to another part of the site, and debug Web applications end-to-end, including VisualBasic and JavaScript code running in the browser.

Using workload simulation tools, such as WebSphere Studio Workload Simulator, you can evaluate how an application will behave in your environment as long as you can recreate a testing environment that matches the projected production environment.

For more information on WebSphere Studio V5.1, please reference http://www-3.ibm.com/software/swnews/swnews.nsf/n/spat5qe2zb?OpenDocument&Site=wsstudio

IBM WebSphere Studio Application Monitor for z/OS and OS/390

IBM WebSphere Studio Application Monitor for z/OS and OS/390 enables application developers and data centre personnel or system administrators to see the actual behaviour of WebSphere applications and of the systems that support them. WebSphere Application Monitor is non-intrusive and requires no modification to the code of the applications being monitored. It provides problem identification at the source by aggregating and analysing the data generated by the applications themselves in real-time, in addition to high-level information such as CPU utilization, resource consumption, response time, and availability indicators. It analyses memory usage and garbage collection statistics, and detects sources of memory leaks.

It assists application developers in pinpointing bottlenecks and problems to a specific area in the application code. It helps application analysts troubleshoot and resolve problems with J2EE applications running on the IBM WebSphere Application

Server for z/OS and OS/390. It allows on-demand changes to the level of application monitoring.

It can also be of use to support staff. WebSphere Studio Application Monitor can perform the following functions:

- Show all or specific client requests.
- Cancel problematic requests in real-time.
- Reveal application threads that take longer than expected to execute cancel, suspend or resume, or change the priority of a thread.

For more information on IBM WebSphere Studio Application Monitor for z/OS please reference http://www-3.ibm.com/software/awdtools/studioapplicationmonitor/

Third party tools

Jinsight 2.1 (Jinsight for Java 2)

This release provides Jinsight tracing and visualization function in IBM Java 2 environments on AIX, Windows, and OS/390.

Jinsight is a tool for visualizing and analysing the execution of Java programs. It is useful for performance analysis, memory leak diagnosis, debugging, or any task in which you need to better understand what your Java program is really doing.

Jinsight brings together a range of techniques that let you explore many aspects of your program:

- Visualization – lets you understand object usage and garbage collection, and the sequence of activity in each thread, all from an object-oriented perspective.
- Patterns – pattern visualizations extract the essential structure in repetitive calling sequences and complex data structures, letting you analyse large amounts of information in a concise form.
- Information exploration – you may specify filtering criteria to

focus your study, or drill down from one view to another to explore details. Create your own units that precisely match features you are studying, and then use them as an additional dimension in many of the views.

- Measurement – study measurements of execution activity or memory summarized at any level of detail, along call paths, and along two dimensions simultaneously.
- Memory leak diagnosis – special features are provided to help you diagnose memory leaks.

For more information on Jinsight see http://www.research.ibm.com/jinsight/docs/index.htm.

Wily IntroScope

IntroScope is an Enterprise Java Application Management solution that helps companies ensure the performance and availability of their mission-critical applications and supporting systems. IntroScope allows IT staff to monitor, improve, and manage enterprise Java applications in each stage of the application life-cycle. It gives users the ability to isolate and resolve performance issues when they arise. IntroScope provides a common language that operations, QA, and development personnel can use to proactively maintain application availability. Introscope is the only platform-independent Java management solution that is fully compatible with any Java IT environment.

IntroScope is certified by IBM as Ready for Tivoli for its integration with Tivoli Enterprise Console.

IntroScope provides a whole application view into an entire Java application environment.

IntroScope is best for performance problems and is not as good for determining memory usage problems.

For more information on IntroScope see http://www.wilytech.com/solutions/index.html.

BugSeeker

BugSeeker for Java 2 is a multi-threaded, cross-platform, source-level, graphical, stand-alone Java debugger that is capable of debugging servlets, EJBs, applets, and applications. Based on JPDA, BugSeeker supports advanced enterprise-level features such as field access and modify watchpoints, exception breakpoints, smart stepping, remote debugging (attaching and listening), powerful expression evaluation, simultaneous multi-process debugging, data tips, and object monitor viewers.

For more information on BugSeeker see http://www.karmira.com.

DiagnoSys

DiagnoSys from H&W lets you analyse the performance of Java applications and their components across the enterprise, from the Web server and application server to databases and the mainframe. It is a non-intrusive system that monitors components without requiring a JVM for each agent. Using patented technology to collect and correlate data from these components, DiagnoSys helps you find problems such as memory leaks, hung threads, and failing resources proactively throughout the life-cycle.

For more information on DiagnoSys see http://www.hwcs.com/products/diagnosys/index.asp.

CONCLUSION

As Java support on the OS/390 platform continues to be enhanced, more features and tools become available to perform Java applications problem determination. We now have application JVM profiling tools and performance analysers that can be used to debug a Java application and analyse its characteristics to help us tune the application to achieve optimum Java performance.

Elena Nanos
IBM Certified Solution Expert in CICS Web Enablement
Zurich NA (USA)

Calling C functions from Assembler – revisited

My previous article, *Calling C functions from Assembler*, published in *MVS Update,* issue 192, September 2002, described how to call C functions from LE-conforming Assembler. This article augments the previous article by discussing how C functions can be called from classic Assembler.

PROBLEM ADDRESSED

As the previous article mentioned, many C functions could usefully be called from Assembler programs. Furthermore, many such Assembler programs may be non-LE-conforming. Rather than upgrading such programs, it may be better to use the approach described in this article.

SOLUTION

IBM supplies four routines to manage and use a persistent C environment. Such a persistent C environment can use standard library functions from system programming C facilities (also the subject of a previous article of mine).

The four routines are:

- EDXCHOTC – set up a persistent C environment (without library)
- EDXCHOTL – set up a persistent C environment (with library)
- EDXCHOTU– invoke a function in the persistent C environment
- EDXCHOTT – terminate the persistent C environment.

These routines are contained in the CEE.SCEESPC library.

Note: a persistent C environment has restrictions:

- The RENT compiler option is not supported
- Exception handling is not supported.

EDXCHOTC – set up a persistent C environment (without library)

The EDXCHOTC function creates a persistent C environment that does not provide C library facilities.

Calling sequence:

```
CALL EDXCHOTC,(handle,stacksz,stackloc),VL

handle   DS    F     returned token
stacksz  DC    F'n'  initial stack size (in bytes)
stackloc DC    F'p'  stack location (Ø < 16MB, 1 = 16MB)
```

Note: the handle (also returned in register 15) is used as input for the EDXCHOTU and EDXCHOTT functions.

EDXCHOTL – set up a persistent C environment (with library)

The EDXCHOTL function creates a persistent C environment that provides C library facilities.

Calling sequence:

```
CALL EDXCHOTL,(handle,stacksz,stackloc),VL

handle   DS    F     returned token
stacksz  DC    F'n'  initial stack size (in bytes)
stackloc DC    F'p'  stack location (Ø < 16MB, 1 = 16MB)
```

Note: the handle (also returned in register 15) is used as input for the EDXCHOTU and EDXCHOTT functions.

EDXCHOTU– invoke a function in the persistent C environment

The EDXCHOTU function runs the specified function in the persistent C environment.

Calling sequence:

```
CALL EDXCHOTU,(handle,funcptr,...)

handle   DS    F        token returned by EDXCHOTL
funcptr  DC    V(funct) address of the function
```

The subsequent parameters are the arguments to be passed to the invoked function. The return value for EDXCHOTU is the value returned by the invoked function.

Note: the arguments passed to the invoked function have the same format as described in the previous paper.

EDXCHOTT – terminate the persistent C environment

The EDXCHOTT function terminates the persistent C environment.

Calling sequence:

```
CALL EDXCHOTT,(handle),VL

handle   DS    F        token returned by EDXCHOTL
```

The subsequent parameters are the arguments to be passed to the invoked function. The return value for EDXCHOTT is the value returned by the invoked function.

BINDER LIBRARY MEMBERS

In addition to the above mentioned functions, the system programming C (SPC) facilities library contains the following three members with special versions of the specified functions. The appropriate library member(s) must be included when the SPC version of these functions is to be used:

malloc(), calloc(), realloc(), free()	**EDCXMEM**
exit()	**EDCXEXIT**
sprintf()	**EDCXSPRT**

EXAMPLE 1

The following example shows a trivial program that uses the puts() C function to output a fixed character string. The program serves to show how persistent C environment functions are used.

Note: for simplicity, the program uses a static save area.

```
TESTPCE  CSECT
TESTPCE  AMODE 31
TESTPCE  RMODE ANY
         STM   R14,R12,12(R13)  save registers
         LR    R3,R15           save entry-point address
         USING TESTPCE,R3       set base register
         LA    R1,SA            local save area
         ST    R13,4(R1)        set low chain
         LR    R13,R1           set local save area
** set up persistent C environment with library functions
         LA    R4,HANDLE        address of returned handle
         LA    R5,STKSIZE       stack size
         LA    R6,STKLOC        stack location
         STM   R4,R6,PARMLIST   set in argument list
         LA    R1,PARMLIST      address of argument list
         L     R15,=V(EDCXHOTL)
         BASR  R14,R15          set up C environment with libr funct
** call C-function
         LA    R4,HANDLE        address of handle
         MVC   PFN,=V(PUTS)
         LA    R5,PFN           A(function)
         LA    R6,STR           A(argument)
         STM   R4,R6,PARMLIST   set in argument list
         LA    R1,PARMLIST      address of argument list
         L     R15,=V(EDCXHOTU)
         BASR  R14,R15          call function
** terminate persistent C environment
         LA    R4,HANDLE        address of handle
         ST    R4,PARMLIST
         LA    R1,PARMLIST
         L     R15,=V(EDCXHOTT)
         BASR  R14,R15          terminate C environment
**
         LR    R1,R13           address of local save area
         L     R13,4(,R13)      return address of lower save area
         LM    R14,R12,12(R13)  restore original registers
         LA    R15,Ø            program return value
         BSM   Ø,R14            return
*
PFN      DS    A                pointer to function
STKSIZE  DC    F'4Ø96'          stack size (in bytes)
STKLOC   DC    F'1'             stack location (Ø < 16MB, 1 >= 16MB)
*
STR      DC    C'alpha',AL1(Ø)  C-string
         LTORG
SA       DS    18F              local save area
PARMLIST DS    4A               parameter list
HANDLE   DS    A                environment handle
         SPACE
* symbolic register equates
```

```
R0        EQU   0
R1        EQU   1
R2        EQU   2
R3        EQU   3
R4        EQU   4
R5        EQU   5
R6        EQU   6
R7        EQU   7
R8        EQU   8
R9        EQU   9
R10       EQU   10
R11       EQU   11
R12       EQU   12
R13       EQU   13
R14       EQU   14
R15       EQU   15
          END
```

EXAMPLE 2

The following example shows a practical use of this capability. In this example, a REXX function (RXSQRT) returns the square root of the passed argument. RXSQRT uses the following functions:

- sscanf() – converts the passed string into a double floating-point number.
- sqrt() – calculates the square root of a double floating-point number (the result is returned as a double floating-point number).
- sprintf() – converts a double floating-point number into a string.

Note: the use of a REXX function shows a practical example where the persistent C environment can be used. For simplicity, the REXX function is not written as re-entrant code.

```
* Invocation y = rxsqrt(x)
* x and y are REXX strings.
* Example:
*  y = rxsqrt(2.1)
*  SAY y /* 1.449138 */
          TITLE 'REXX FUNCTION - RETURN SQUARE ROOT'
          PRINT NOGEN
```

```
RXSQRT   CSECT
RXSQRT   AMODE 31
RXSQRT   RMODE ANY
** save environment
         STM   R14,R12,12(R13)  save registers
         LR    R3,R15           save entry-point address
         LR    R2,R1            save call parameters
         LR    R7,RØ            save environment
         USING RXSQRT,R3        set base register
         LA    R1,SA            local save area
         ST    R13,4(R1)        set low chain
         LR    R13,R1           set local save area
** set up C environment with librarian functionality
         LA    R4,HANDLE        address of returned handle
         LA    R5,STKSIZE       stack size
         LA    R6,STKLOC        stack location
         O     R6,=X'8ØØØØØØØ'  set VL flag
         STM   R4,R6,PARMLIST   set in argument list
         LA    R1,PARMLIST      address of argument list
         L     R15,=V(EDCXHOTL)
         BASR  R14,R15          set up C environment with libr funct
         LA    R1,FN
         ST    R1,PFN
* get parameters
         LR    R1,R2            restore call parameters
         USING ENVBLOCK,R7
         USING EFPL,R1
         L     R2,EFPLEVAL      PTR(Evaluation Block)
         L     R9,Ø(R2)         A(Evaluation Block)
         USING EVALBLOCK,R9
         SPACE 1
         L     R1Ø,EFPLARG      A(parsed Argument List)
         USING ARGTABLE_ENTRY,R1Ø
* one call parameter
NEXTPARM LM    R4,R5,ARGTABLE_ARGSTRING_PTR
* R4: A(argument)
* R5: L(argument)
         LM    R14,R15,INARG
         CR    R15,R5           test shortest length (avoid overflow)
         JL    *+6
         LR    R15,R5           set shortest length
         MVCL  R14,R4
         MVI   Ø(R14),NULL      make C string
         SPACE 1
         L     R8,ENVBLOCK_IRXEXTE
         USING IRXEXTE,R8
         SPACE 1
* sscanf(instr,"%f",&f1);
         MVC   FN,=V(SSCANF)
         MVC   PARMS(PARM1L),PARM1
```

```
         BAS   R1Ø,CALL
* f2 = sqrt(f1);
         MVC   FN,=V(SQRT)
         MVC   PARMS(PARM2L),PARM2
         BAS   R1Ø,CALL
* sprintf(str,"%f",f2);
         LA    RØ,EVALBLOCK_EVDATA   set result address
         ST    RØ,PARM31
         MVC   PARMS(PARM3L),PARM3
         MVC   FN,=V(SPRINTF)
         BAS   R1Ø,CALL
         SPACE 1
         ST    R15,EVALBLOCK_EVLEN   result length
*
EXIT     LA    R1,PARMLIST
         OI    Ø(R1),X'8Ø'      set VL flag
         L     R15,=V(EDCXHOTT)
         BASR  R14,R15          terminate C environment
** return
         LR    R1,R13           address of local save area
         L     R13,4(,R13)      return address of lower save area
         LM    R14,R12,12(R13)  restore original registers
         LA    R15,Ø            program return value
         BSM   Ø,R14            return
         SPACE 2
CALL     DS    ØH               call C function
* <FN> has been preloaded with the address of the function
* <PARMS> (part of PARMLIST) contains the function arguments
         LA    R1,PARMLIST
         L     R15,=V(EDCXHOTU)
         BASR  R14,R15
         BR    R1Ø              return
         SPACE 2
* register equates
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
```

```
         SPACE 1
STKSIZE  DC    F'4Ø96'          stack size
STKLOC   DC    F'1'             stack location (Ø < 16MB, 1 >= 16MB)
SA       DS    18F              local save area
         SPACE 1
FN       DS    A
         SPACE 1
HANDLE   DS    A
PARMLIST DS    ØA
PHANDLE  DS    A
PFN      DS    A
PARMS    DS    XL16             max. 16 bytes (in this program)
         SPACE 1
PARM1    DC    A(ARG,MASK,F1)
PARM1L   EQU   *-PARM1
         SPACE 1
PARM2    DC    A(F2)
F1       DS    DL8
PARM2L   EQU   *-PARM2
         SPACE 1
PARM3    DS    A
PARM31   EQU   PARM3
         DC    A(MASK)
F2       DS    DL8
PARM3L   EQU   *-PARM3
         SPACE 1
MASK     DC    C'%f',AL1(Ø)
         SPACE 1
NULL     EQU   X'Ø'
INARG    DC    A(ARG,L'ARG)
ARG      DS    CL16,X
         SPACE 1
         LTORG
         TITLE 'REXX DSECTs'
         IRXEXECB
         IRXEXTE
         IRXEFPL
         IRXEVALB
         IRXARGTB
         IRXENVB
         END
```


SMP/E GIMAPI interface

Sometimes we need detailed information from SMP/E; for example, a PTF list containing PTFs applied on a specific date, or a holddata list containing only the ACTION reason. It is impossible to get reports like this from SMP/E batch or the SMP/E ISPF interface.

However, it is possible to get useful SMP/E reports with the SMP/E GIMAPI interface. This utility consists of three parts – an ISPF panel, a REXX EXEC, and an Assembler program. On the ISPF panel there are five parameters:

- CSI name – a character string that specifies the name of the global CSI to be searched by the QUERY command.
- ZONE name – a character string that specifies the zones from which data is to be retrieved. You may enter one or more specific zone names separated by commas or blanks, or any of these values:
  - GLOBAL – the global zone
  - ALLTZONES – all target zones
  - ALLDZONES – all distribution zones
  - * – all zones defined in the GLOBAL zone index.
- ENTRY – a character string that indicates the entry types from the specified zone to be searched. It is possible to enter more than one entry name separated by blanks or commas.
- SUBENTRY – a character string used to indicate the subentries for which data is retrieved. It is possible to enter more than one subentry separated by commas or blanks.
- FILTER – a character string that specifies the set of conditions with which to limit the set of entries being retrieved. A condition is in the following form:

```
subentry operator 'value'
```

For example, FMID='HBB7703' or INSTALLDATE>='03145'.

It is possible to enter complex conditional phrases using & (AND) or ! (OR) operators. Parentheses may be used to group search conditions.

CSI NAME, ZONE NAME, ENTRY, and SUBENTRY parameters must be entered. The FILTER parameter is optional. For more information about query parameters, see *Chapter 6, SMP/E CSI Application Programming Interface* in the *SMP/E Reference* book. ISPF panel source code must be located in a dataset that is concatenated to ISPPLIB. Also, the Assembler program must be located in a dataset that is concatenated to ISPLLIB.

The entry panel looks as follows:

```
|------------------------ SMP/E Query Panel --------------------------|
|                                                                      |
|                                                                      |
|                                                                      |
|                                                                      |
|    CSI    Name :  SMPE.GLOBAL.CSI                                    |
|    ZONE   Name :  MVST100                                            |
|    ENTRY       :  SYSMOD                                             |
|    SUBENTRY    :  *                                                  |
|    FILTER      :  INSTALLDATE='03231' & FMID='HBB773' & SMODTYPE='P  |
|                   TF'                                                |
|                                                                      |
|                                                                      |
|                                                                      |
|                                                                      |
|                                                                      |
|    Press ENTER to go on, if you want to exit press PF3.              |
|                                                                      |
|                                                                      |
|----------------------------------------------------------------------|
```

The output looks as follows:

```
GIMAPI INTERFACE QUERY REPORT
GIMAPI VERSION 03.01.00 PTF 18          REPORT DATE : 19.08.2003
REPORT TIME : 16:33:38

QUERY PARAMETERS...
CSI NAME         : SMPE.GLOBAL.CSI
ZONE NAME        : MVST100
ENTRY NAME       : SYSMOD
SUBENTRY NAME    : *
```

```
FILTER          : INSTALLDATE='Ø3231' & FMID='HBB77Ø3' & SMODTYPE='PTF'
-------------------------------------------------------------------------
SYSMOD       --> UW95456
ZONE NAME    --> MVST1ØØ
   APPLY        : YES
   BYPASS       : NO
   DELLMOD      : NO
   ELEMMOV      : NO
   ERROR        : NO
   FMID         : HBB77Ø3
   INSTALLDATE  : Ø3231
   INSTALLTIME  : Ø9:28:55
   JCLIN        : NO
   MOD          : IXLM2PRQ
                  IXLR1RT
   PRE          : UW92483
                  UW93182
   SUPING       : AW54584
                  BW54584
                  CW54584
                  JW54584
   RECDATE      : Ø3197
   RECTIME      : 14:22:35
   REGEN        : NO
   RENLMOD      : NO
   RESTORE      : NO
   SMODTYPE     : PTF
   SOURCEID     : HIPER
                  PUTØ211
                  RSUØ212
                  SYSPLXDS
-------------------------------------------------------------------------
SYSMOD       --> UW95494
ZONE NAME    --> MVST1ØØ
   APPLY        : YES
   BYPASS       : YES
   DELLMOD      : NO
   ELEMMOV      : NO
   ERROR        : NO
   FMID         : HBB77Ø3
   INSTALLDATE  : Ø3231
   INSTALLTIME  : Ø9:28:56
   JCLIN        : NO
   MOD          : IXCF1SCF
                  IXCF1TF3
                  IXCL1CTD
                  IXCL1CTS
                  IXCL1ERE
                  IXCL1IOC
                  IXCL1IOP
```

```
                IXCL1IOR
                IXCL1RS
                IXCL1STL
                IXCL1SWT
                IXCL1UNL
                IXCL1WS
                IXCM2MST
   MSGENU       : IXCXCFEN
   PRE          : UW831Ø7
                  UW84672
                  UW86986
                  UW927Ø9
   SUPING       : AW46877
                  AW49159
                  AW51741
                  BW46877
                  BW49159
                  BW51741
                  CW46877
                  CW51741
                  FW49159
                  FW51741
                  HW49159
                  HW51741
                  IW49159
                  IW51741
                  JW46877
                  JW49159
                  JW51741
                  UW76335
                  UW84336
   RECDATE      : Ø3197
   RECTIME      : 14:22:43
   REGEN        : NO
   RENLMOD      : NO
   RESTORE      : NO
   SMODTYPE     : PTF
   SOURCEID     : HIPER
                  PUTØ212
                  RSUØ212
                  SMCCOR
                  SYSPLXDS
```

SMPQPNL PANEL SOURCE CODE

```
)ATTR FORMAT(MIX)
 -   TYPE(NEF) PADC(USER) CAPS(ON)
 *   TYPE(NT)  SKIP(ON)
 >   TYPE(CH)
```

```
)BODY WINDOW(7Ø,18) ASIS
+
+
+
+
+  >CSI    Name : -csinm                                          +
+  >ZONE   Name : -zonenm                                           +
+  >ENTRY       : -entry                                            +
+  >SUBENTRY    : -subentry                                         +
+  >FILTER      : -filter1                                          *
+                 -filter2                                          *
+                 -filter3                                          *
+                 -filter4                                          *
+                 -filter5                                          +
+
+
+  >Press ENTER to go on, if you want to exit press PF3.
+
+
)INIT
  .CURSOR   = csinm
  .ATTR(csinm)    = 'JUST(ASIS)'
  .ATTR(zonenm)   = 'JUST(ASIS)'
  .ATTR(entry)    = 'JUST(ASIS)'
  .ATTR(subentry) = 'JUST(ASIS)'
  .ATTR(filter1)  = 'JUST(ASIS)'
  .ATTR(filter2)  = 'JUST(ASIS)'
  .ATTR(filter3)  = 'JUST(ASIS)'
  .ATTR(filter4)  = 'JUST(ASIS)'
  .ATTR(filter5)  = 'JUST(ASIS)'
  &csinm    = ''
  &zonenm   = ''
  &entry    = ''
  &subentry = ''
  &filter   = ''
  &filter1  = ''
  &filter2  = ''
  &filter3  = ''
  &filter4  = ''
  &filter5  = ''
  &ZWINTTL = 'SMP/E Query Panel'
)PROC
  VER(&csinm,NB,DSNAME)
  VER(&zonenm,NB,EBCDIC)
  VER(&entry,NB,EBCDIC)
  VER(&subentry,NB,EBCDIC)
  &filter = '&filter1&filter2&filter3&filter4&filter5'
)END
```

SMPQREXX REXX SOURCE CODE

```
/***************************** REXX ***********************************/
/*                                                                    */
/* Description :                                                      */
/*                                                                    */
/* This REXX allocates a temporary dataset 'userid.PSØ.SMPQUERY.TEMP' */
/* for output. After the allocation of the temporary dataset, the     */
/* REXX calls SMPQPNL ISPF panels for entering the query parameters.  */
/* And then the REXX calls the Assembler program SMPQASM with the     */
/* query parameters. When the SMPQASM ends, REXX browses the          */
/* temporary dataset for query output.                                */
/*                                                                    */
/**********************************************************************/
SMPQREXX:
  userid = SYSVAR($sysuid$)
  dsnm   = userid || '.PSØ.SMPQUERY.TEMP'
  msgstat = MSG($OFF$)
  'ALLOC  FI(SYSPRINT) DA('dsnm') SHR REUSE'
  if RC ¬= Ø then
   do
    'ALLOC DS('dsnm') NEW DSORG(PS) RECFM(F,B) LRECL(132) BLKSIZE(132Ø)
          SPACE(132ØØØ,132Ø) RELEASE'
     'ALLOC FI(SYSPRINT) DA('dsnm') SHR REUSE'
   end
  address ISPEXEC 'ADDPOP POPLOC(F1)'
  address ISPEXEC 'DISPLAY PANEL(SMPQPNL)'
  if RC = 8 then call exit
  address ISPEXEC 'REMPOP'
  x = SMPQASM(csinm,zonenm,entry,subentry,filter)
  address ISPEXEC 'BROWSE DATASET('dsnm')'
  if RC = Ø then call SMPQREXX
exit:
  'FREE FI(SYSPRINT)'
  'FREE FI(SMPCSI)'
  exit
```

SMPQASM ASSEMBLER SOURCE CODE

```
TITLE 'GIMAPI INTERFACE ROUTINE'
*--------------------------------------------------------------------*
* NAME            : SMPQASM                                          *
* LANGUAGE        : HLASM                                            *
* SYSTEM          : OS/39Ø V2.1Ø                                     *
* AUTHOR          : Ayhan YALKUT                                     *
* REMARKS         : This program will query and print the contents of *
*                   the SMP/E CSI using the GIMAPI interface .       *
* INVOCATION      : From a SMPQREXX REXX .                           *
*--------------------------------------------------------------------*
SMPQASM  CSECT
```

```
*-------------------------------------------------------------------*
* REGISTER DECLARES                                                 *
*-------------------------------------------------------------------*
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
*-------------------------------------------------------------------*
* STANDART ENTRY LINKAGE                                            *
*-------------------------------------------------------------------*
         STM   14,12,12(13)                Save Caller GPRS
         LR    12,15                       Program Base Register
         USING SMPQASM,12                  Inform The Assembler
         ST    13,SAVE+4                   Save Backward  Addr
         LA    14,SAVE                     New  Save Area Addr
         ST    14,8(13)                    Save Forward   Addr
         LR    13,14                       New Save Area Addr
         ST    1,PARMADDR                  Save Input Parmeters Addr
*-------------------------------------------------------------------*
* OPEN SYSPRINT DD                                                  *
*-------------------------------------------------------------------*
         OPEN  (SYSPRINT,OUTPUT)           Open SYSPRINT DD
*-------------------------------------------------------------------*
* PRINT HEADER1                                                     *
*-------------------------------------------------------------------*
         PUT   SYSPRINT,HEADER1            Print HEADER1
*-------------------------------------------------------------------*
* LOAD THE API                                                      *
*-------------------------------------------------------------------*
         LA    2,APIPGM                    APIPGM Addr
         LOAD  EPLOC=(2),LOADPT=PGMADR     Load GIMAPI to memory
*-------------------------------------------------------------------*
* DO THE VERSION QUERY                                              *
*-------------------------------------------------------------------*
         L     15,PGMADR                   GIMAPI Load Point
         CALL  (15),(QUERYVER,Ø,CMDOUT,APILANG,RC,CC,MSG$)
*-------------------------------------------------------------------*
* SEE WHAT WAS RETURNED                                             *
```

```
*-------------------------------------------------------------------*
         L     3,RC                          Return code of GIMAPI
         LTR   3,3                           Is Return Code good?
         BNZ   ERRORPRT                      No, go to ERRORPRT
*-------------------------------------------------------------------*
* ESTABLISH ADDRESSABILITY OF RETURNED INFORMATION                   *
*-------------------------------------------------------------------*
         L     3,CMDOUT                      Output Addr of The QUERY
         USING API_VERSION,3                 Inform The Assembler
*-------------------------------------------------------------------*
* GET LOCAL SYSTEM DATE AND TIME                                     *
*-------------------------------------------------------------------*
         TIME  DEC,TIMEDATE,ZONE=LT,DATETYPE=DDMMYYYY,LINKAGE=SYSTEM
*-------------------------------------------------------------------*
* PRINT HEADER2                                                      *
*-------------------------------------------------------------------*
         MVC   AVER(2),APIVER                Move API Version to AVER
         MVC   AREL(2),APIREL                Move API Release to AREL
         MVC   AMOD(2),APIMOD                Move API Mod.    to AMOD
         MVC   APTF(2),APIPTF                Move API PTF     to APTF
         MVC   OUTTOD,MASK                   Set up OUTTOD for ED
         ED    OUTTOD,TIMEDATE               Edit OUTTOD
         MVC   LDATE(2),OUTTOD+17            Move Local Day  to LDATE
         MVC   LDATE+3(2),OUTTOD+19          Move Local Mon  to LDATE
         MVC   LDATE+6(4),OUTTOD+21          Move Local Year to LDATE
         MVC   LTIME(2),OUTTOD+1             Move Local Hour to LTIME
         MVC   LTIME+3(2),OUTTOD+3           Move Local Min  to LTIME
         MVC   LTIME+6(2),OUTTOD+5           Move Local Sec  to LTIME
         PUT   SYSPRINT,HEADER2              Print HEADER2
         PUT   SYSPRINT,BLANK                Print BLANK
         PUT   SYSPRINT,HEADER3              Print HEADER3
         DROP  R3                            Inform The Assembler
*-------------------------------------------------------------------*
* FREE THE STORAGE OBTAINED DURING THE VERSION QUERY                 *
*-------------------------------------------------------------------*
         L     15,PGMADR                     GIMAPI Load Point
         CALL  (15),(FREECMD,Ø,CMDOUT,APILANG,RC,CC,MSG$)
*-------------------------------------------------------------------*
* ESATBLISH ADDRESSABILITY OF INPUT PARAMETERS FOR QUERY              *
*-------------------------------------------------------------------*
         L     1,PARMADDR                    REXX Input Parmeters Addr
         USING EFPL,1                        Inform The Assembler
         L     2,EFPLARG                     ARGUMENT TABLE Addr
         USING ARGTABLE_ENTRY,2              Inform The Assembler
         L     1Ø,EFPLEVAL                   EFPLEVAL Addr
         L     1Ø,Ø(,1Ø)                     EVALBLOCK Addr
         USING EVALBLOCK,1Ø                  Inform The Assembler
*-------------------------------------------------------------------*
* SET UP AND PRINT QUERY PARAMETERS                                  *
*-------------------------------------------------------------------*
```

```
         USING PARMDATA,11                    Inform The Assembler
         XC    QUERY_PARMS,QUERY_PARMS        Clear QUERY_PARMS
         L     11,ARGTABLE_ARGSTRING_PTR      CSI NAME Pointer
         ST    11,PCSI                        Save CSI NAME Pointer
         L     4,ARGTABLE_ARGSTRING_LENGTH    CSI NAME Length
         ST    4,CSILEN                       Save CSI NAME Length
         MVC   OUTREC(132),BLANK              Move BLANK to OUTREC
         MVC   OUTREC(18),=C'CSI NAME         : '
         BCTR  4,Ø                            Decrease Length by 1
         EX    4,$MOVPARM                     Move CSI NAME to OUTREC
         PUT   SYSPRINT,OUTREC                Print OUTREC
         LA    2,ARGTABLE_NEXT-ARGTABLE_ENTRY(R2)
*                                             Get next Input Parameter
         L     11,ARGTABLE_ARGSTRING_PTR      ZONE NAME Pointer
         ST    11,PZONE                       Save ZONE NAME Pointer
         L     4,ARGTABLE_ARGSTRING_LENGTH    ZONE NAME Length
         ST    4,ZONELEN                      Save ZONE NAME Length
         MVC   OUTREC(132),BLANK              Move BLANK to OUTREC
         MVC   OUTREC(18),=C'ZONE NAME        : '
         BCTR  4,Ø                            Decrease Length by 1
         EX    4,$MOVPARM                     Move ZONE NAME to OUTREC
         PUT   SYSPRINT,OUTREC                Print OUTREC
         LA    2,ARGTABLE_NEXT-ARGTABLE_ENTRY(R2)
*                                             Get next Input Parameter
         L     11,ARGTABLE_ARGSTRING_PTR      ENTRY NAME Pointer
         ST    11,PENTRY                      Save ENTRY NAME Ptr
         L     4,ARGTABLE_ARGSTRING_LENGTH    ENTRY NAME Length
         ST    4,ENTRYLEN                     Save ENTRY NAME Length
         MVC   OUTREC(132),BLANK              Move BLANK to OUTREC
         MVC   OUTREC(18),=C'ENTRY NAME       : '
         BCTR  4,Ø                            Decrease Length by 1
         EX    4,$MOVPARM                     Move ENAME to OUTREC
         PUT   SYSPRINT,OUTREC                Print OUTREC
         LA    2,ARGTABLE_NEXT-ARGTABLE_ENTRY(R2)
*                                             Get next Input Parameter
         L     11,ARGTABLE_ARGSTRING_PTR      SUBENTRY NAME Pointer
         ST    11,PSUBENTRY                   Save SUBENTRY NAME Pointer
         L     4,ARGTABLE_ARGSTRING_LENGTH    SUBENTRY NAME Length
         ST    4,SUBENTRYLEN                  Save SUBENTRY NAME Length
         MVC   OUTREC(132),BLANK              Move BLANK to OUTREC
         MVC   OUTREC(18),=C'SUBENTRY NAME    : '
         BCTR  4,Ø                            Decrease Length by 1
         EX    4,$MOVPARM                     Move SUBENTRY to OUTREC
         PUT   SYSPRINT,OUTREC                Print OUTREC
         LA    2,ARGTABLE_NEXT-ARGTABLE_ENTRY(R2)
*                                             Get next Input Parameter
         L     11,ARGTABLE_ARGSTRING_PTR      FILTER Pointer
         ST    11,PFILTER                     Save FILTER Pointer
         L     4,ARGTABLE_ARGSTRING_LENGTH    FILTER Length
         ST    4,FILTERLEN                    Save FILTER Length
```

```
         MVC   OUTREC(132),BLANK            Move BLANK to OUTREC
         MVC   OUTREC(18),=C'FILTER         : '
         CL    4,=F'Ø'                      FILTER Length = Ø ?
         BE    PRNTF                        Yes, go to PRNTF
         BCTR  4,Ø                          Decrease Length by 1
         EX    4,$MOVPARM                   Move FILTER to OUTREC
PRNTF    PUT   SYSPRINT,OUTREC              Print OUTREC
*-------------------------------------------------------------------*
* DO THE QUERY                                                      *
*-------------------------------------------------------------------*
         L     15,PGMADR                    GIMAPI Load Point
         CALL  (15),(QUERYCMD,QUERY_PARMS$,CMDOUT,APILANG,RC,CC,MSG$)
*-------------------------------------------------------------------*
* SEE WHAT WAS RETURNED                                             *
*-------------------------------------------------------------------*
         L     3,RC                         Return Code of f GIMAPI
         LTR   3,3                          Is the Return Code good?
         BNZ   ERRORPRT                     No, go to ERRORPRT
*
*-------------------------------------------------------------------*
* ESTABLISH ADDRESSABILITY OF RETURNED INFORMATION                  *
*-------------------------------------------------------------------*
         L     3,CMDOUT                     Output Addr of The QUERY
         USING ENTRY_LIST,3                 Inform The Assembler
         L     4,ENTRIES                    ENTRIES Addr
         USING CSIENTRY,4                   Inform The Assembler
         L     5,SUBENTRIES                 SUBENTRIES Addr
         USING SUBENTRY,5                   Inform The Assembler
         L     6,SUBENTDATA                 SUBENTDATA Addr
         USING ITEM_LIST,6                  Inform The Assembler
         L     7,DATA                       DATA Addr
         USING RETDATA,7                    Inform The Assembler
         USING VER,8                        Inform The Assembler
*-------------------------------------------------------------------*
* PRINT ENTRIES                                                     *
*-------------------------------------------------------------------*
PRTENT   LTR   4,4                          Last Entry ?
         BZ    NEXTENTR                     Yes, go to NEXTENTR
         MVI   OUTREC,C'-'
         MVC   OUTREC+1(132),OUTREC         Move '-' to OUTREC
         PUT   SYSPRINT,OUTREC              Print OUTREC
         MVC   OUTREC(132),BLANK            Clear OUTREC
         MVC   OUTREC(12),TYPE              Move ENTRY TYPE to OUTREC
         MVC   OUTREC+13(4),=C'-->'        Move '-->' to OUTREC
         MVC   OUTREC+18(8),ENTRYNAME       Move ENTRY NAME to OUTREC
         PUT   SYSPRINT,OUTREC              Print OUTREC
         MVC   OUTREC(132),BLANK            Clear OUTREC
         MVC   OUTREC(17),=C'ZONE NAME    -->'
         MVC   OUTREC+18(7),ZONENAME        Move ZONENAME to OUTREC
         PUT   SYSPRINT,OUTREC              Print OUTREC
```

```
         XR    8,8                          Clear R8
         L     5,SUBENTRIES                 SUBENTRIES Addr
         B     PRTSUB                       Branch PRTSUB
MOREENT  LTR   8,8                          More ENTRY after VER?
         BNZ   NEXTVER                      Yes, go to NEXTVER
         L     4,CSINEXT                    Next ENTRY Addr
         B     PRTENT                       Branch PRTENT
NEXTENTR L     3,NEXT                       NEXT ENTRY Addr
         LTR   3,3                          Last ENTRY in the LIST ?
         BZ    CLOSEOUT                     Yes, go to CLOSEOUT
         L     4,ENTRIES                    New ENTRIES Addr
         B     PRTENT                       Branch PRTENT
*-------------------------------------------------------------------*
* GET NEXT SUBTYPE AFTER VER                                        *
*-------------------------------------------------------------------*
NEXTVER  L     5,VERNEXT                    Next SUBENTRIES Addr
         XR    8,8                          Clear R8
*-------------------------------------------------------------------*
* PRINT SUBENTRIES                                                  *
*-------------------------------------------------------------------*
PRTSUB   LTR   5,5                          Last SUBENTRIES ?
         BZ    MOREENT                      Yes, go to MOREENT
         CLC   SUBTYPE(12),=C'VER        '  SUBTYPE = VER ?
         BE    PRTVSUB                      Yes, go to PRTVSUB
         MVC   OUTREC(132),BLANK            Clear OUTREC
         MVC   OUTREC+3(11),SUBTYPE         Move SUBTYPE to OUTREC
         MVC   OUTREC+16(1),=C':'           Move ':' to OUTREC
         L     6,SUBENTDATA                 SUBENTDATA Addr
         L     7,DATA                       DATA Addr
         B     PRTDATA                      Branch PRTDATA
MORESUB  L     5,SUBNEXT                    Next SUBENTRIES Addr
         B     PRTSUB                       Branch PRTSUB
*-------------------------------------------------------------------*
* PRINT VER SUBENTRIES                                              *
*-------------------------------------------------------------------*
PRTVSUB  LR    8,5                          Next SUBENTRIES Addr after
*                                                                VER
         L     5,4(5)
         L     5,4(5)
         B     PRTSUB                       Branch PRTSUB
*-------------------------------------------------------------------*
* PRINT DATA                                                        *
*-------------------------------------------------------------------*
PRTDATA  LTR   7,7                          Last DATA ?
         BZ    MORESUB                      Yes, go to MORESUB
         L     9,DATALEN                    DATALEN
         C     9,=F'Ø'                      DATALEN = Ø ?
         BE    PUTREC                       Yes, go to PUTREC
         BCTR  9,Ø                          Decrease DATALEN By 1
         EX    9,$MOVDATA                   Move RETDATA to OUTREC
```

```
PUTREC   PUT   SYSPRINT,OUTREC              Print OUTREC
         MVC   OUTREC(132),BLANK           Clear OUTREC
NEXTITEM L     6,ITMNEXT                   ITMNEXT Addr
         LTR   6,6                         Last ITMNEXT ?
         BZ    MORESUB                     Yes, go to MORESUB
         L     7,DATA                      Load DATA
         B     PRTDATA                     Branch PRTDATA
*
*-------------------------------------------------------------------*
* PRINT ERROR MESSAGES                                              *
*-------------------------------------------------------------------*
ERRORPRT L     6,MSG$                      Error Message Addr
         L     7,DATA                      DATA
         L     9,DATALEN                   DATALEN
CHKMSGLN C     9,OUTRECLN                  DATALEN > OUTRECLN ?
         BNH   SETMSGLN                    No, go to SETMSGLN
         L     9,OUTRECLN                  OUTRECLN
SETMSGLN BCTR  9,Ø                         Decrease DATALEN By 1
         MVC   OUTREC(132),BLANK           Clear OUTREC
         EX    9,$MOVDATA                  Move Messages to OUTREC
         PUT   SYSPRINT,OUTREC             Print OUTREC
         L     9,DATALEN                   DATALEN
         C     9,OUTRECLN                  DATALEN > OUTRECLN ?
         BNH   CLOSEOUT                    No, go to CLOSEOUT
         S     9,OUTRECLN                  Subtract DATALEN-OUTRECLN
         ST    9,DATALEN                   Save new DATALEN
         A     4,OUTRECLN                  Add
         B     CHKMSGLN                    Branch CHKMSGLN
*-------------------------------------------------------------------*
* CLOSE SYSPRINT DD                                                 *
*-------------------------------------------------------------------*
CLOSEOUT CLOSE   SYSPRINT                  Close SYSPRINT DD
*-------------------------------------------------------------------*
* FREE THE STORAGE OBTAINED DURING THE QUERY                        *
*-------------------------------------------------------------------*
CLEANUP  L     15,PGMADR                   PGMADR Addr
         CALL  (15),(FREECMD,Ø,CMDOUT,APILANG,RC,CC,MSG$)
*-------------------------------------------------------------------*
* DELETE GIMAPI                                                     *
*-------------------------------------------------------------------*
         DELETE  EPLOC=APIPGM              Delete GIMAPI from memory
*-------------------------------------------------------------------*
* SET EVALBLOCK TO RETURN INFORMATION                               *
*-------------------------------------------------------------------*
         MVC   EVALBLOCK_EVLEN(4),=F'1'    Set EVALBLOCK for Return
         MVC   EVALBLOCK_EVDATA(4),=C'Ø'   Set EVALBLOCK for Return
*-------------------------------------------------------------------*
* EXIT                                                              *
*-------------------------------------------------------------------*
EXIT     XR     15,15                      Clear R15 for Return Code
```

```
         L      13,4(,13)                    Load Save Area Address
         L      14,12(,13)                   Load Return Address
         LM     ØØ,12,2Ø(13)                 Load Register
         BR     14                           Return to Operating System
$MOVDATA MVC    OUTREC+18(Ø),RETDATA
$MOVPARM MVC    OUTREC+18(Ø),PARMDATA
SYSPRINT DCB    DDNAME=SYSPRINT,DSORG=PS,MACRF=PM
APIPGM   DC     CL8'GIMAPI  '                API Name
QUERYCMD DC     CL8'QUERY   '                API Query Command
QUERYVER DC     CL8'VERSION '                API Version Command
FREECMD  DC     CL8'FREE    '                API Free Command
APILANG  DC     CL3'ENU'                     API Language
         DS     ØF
QUERY_PARMS$ DC AL4(QUERY_PARMS)             Parameter Area Address
PGMADR   DC     AL4(Ø)                       GIMAPI Entry Point Addr
CMDOUT   DC     AL4(Ø)                       API Output Address
MSG$     DC     AL4(Ø)                       API Error Address
RC       DS     F'Ø'                         API Return Code
CC       DS     F'Ø'                         API Completion Code
HEADER1  DS     ØCL132
         DC     51CL1' '
         DC     CL29'GIMAPI INTERFACE QUERY REPORT'
         DC     52CL1' '
HEADER2  DS     ØCL132
         DC     CL15'GIMAPI VERSION '
AVER     DS     CL2                          GIMAPI Version
         DC     CL1'.'
AREL     DS     CL2                          GIMAPI Release
         DC     CL1'.'
AMOD     DS     CL2                          GIMAPI Modification
         DC     CL5' PTF '
APTF     DS     CL2                          GIMAPI PTF
         DC     CL51' '
         DC     CL14'REPORT DATE : '
LDATE    DC     CL1Ø'  .  .    '             Local Date
         DC     CL18'    REPORT TIME : '
LTIME    DC     CL9'  :  :   '               Local Time
HEADER3  DS     ØCL132
         DC     CL19'QUERY PARAMETERS...'
         DC     CL113' '
SAVE     DS     18F'Ø'                       Save Area
PARMADDR DS     F                            Input Parameter Address
TIMEDATE DS     PL16                         Local Time & Date
OUTTOD   DS     CL32                         Local Time & Date in CHAR
OUTREC   DS     CL132                        Output Record
OUTRECLN DC     F'12Ø'                       Output Record Length
BLANK    DC     CL132' '
MASK     DC     X'402020202020202020202020202020202020202020202020202020202X
                 Ø2020202Ø2Ø'
         GIMMASM                             SMP/E API Macro
```

```
         IRXEFPL                          External Func Parm List
         IRXARGTB                         REXX Argument Table Block
         IRXEVALB                         REXX Evaluation Block
RETDATA  DSECT  CLØ                       Return Data
PARMDATA DSECT  CLØ                       Parameter Data
         END    SMPQASM
```

Ayhan Yalkut
OS/390 System Programmer
Pamukbank (Turkey)

Simple conversion of data codes

It is often necessary to write programs for data encoding of both application and base system software. Some typical examples would arise from the necessity to:

- Convert ASCII/EBCDIC.
- Run user-developed cryptographic algorithms.
- Convert certain special printer characters.
- Convert upper/lower case characters.
- Correct specific fields in a data flow.

I have tried to write code that is easy to maintain (COBOL/LE) and reusable in any situation.

My solution entails a primary program called MYDECODE, which dynamically calls a routine MYARRAY*x*.

This last routine contains all the encoding tables we intend to use, and therefore this modular design allows me easily to write as many routines as I need for conversions.

The routine contains two arrays:

- Source – representing the data values that are to be converted – data input.

- Result – representing the converted data values to be used – data output.

The primary program does not need any changes to work with different codes.

The only parameters it requires are:

- I/O working area (in the example up to 500 bytes).
- Physical record length (5 bytes/zoned).
- The name of the routine with the conversion tables (8 bytes).
- A debug keyword (optional) to do additional verifications of the routine (5 bytes).

The program MYCODE can be called using any z/OS Language Environment standard. For testing purposes on a single record, I found that REXX in background mode offered the best combination for ease of use.

MYDECODE COBOL MAIN PROGRAM

```
       IDENTIFICATION DIVISION.
UPDAT PROGRAM-ID.     MYDECODE.
      ** ------------------------------------ **
      ** COMPILER OPTIONS: DYNAM,OPT          **
      ** ------------------------------------ **
       ENVIRONMENT    DIVISION.
       CONFIGURATION   SECTION.
       INPUT-OUTPUT    SECTION.
      ** ------------------------------------ **
       DATA           DIVISION.
       WORKING-STORAGE SECTION.
      ** ------------------------------------ **
       77  W-USER-RC                 PIC S9(4) USAGE BINARY VALUE Ø.
       77  W-WORK                    PIC X(8).

       Ø1  W-ARRAYUSR-LENGTH EXTERNAL PIC 9(3).
       Ø1  W-ARRAYUSR-SOURCE EXTERNAL PIC X(256).
       Ø1  W-ARRAYUSR-RESULT EXTERNAL PIC X(256).

       Ø1  W-MAIN-ARRAY-WORK.
           Ø3 W-MAIN-ITEM-WORK        PIC X
UPDAT        OCCURS 1 TO 5ØØ TIMES
               DEPENDING ON L-REC-LENGTH
```

```
                INDEXED BY W.

       Ø1  W-MAIN-ARRAY-RESULT.
           Ø3 W-MAIN-ITEM-RESULT      PIC X
              OCCURS 1 TO 256 TIMES
               DEPENDING ON W-ARRAYUSR-LENGTH
                 INDEXED BY O.

       Ø1  W-MAIN-ARRAY-SOURCE.
           Ø3 W-MAIN-ITEM-SOURCE
              OCCURS 1 TO 256 TIMES
               DEPENDING ON W-ARRAYUSR-LENGTH
                ASCENDING KEY IS  W-MAIN-KEY-SOURCE
                 INDEXED BY I.
              Ø5 W-MAIN-KEY-SOURCE    PIC X.
      ** ------------------------------------ **
       LINKAGE         SECTION.
      ** ------------------------------------ **
       Ø1 L-REC-PARM.
           Ø3 P-ITEM-PARM             PIC X
UPDAT        OCCURS 1 TO 5ØØ TIMES
               DEPENDING ON L-REC-LENGTH
                INDEXED BY L.

       Ø1 L-REC-LENGTH                PIC 9(5).
       Ø1 L-ARRAY-NAME-ROUT           PIC X(8).
       Ø1 L-DEBUG                     PIC X(5).
      ** ------------------------------------ **
       PROCEDURE      DIVISION
             USING L-REC-PARM L-REC-LENGTH L-ARRAY-NAME-ROUT L-DEBUG.
      ** ------------------------------------ **
       ML-START.
           IF     L-REC-LENGTH IS NOT NUMERIC
            OR     L-ARRAY-NAME-ROUT IS NUMERIC
             OR     L-REC-LENGTH = Ø
              MOVE   4Ø TO W-USER-RC
               GO TO  ML-END
           END-IF.

           CALL   L-ARRAY-NAME-ROUT.
           MOVE   W-ARRAYUSR-RESULT TO W-MAIN-ARRAY-RESULT.
           MOVE   W-ARRAYUSR-SOURCE TO W-MAIN-ARRAY-SOURCE.

           SET    W TO 1.
           MOVE   L-REC-PARM TO W-MAIN-ARRAY-WORK.

       ML-SEARCH.
           SEARCH  ALL  W-MAIN-ITEM-SOURCE
                 WHEN  W-MAIN-KEY-SOURCE (I) = W-MAIN-ITEM-WORK (W)
                  SET O TO I
```

```
                MOVE W-MAIN-ITEM-RESULT (O) TO W-MAIN-ITEM-WORK (W)
                 GO TO  ML-OK
            END-SEARCH.

       ML-ERR.
            MOVE  2Ø  TO W-USER-RC
            DISPLAY "-> E: ITEM NOT FOUND = " W-MAIN-ITEM-WORK (W).

       ML-OK.
           IF     W NOT = L-REC-LENGTH
            SET    W UP BY 1
             GO TO  ML-SEARCH
           END-IF.


       ML-END.
           IF     L-DEBUG  = "DEBUG"
            OR     W-USER-RC = 4Ø
               ACCEPT  W-WORK  FROM DATE YYYYMMDD
               DISPLAY "-> I> DATE         : " W-WORK
               DISPLAY "-> I> REC_INPUT    : " L-REC-PARM
               DISPLAY "-> I> REC_OUTPUT   : " W-MAIN-ARRAY-WORK
               DISPLAY "-> I> REC_LENGTH   : " L-REC-LENGTH
               DISPLAY "-> I> ARRAY_NAME   : " L-ARRAY-NAME-ROUT
               DISPLAY "-> I> DEBUG_VALUE  : " L-DEBUG
               DISPLAY "-> I> RETURN_CODE  : " W-USER-RC
           END-IF.

           MOVE W-MAIN-ARRAY-WORK TO L-REC-PARM.
           MOVE W-USER-RC TO RETURN-CODE.


           GOBACK.
```

MYARRAY0 COBOL SUBPROGRAM:

```
       IDENTIFICATION DIVISION.
UPDAT PROGRAM-ID.     MYARRAYØ.
      ** ------------------------------------ **
      ** COMPILER OPTIONS: DYNAM,OPT          **
      ** ------------------------------------ **
       ENVIRONMENT    DIVISION.
       CONFIGURATION   SECTION.
       INPUT-OUTPUT    SECTION.
      ** ------------------------------------ **
       DATA           DIVISION.
       WORKING-STORAGE SECTION.
      ** ------------------------------------ **
UPDAT Ø1  W-ARRAYUSR-LENGTH  EXTERNAL       PIC 9(3).
```

```
       Ø1  W-ARRAYUSR-SOURCE  EXTERNAL       PIC X(256).
       Ø1  W-ARRAYUSR-RESULT  EXTERNAL       PIC X(256).

       Ø1  W-SORTED-ARRAY-SOURCE.
UPDAT     Ø3  FILLER         PIC X       VALUE  "Ø".
UPDAT     Ø3  FILLER         PIC X       VALUE  "1".
UPDAT     Ø3  FILLER         PIC X       VALUE  "2".
UPDAT     Ø3  FILLER         PIC X       VALUE  "3".
UPDAT     Ø3  FILLER         PIC X       VALUE  "4".
UPDAT     Ø3  FILLER         PIC X       VALUE  "5".
UPDAT     Ø3  FILLER         PIC X       VALUE  "6".
UPDAT     Ø3  FILLER         PIC X       VALUE  "7".
UPDAT     Ø3  FILLER         PIC X       VALUE  "8".
UPDAT     Ø3  FILLER         PIC X       VALUE  "9".
UPDAT     Ø3  FILLER         PIC X(246)  VALUE  HIGH-VALUE.

       Ø1  W-UNSORTED-ARRAY-RESULT.
UPDAT     Ø3  FILLER         PIC X       VALUE  "/".
UPDAT     Ø3  FILLER         PIC X       VALUE  "A".
UPDAT     Ø3  FILLER         PIC X       VALUE  "B".
UPDAT     Ø3  FILLER         PIC X       VALUE  "C".
UPDAT     Ø3  FILLER         PIC X       VALUE  "D".
UPDAT     Ø3  FILLER         PIC X       VALUE  "E".
UPDAT     Ø3  FILLER         PIC X       VALUE  "F".
UPDAT     Ø3  FILLER         PIC X       VALUE  "G".
UPDAT     Ø3  FILLER         PIC X       VALUE  "I".
UPDAT     Ø3  FILLER         PIC X       VALUE  "H".
UPDAT     Ø3  FILLER         PIC X(246)  VALUE  HIGH-VALUE.
      ** ------------------------------------ **
       PROCEDURE      DIVISION.
      ** ------------------------------------ **
       ML-START.
UPDAT         MOVE 1Ø TO W-ARRAYUSR-LENGTH.

               MOVE W-SORTED-ARRAY-SOURCE   TO W-ARRAYUSR-SOURCE.
               MOVE W-UNSORTED-ARRAY-RESULT TO W-ARRAYUSR-RESULT.
       ML-END.
           GOBACK.
```

EXAMPLE JCL TO RUN MYDECODE WITH REXX

```
//........your jobcard.........................................
//** ------------------------------------------------------ **
//POTEMP   EXEC PGM=ICEGENER
//SYSUT1   DD   DATA,DLM=
    RECORD     = 'Ø98765432'
    LENGHT_REC = 'ØØØØ9'
    ROUTINE    = 'MYARRAYØ'
    FLAG       = 'DEBUG'
```

```
    ADDRESS  LINKPGM   "MYDECODE RECORD LENGHT_REC ROUTINE FLAG "

//SYSUT2   DD   DISP=(,PASS),UNIT=VIO,SPACE=(TRK,(1,1,1)),
//   DSN=&T(SAMPLE),DCB=(LRECL=8Ø,DSORG=PO)
//SYSPRINT DD   DUMMY
//** ------------------------------------------------------ **
//RXBATCH  EXEC PGM=IRXJCL,PARM='SAMPLE'
//STEPLIB  DD   DISP=SHR,DSN=your.loadlib
//SYSEXEC  DD   DISP=(OLD,PASS),DSN=&T
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   DUMMY
```

Softek, part of the Fujitsu group, has announced Softek Replicator, which runs on z/OS as well as AIX, Windows, HP-UX, Linux, and Solaris, and supports any storage array, such as those from EMC, HDS, H-P, IBM, or StorageTek.

The product replicates data writes to one drive array or disk to another across an IP link. The replication is done at the host level, not by the drive array controller.

For further information contact:
Softek, 1250 East Arques Avenue, M/S 317, Sunnyvale, CA 94085, USA.
Tel: (408) 746 7638.
URL: http://www.softek.fujitsu.com/en/products/replicator.

* * *

Advanced Software Products Group has announced Version 2.0 of ERQ (Easy RACF Query).

The product includes an online function to allow automated security administration. Using buttons, reports can be produced, RACF commands can be generated, and 'clean-up' tasks can be streamlined. There are also extensive help messages.

There is an API-type interface that retrieves RACF information for REXX and CLIST customized RACF applications.

For further information contact:
ASPG, 3185 Horseshoe Drive South, Naples, FL 34104, USA.
Tel: (239) 649 1548.
URL: http://www.aspg.com/erq.htm..

* * *

DataMirror has announced iFederate, which is designed to extract data from mainframe environments. Customers can access timely reports from a variety of mainframe data stores.

iFederate allows customers to define queries against a virtual database that consolidates common data storage formats including DB2, VSAM, ISAM, IMS, and SAM. iFederate can perform complex joins across any supported mainframe data store.

For further information contact:
DataMirror, 3100 Steeles Avenue East, Suite 1100, Markham, ON, Canada L3R 8T3.
Tel: (905) 415 0310.
http://www.datamirror.com/products/ifederate.

* * *

Data 21 has announced Version 3.0 of ZIP/390, its cross-platform compatible compression utilities and developers' API for z/OS and OS/390.

The product combines compression and encryption technologies with built-in TCP/IP communications (ie e-mail and FTP).

Version 3.0 offers improved performance and ease of use. A refinement in the product's UNZIP algorithm reduces UNZIP time on the mainframe an average of 30%. New features such as dynamic file allocation and generic file name support greatly simplify the JCL.

For further information contact:
Data 21, 3510 Torrance Blvd, Suite 300, Torrance, CA 90503, USA.
Tel: (310) 792 1771.
URL: http://www.data21.com/products/zip/default.asp.