205

MVS update

October 2003

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

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.

Some useful ISPF utilities

The following ISPF utilities are provided below:

- VIEWHELP – allows users to view the TSO HELP output in a dataset.
- VIEWDD – a routine to view a dataset given the DDname. It is used by VIEWHELP.
- MEMFIND – allows users to search multiple datasets for a particular member.
- MEMCHK – a routine that can be used to check whether a specific member is present in a dataset. It can be invoked directly as a command or a routine. Used by MEMFIND.
- MEMDISP – a command, when invoked with the dataset name as a parameter displays the members list panel.

VIEWHELP

VIEWHELP allow users to view the TSO HELP output in a dataset, which provides the following benefits:

- Allows searching for strings.
- Allows scrolling forward or backward.
- The information, if required, can be saved in a dataset.

Note that **TSO VIEWHELP <hlptopic>** is to be invoked within an ISPF environment, eg **TSO VIEWHELP ALLOC** or **TSO VIEWHELP LISTCAT**.

```
/*************************REXX*************************************\
*  REXX Program - VIEWHELP                                          *
*  REXX Program that shows the TSO HELP information in a VIEW PANEL  *
*  Used for allowing search and scrolling of HELP information.      *
\******************************************************************/
arg hlptopic parm
y = outtrap("hlp.",'*',"concat")
"help "hlptopic parm
```

```
if (POS('HELP NOT AVAILABLE',hlp.1) ¬= Ø) then
do
   err_msg_pos = pos('ENTER HELP',hlp.2)
   err_msg_pos = err_msg_pos + 5
   hlp.2 = insert('VIEW',hlp.2,err_msg_pos)
end
x = outtrap("off")
address ispexec "control errors return"
"alloc dd($hlpfl$) unit(vio) lrecl(8Ø) blksize(312Ø) dsorg(ps)",
     "space(1,1) track new reu"
if rc¬=Ø then
do
   if rc=12 then
   do
      say ' '
      say 'YOU ARE ALREADY IN VIEWHELP!!!!'
      say 'EXIT FROM THIS PANEL TO INVOKE VIEWHELP AGAIN'
   end
   exit
end
"execio * diskw  $hlpfl$ (stem hlp. finis"
if rc=Ø then
   address tso     "%viewdd $hlpfl$ "
"free dd($hlpfl$)"
address ispexec "control errors cancel"
exit
```

VIEWDD

The VIEWDD REXX routine can be used to view a dataset given its DDname. It is invoked by VIEWHELP. A view can be modified to browse or edit, depending on the requirement.

```
/**************************REXX***************************************\
*  REXX Routine - VIEWDD                                              *
*  Used to VIEW a dataset given the DDNAME                            *
\*********************************************************************/
arg ddn
address ispexec "lminit dataid("did") ddname(&ddn) enq(shr)"
if rc ¬= Ø Then
Do
   say 'LMINIT - Failed with RC ' || rc
   exit
end
/* If required, change the following to Browse or Edit */
address ispexec "view dataid("did")"
if rc ¬= Ø Then
Do
```

```
   say 'VIEW - Failed with RC ' || rc
end
address ispexec "lmfree dataid("did")"
return
```

MEMFIND

The MEMFIND command can be invoked to get a list of datasets having a particular member. It takes the dataset pattern with wild characters – similar to option 3.4 – and displays the list of datasets having the specific member.

The routine can be modified to return the list of datasets matching the pattern along with the required dataset attributes, or to filter the list of datasets based on their attributes – eg list of migrated datasets.

```
/**************************REXX*************************************\
*  REXX Routine - MEMFIND                                            *
*  INPUT:                                                            *
*           Dataset Pattern - with wild characters                   *
*           memname- Member to be searched for                       *
*  RETURNS:                                                          *
*           List of datasets containing the member                   *
*  Invokes MEMCHK routine to find if a member is present in a dataset*
\*******************************************************************/
arg dspattern memname
address ispexec
if dspattern = "" | memname = "" | dspattern = "?" then
do
   say "Command syntax is MEMFIND <Dataset Pattern> <Member to find>"
   exit
end
/* Get the list of datasets matching the given pattern */
"lmdinit listid("lstid") level("dspattern")"
if rc ¬= Ø Then
Do
   say 'LMDINIT - Failed with RC ' || rc
   exit
end
"lmdlist listid("lstid") option(LIST) dataset(dsvar) STATS(YES)"
if rc = 4 Then
Do
   say 'No DATASET matching this Pattern ' || dspattern
   exit
end
say 'Dataset Pattern is ' || dspattern
```

```
mcnt = Ø
do while rc=Ø
   /* check ONLY if the DATASET is not migrated and a PDS      */
   if ZDLMIGR = 'NO' & ZDLDSORG = 'PO' then
   do
      address tso "%memchk " dsvar memname "NODISP"
      if rc = Ø then
      do
         say 'Dataset ' || dsvar
         mcnt = mcnt + 1
      end
   end
   /* Get the next Dataset matching the pattern               */
   "lmdlist listid("lstid") option(LIST) dataset(dsvar) STATS(YES)"
end
if rc > 8 Then
do
   say 'LMDLIST - Failed with RC ' || rc
exit
end
if mcnt = Ø then
   say 'Member ' || memname || ' NOT FOUND in qualifying DATASETS '
else
   say 'Member ' || memname || ' is FOUND in ' || mcnt || ' Datasets'

say 'NOTE: The Migrated datasets are not considered '
"lmdlist listid("lstid")"
exit
```

MEMCHK

The MEMCHK command can be used to check whether a specific member is present in a dataset. It can be invoked directly as a command or as a routine.

```
/***************************REXX*************************************\
*  REXX Routine - MEMCHK                                             *
*  Checks whether a specific member is found in the dataset          *
*  Can be invoked as a TSO COMMAND or as a routine from programs     *
*  INPUT:                                                            *
*           dsname - fully-qualified dataset name                    *
*           memname- member to be searched for                       *
*           dispopt- "NODISP" value to be sent to avoid displays     *
*  RETURNS:                                                          *
*           Ø if the member is FOUND in the dataset                  *
*           1 if the member is NOT FOUND in the dataset              *
\********************************************************************/
arg dsname memname dispopt
address ispexec
```

```
if dsname = "" | memname = "" | dsname = "?" then
do
   say "Command syntax is MEMCHK <Dataset> <Member to find>"
   exit
end
retval = 1  /* Set the RETURN CODE to NOT FOUND as Default value     */
address ispexec
"lminit dataid("did") dataset('&dsname') enq(shr) org(orgds)"
if rc ¬= Ø Then
Do
   say 'LMINIT - Failed with RC ' || rc || ' for dataset ' || dsname
   exit
end
/* Check to ensure that the Dataset is a PDS */
if orgds <> "PO" then
do
   "lmfree dataid("did")"
   exit
end
"lmopen dataid("did") option(input)"
if rc ¬= Ø Then
Do
   say 'LMOPEN - Failed with RC ' || rc || ' for dataset ' || dsname
   exit
end
"lmmfind dataid("did") member(&memname)"
if rc > 8 Then
Do
   say 'LMMFIND - Failed with RC ' || rc
   exit
end
/* set the return value to Ø, if member is FOUND */
if rc = Ø Then
   retval = Ø
/* Display if the dataset if found or not - ONLY if NODISP is not set */
if dispopt <> 'NODISP' then
do
   if retval = Ø then
      say 'Member ' || memname || ' FOUND in ' || dsname
   else
      say 'Member ' || memname || ' NOT FOUND in ' || dsname
end
address ispexec "lmfree dataid("did")"
exit retval
```

MEMDISP

The MEMDISP command can be invoked with the dataset name as a parameter to display the members list panel.

The dataset name can be given with or without quotes, depending on whether you want the prefix to be included or not.

```
/**************************REXX*************************************\
*  REXX Routine - MEMDISP                                           *
*  Directly displays the MEMBER LIST of a given DATASET             *
*  Dataset name to be passed as command line argument               *
*  Dataset name can be with or without Quotes depending on the reqt. *
\*******************************************************************/
arg dsname
address ispexec
if dsname = "" Then
Do
   say 'Enter the Dataset Name as an Argument '
   exit
end
"lminit dataid("did") dataset(&dsname) enq(shr) org(orgds)"
if rc ¬= Ø Then
Do
   say 'Invalid Dataset '
   say 'LMINIT - Failed with RC ' || rc
   exit
end
if orgds <> "PO" then
do
   say 'Dataset ' || dsname || ' is not a PDS'
   "lmfree dataid("did")"
   exit
end
address ispexec "memlist dataid("did") member(*)"
if rc ¬= Ø Then
Do
   say 'MEMLIST - Failed with RC ' || rc
end

address ispexec "lmfree dataid("did")"
return
```

Sasirekha Cota
Tata Consultancy Services (India)

A fresh look at Java for OS/390 – enhancements in functionality, scalability, and performance

Java has been supported on the OS/390 platform since September 1997, and has been continually enhanced to meet e-business requirements. As originally ported to the OS/390 environment, JDK 1.1 had a number of limitations in efficiency, capability, and serviceability, and issues with locking, memory management, and use of system services. Since then Java functionality, resource management, and consumption and performance have been drastically improved.

The current supported releases of Java for OS/390 are Java 2 Technology Edition, SDK 1.3.1 (5655-D35), and IBM SDK 1.4 for z/OS, Java 2 Technology Edition (5655-I56). These Java products are full-function products that have passed the Java-compatible test suites, carry the Java-compatible logo, and provide the Java language and function for the OS/390 and z/OS platforms.

Java is seen as the programming language of the future with 'write once, run anywhere' capabilities. Developers are required to design applications with better performance, availability, and transactional and scalability features. This makes OS/390 a platform of choice to run new Java applications, because it provides the stability, availability, and reliability required by corporate business environments, for which it is critical that a variety of workloads be able to coexist without compromising these attributes.

HIGHLIGHTS

The highlights are:

- Java for OS/390 and z/OS is free and can be downloaded from the IBM Web site http://www-1.ibm.com/servers/eserver/zseries/software/java/getsdk13.html, or ordered from IBM in SMP/E installable format.

- The Java 2 Technology Edition became generally available in July 2000. The product provides the Sun SDK 1.3 APIs and is periodically updated with cumulative service and improvements.
- Current release of Java SDK V1.3.1 requires at least OS/390 V2.8 or z/OS. It will be serviced until December 2003.
- Latest release V1.4 will run only on z/OS V1.4 or z/OS.e V1.4 or higher. Version 1.4 will be serviced until September 2005.
- Java for OS/390 and z/OS requires use of OS/390 Unix Services.
- Current versions of OS/390 provide OS/390 Unix kernel as part of the OS/390 system itself.
- Unix System Services (USS) supports an entirely new file system for OS/390 called the Hierarchical File System (HFS).
- You can navigate the HFS by using the ishell TSO script or OMVS shell. Both options can be accessed from ISPF Option 6.
- Java for OS/390 and z/OS has connectivity to OS/390 and z/OS services and subsystems.
- To improve performance, Java runtimes include a just-in-time, or JIT, compiler for JVM.
- JVM is optimized for performance, reliability, function, and security.
- All Java threads are run multithreaded on an OS/390 TCB.
- Extensive reuse of existing data and applications:
  - Gateways – access to CICS and IMS
  - MQ – Java Messaging Support (JMS)
  - Java Native Interface – access to/from procedural programs

– Access to relational databases with Java Database Connectivity (JDBC) and SQL/J.

WHERE DO I START?

If you just inherited Java support on an OS/390 platform, you may want to see what is already installed**.** You can tell which version of Java you have installed by typing **java -fullversion** or **java -version.**

For example:

```
TEST:Userid:/u/userdir$ java –fullversion
java full version “J2RE 1.3.1 IBM OS/390 Persistent Reusable VM build
cm131s-20030510a”
```

Java 1.3.1 is usually installed in the /usr/lpp/java/J1.3 directory. You can check where Java is installed in your environment by typing **echo $PATH** while in the OMVS shell. This will return a PATH environment statement, showing Java CLASSPATH, along with other products installed, like TCP/IP, DB2 and so on.

Next, you need to know what environment variables are used by the JVM and USS.

Here are some key environment variables you will need to be aware of:

- JAVA_HOME – base directory for your Java for OS/390 installation. It is the directory where the Java command is located. This directory will be set during the installation process, so you do not need to set it manually.
- CLASSPATH – this is the regular Java CLASSPATH variable. It is used as a search path for Java classes. It consists of a list of directories and/or jar files and/or zip files separated by colons.
- JAVA_COMPILER – turns off the JIT (just-in-time) compiler or allows you to substitute an alternative JIT compiler. Please note that JIT must be turned off when debugging with the Remote Debugger.

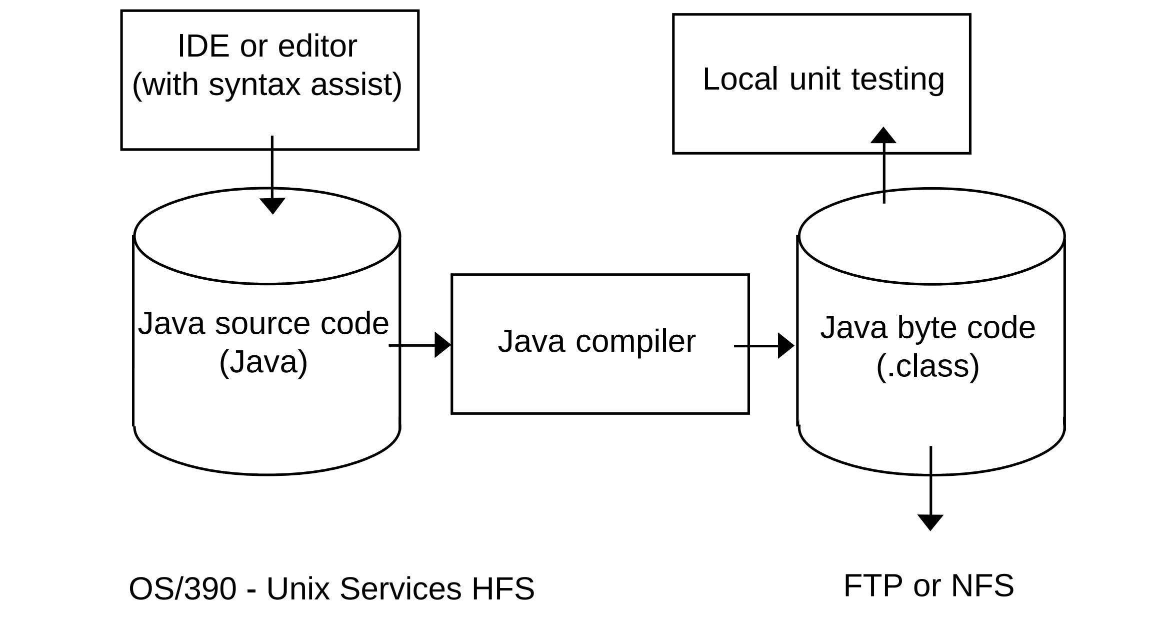


Figure 1: Java deployment process on OS/390 and z/OS

- LIBPATH – this is a USS environment variable containing a list of directories separated by colons that is used as a search path when loading DLLs from the HFS.

CREATING EXECUTABLE CODE WITH JAVA FOR OS/390

Next, let's take a quick look at how to create a sample Java program and create an executable on the OS/390 platform.

Java for OS/390 and z/OS includes JVM, debugger, classes, and compiler. It creates Java bytecodes which are *not* directly executable OS/390 instructions. They are a bytecode for a virtual machine, which must be 'executed' or interpreted by that virtual machine.

Here is an example of a very simple Java program, HelloOS390World:

```
* HelloOS39ØWorld class implements an application that displays
* "Hello, welcome to Java on OS/39Ø world!"
*    to the standard output device.
*/
class HelloOS39ØWorld {
      public static void main(String[] args) {
          System.out.println("Hello, welcome to Java on OS/39Ø world!");
//display string
}
```

Figure 1 illustrates the Java deployment process on OS/390 and z/OS.

Steps to run the HelloOS390World Java program are:

1 Save Java source as HelloOS390World.java.

2 Compile – javac HelloOS390World.java.

   Please note that HelloOS390World.class will be created by the class statement above. The Classname defined in the source must match the saved program name.

3 Execute – java HelloOS390World.

This should return at the prompt:

```
Hello, welcome to Java on OS/390 world!
```

COMPILING A JAVA PROGRAM IN BATCH

Java programs can be compiled using an interactive OMVS TSO session or in batch mode. Below is an example of JCL that can be used to run the BPXBATCH program to compile Java program:

```
//COMP     EXEC PGM=BPXBATCH,
//             PARM='sh javac HelloOS39ØWorld.java'
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//STDOUT   DD  PATH='/usr/lpp/java/IBM/J1.3/javapgm.out',
//             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//             PATHMODE=SIRWXU
//STDERR   DD  PATH='/usr/lpp/java/IBM/J1.3/javapgm.err',
//             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//             PATHMODE=SIRWXU
//STDENV   DD  DUMMY
//*
//COMPOUT   EXEC PGM=IKJEFTØ1,DYNAMNBR=3ØØ,COND=EVEN
//SYSTSPRT  DD SYSOUT=*
//HFSOUT    DD PATH='/usr/lpp/java/IBM/J1.3/javapgm.out'
//HFSERR    DD PATH='/usr/lpp/java/IBM/J1.3/javapgm.err'
//STDOUTL   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=133,BLKSIZE=137)
//STDERRL   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=133,BLKSIZE=137)
```

```
//SYSPRINT  DD SYSOUT=*
//SYSTSIN   DD DATA,DLM='/>'
 ocopy indd(HFSOUT) outdd(STDOUTL)
 ocopy indd(HFSERR) outdd(STDERRL)
/>
//
```

Here is another example of how BPXBATCH can be used for housekeeping, deleting application logs over a week old – which is a very common use of the batch interface:

```
//TSO    EXEC PGM=IKJEFTØ1,REGION=ØM
//*
//*     Delete all old log files from backup folder
//*
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD   *
 BPXBATCH SH find +
   /usr/myApplication/backuplogs +
              -type f -mtime +7 -exec rm -f {} \;
```

JAVA NATIVE INTERFACE (JNI)

In order to better understand how Java on OS/390 can be used, it is helpful to be aware of the Java Native Interface (JNI).

Java defines a standard feature for the Java platform called the

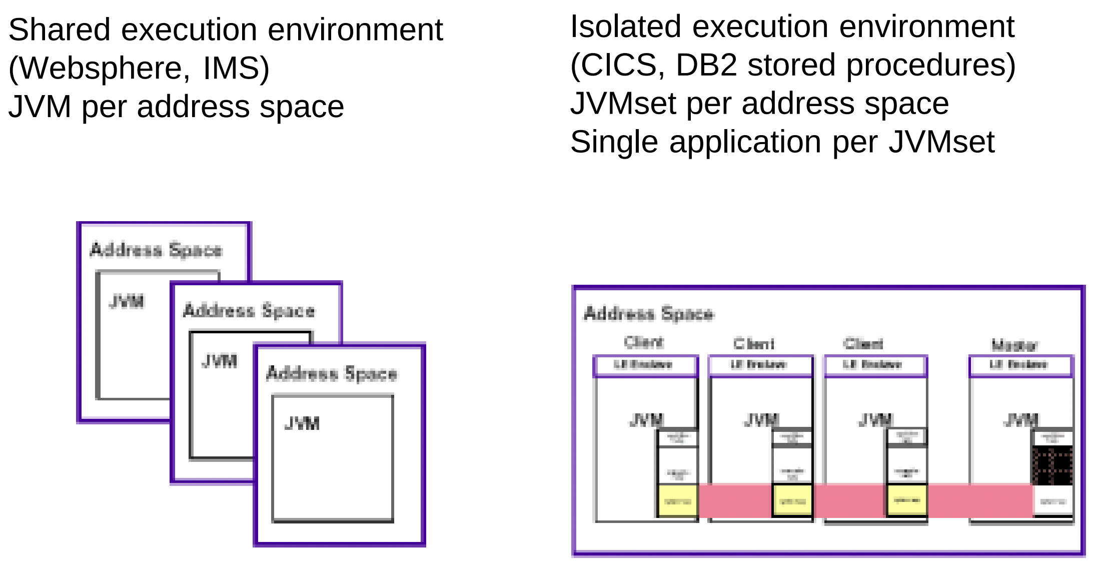


Figure 2: Shared and isolated execution environments

Java Native Interface (JNI). This is a series of APIs and rules, allowing communication between Java programs and non-Java programs. This makes it possible for programmers who want to integrate Java into their applications to do so without having to completely redesign them by shielding them from the most common machine-level details. JNI allows access to Java fields and methods in a standard way, so that you can use the values in native programs. Although all JVMs are required to support the JNI, a particular JVM is also permitted to implement its own, possibly more efficient, non-portable techniques for accessing Java objects. Using the JNI will allow your native programs to operate properly with any JVM that runs under OS/390.

JVM ALLOCATION

JVM can run in a shared or isolated execution environment, depending on the product running on OS/390.

CICS and DB2/390 use multiple JVMs per address space to achieve scalability and run a single application per JVM at any one time. WebSphere and IMS, on the other hand, use a single JVM per address space. WebSphere for z/OS creates multiple long-running JVMs per image.

A JVM running in persistent reusable mode is a member of a collection of JVMs called a JVMset. These infrastructures provide high performance and application isolation for environments where many short-running applications are executed. CICS and DB2 with stored procedures use this architecture.

The contrast between shared and isolated execution environments is illustrated in Figure 2.

JVM HEAPS

The next step to exploring the JVM world is to learn about the heaps used by JVM and how they are managed.

System heap

The (Java) system heap contains class objects that persist for

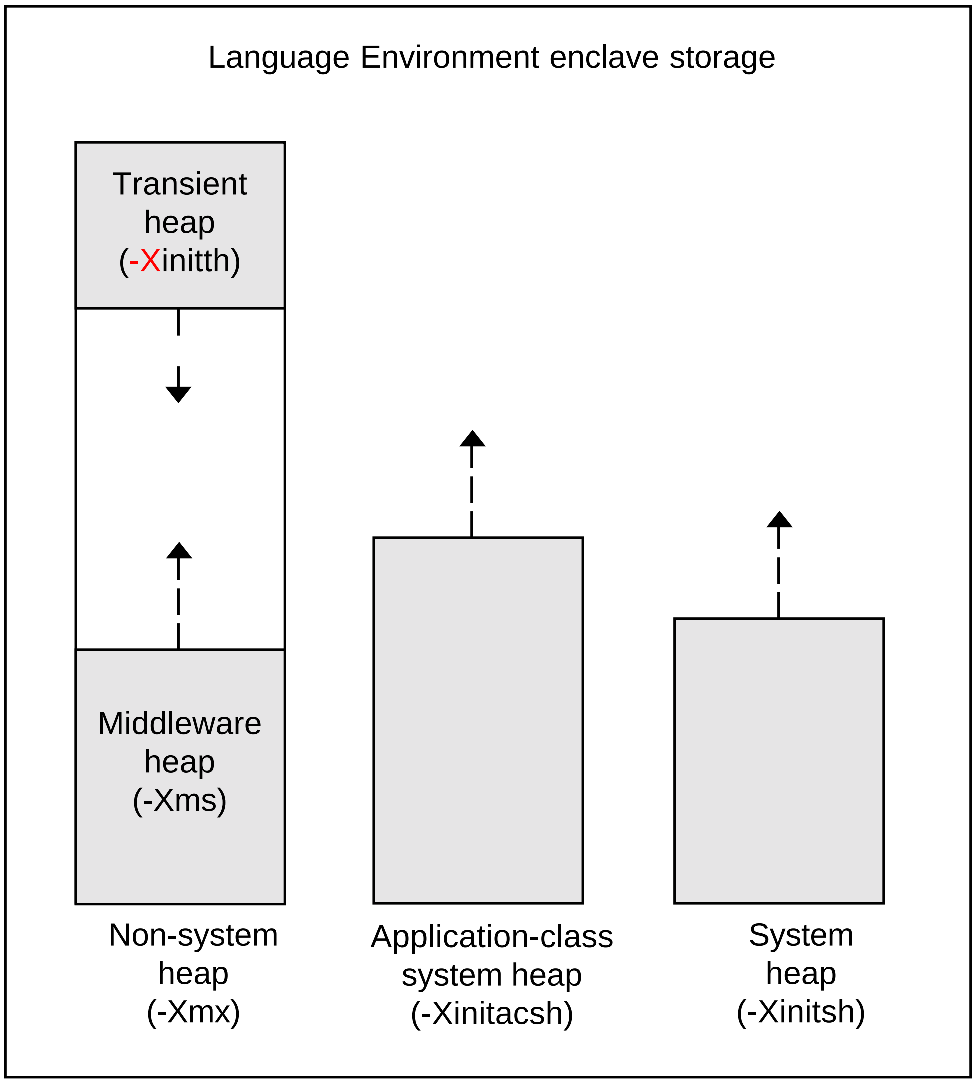

Language Environment enclave storage
Transient heap (-Xinitth)
Middleware heap (-Xms)
Non-system heap (-Xmx)
Application-class system heap (-Xinitacsh)
System heap (-Xinitsh)

Figure 3: Heaps

the lifetime of a JVM. Since the objects in the system heap persist for the life of the JVM, there is never any garbage in the system heap, so garbage collection never runs against the system heap.

Transient heap

A transient heap is the portion of a Java heap used for objects that exist only for the life of a single transaction. The transient heap exists only in JVMs running in persistent reusable mode.

Middleware heap

The middleware heap is a portion of the heap when using resettable mode. It is used for objects that have a lifetime greater than a single transaction. When running in non-resettable mode, the middleware heap is the entire heap.

The middleware heap is allocated from low storage in the non-system heap and expands upwards; the transient heap is allocated from high storage in the non-system heap, and expands down towards low storage. They can expand only until the two heaps meet.

Figure 3 illustrates the heaps allocation process.

JVM STORAGE USAGE

Now that we know about JVM heaps, let's take a look at how JVM storage is used.

When a request is made by Java Application to create a thread via the **pthread_create()** command, MVS TCB allocates storage for this request. The storage used per thread is controlled by the **-oss** flag. It is obtained in private storage above 16MB. The size of storage used per thread for the native stack is controlled by the **-ss** flag.

JVM manages the heap size of the storage allocated. The size of the heap is controlled by the **-Xmx** and **-Xms** values. The **-Xmx** value is 'malloced' in one go at initialization, but it isn't touched, so the unused pages remain in FREF state. Because it is obtained via **malloc**, it ends up being GETMAINed by LE in Supbool 2, Key 8. This is the area of storage where your objects are allocated, and is subject to the attentions of the garbage collector.

The JVM throws a java.lang.OutOfMemory exception when the heap is full and is unable to find space for object creation. This is analogous to the abend878 rc10 most of us have seen.

JVM HEAP SIZE PARAMETERS RELEVANT TO TUNING – JAVA COMMAND LINE OPTIONS

Below is a brief description of some of the JVM heap size parameters relevant to tuning. Some have already been mentioned in this article.

Please note that these parameters start with -X, refer to a non-standard Java interpreter option, and are unique to IBM and/or z/OS.

- -**Xms** – in resettable or non-settable modes, this option sets the initial size of the middleware heap within the non-system heap. If this option is not specified, a value of 1MB is used.
- -**Xmx** – in non-resettable mode sets the maximum heap size. If this option is not specified, the default value is used. In resettable mode, this option sets the maximum size for the combined middleware and transient heaps. The default value for **-Xmx** is 64MB. This is one of the key tuning parameters.
- **-Xinitacsh** – sets the initial size of the application-class system heap. In the persistent reusable JVM, classes in this heap exist for the lifetime of the JVMset. They are reset during a ResetJavaVM(), and so are serially reusable by applications running in the JVM. There is only one application-class system heap per persistent reusable JVM. In non-resettable mode, this option is ignored. For example:

  ```
  -Xinitacsh256k Default: 128 KB
  ```

- **-Xinitsh** – sets the initial size of the system heap. In the persistent reusable JVM, classes in this heap exist for the lifetime of the JVM. They are unaffected by a ResetJavaVM(), and so are serially reusable by applications running in the JVM. The system heap is never subjected to garbage collection. The maximum size of the system heap is unbounded. For example:

  ```
  -Xinitsh256k Default: 128 KB
  ```

- -**Xinitth** – in resettable mode, this sets the initial size of the transient heap within the non-system heap. If this is not specified and **-Xms** is, the initial size is taken to be half the **-Xms** value. If **-Xms** is not specified, a value of half the platform-dependent default value is used. In non-reusable mode, this option is ignored because there is no transient heap. For example:

```
-Xinitth2M Default: 1000 KB/2 = 500 KB
```

MANAGING LE STORAGE

When you run a Java application under OS/390, it creates a thread; the Java virtual machine creates a corresponding system

Option	*IBM-supplied default (as of R10)*	*Java 1.3 default*	*Purpose*
ANYHEAP	(16K,8K,ANYWHERE, FREE)	(2MB,512K,ANYW HERE,KEEP)	Allocates library heap storage above or below 16MB line.
BELOWHEAP	(8K,4K,FREE)	(8K,4K,FREE)	Allocates library heap storage below 16MB line
HEAP	(32K,32K,ANYWHERE, KEEP,8K,4K)	(8M,2M,ANYWH ERE,KEEP)	Allocates storage for user-controlled dynamically-allocated variables
LIBSTACK	(4K,4K,FREE)	(1K,1K,FREE)	Used by library routine stack frames that must be below 16MB

Figure 4: LE runtime settings in R10 and Java 1.3

thread. For each thread an application creates, the OS/390 operating system allocates storage and manages it with Language Environment (LE).

LE provides a common runtime environment for Java Applications.

STACK	(128K,128K,BELOW, KEEP,512K,128K,OVR)	(48K,16K,ANYWHERE,KEEP)	Used by library routine stack frames that can reside anywhere in storage.
STORAGE	(NONE,NONE, NONE,8K)	(NONE,NONE, NONE,1K)	Controls the initial content and amount of storage reserved for the out-of-storage condition.

Figure 5: STACK and STORAGE

It handles all the runtime services, such as message handling, condition handling, storage management, and math functions that Java applications use.

In order to optimize Java technology, it helps to have a good understanding of LE storage management.

To tune the stack size to the optimum value, you can turn on LE's storage report generation with the RPTSTG runtime option:

```
export_CEE_RUNOPTS="$_CEE_RUNOPTS:rptstg(on)
```

This option monitors stack usage as your program runs, giving you an insight to how much storage is being used, so you can set stack size appropriately. Please beware that RPTSTG(ON) will create additional overhead and should not be used in a production environment.

For optimal performance results, set the initial stack segment large enough to satisfy all requests for stack storage, but do not allocate it, since this may impact performance of other applications running on the machine. As the Java virtual machine allows you to override its defaults by exporting environment variables, you can export the right runtime option and optimize the stack size for a specific application with the following:

```
export_CEE_RUNOPTS="$_CEE_RUNOPTS:stack (256K,16K,ANYWHERE,KEEP)"
```

Figure 4 compares standard LE runtime settings in R10 and the Java 1.3 platform.

Here are some general suggestions and things to bear in mind when you are considering tuning LE parameters:

- HEAP – size depends very much on the application load. The initial heap size (8MB) should be at least as large as the sum of the **-Xmx** and **-oss options**, plus it will also be used for storage for internal control blocks and JITted code. A sufficiently large initial size avoids the overhead of LE stack expansion.
- STACK – option defines the initial stack segment size and increment size if a segment overflows; see Figure 5.
- STORAGE – should not be set to a value other than NONE because this may affect performance.
- ANYWHERE – has to be specified to avoid storage allocation BELOW the 16MB line.
- LIBSTACK – storage is always BELOW the 16MB line; it should be kept as small as possible.

JVM HEAP SIZE TUNING BASED ON THE APPLICATION

Tuning JVM heap size for a Java application running on OS/390 should be approached based on how JVM is allocated (shared or isolated) and specific storage requirements of the application.

Here are suggestions for WebSphere on Z/OS V4.0.1 and V5.0.

For WebSphere with DB2 on z/OS, IBM recommends JVM heap size ranges from 300MB to 400MB. The default size WebSphere V4.0.1 is 256MB. SQLJ will benefit from a larger JVM heap size than the default setting and performance can be improved by increasing the JVM heap size value. It also should be noted that a proportion of the JVM heap is given over to WebSphere objects.

Large WebSphere for z/OS shops should set JVM heap size as high as 512M. Here is an example of JVM HEAP size parameters you can specify in WebSphere environment file:

```
JVM_HEAPSIZE=512
JVM_MINHEAPSIZE=512
```

Please be aware, per PQ77021, WebSphere for z/OS V5.0 came out in error with a default JVM heap size of 48MB for an application server region, which is not enough for most production applications.

CICS TS 2.2 EJB server greatly differs from WebSphere for z/OS in JVM allocation.

CICS TS 2.2 EJB server can allocate multiple JVMs, each running on its own TCB, up to MAXJVMTCBS TCB limit (default is 5). Multiple transactions can be executed serially in a single long-running JVM, by using persistent reusable JVMs. The serial reuse of a JVM for multiple transactions, while resetting the JVM to a known state between each transaction, provides isolation without paying the high cost of a full JVM initialization for every transaction. There is only one transaction using a JVM at any one time.

The recommended values for CICS TS 2.2 are:

- **Xmx= 60M** to decrease GC collection and CPU overhead associated with it.
- **Xms=30M** to avoid unnecessary heap size expansion, also causing additional CPU overhead.
- **MAXJVMTCBS** – JVM TCB limit (set in SIT) should not be over-allocated. Setting it to 7 is recommended to prevent exhausting LE storage allocation both above and below the line.

It should be noted that CICS is very efficient in queueing JVM requests. The overhead and memory allocation for creating a new JVM (and TCB associated with it) is much higher than the overhead of JVM request, queueing up for available JVM.

CICS Transaction Gateway on OS/390 should not be confused with CICS TS when it comes to JVM heap size. It uses one JVM per address space, like WebSphere for z/OS. Here is an example of recommended heap size setting on CTG start-up script:

```
exec java -Xmx350M -Dcom.ibm.ctg.cicscli="$CICSCLI"
com.ibm.ctg.server.Jgate
```

PERFORMANCE

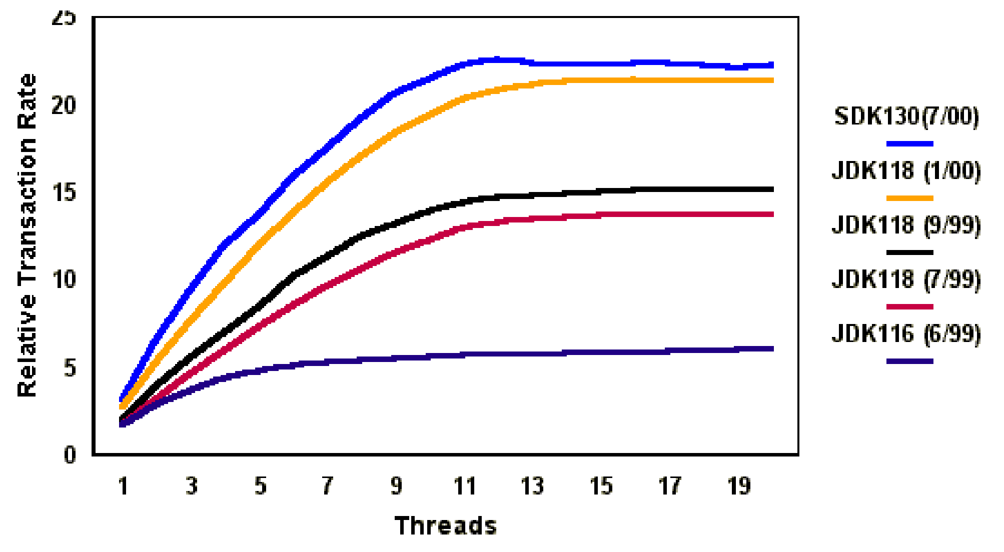
Performance has been a key requirement for Java on OS/390. Each SDK release has addressed performance issues. The latest release, SDK 1.4, has substantial improvements in performance.

SDK V1.4 takes advantage of enhanced z/OS linkage capabilities (XPLINK) for greatly improved performance. This enhancement is completely transparent to the end users. However, any application code that creates a JVM itself and interacts with the JVM via JNI or any other 'call' interface must create the LE enclave specifying that the LE should set up an XPLINK environment, which includes an XPLINK-specific runtime library and stack format. This XPLINK LE enclave must be in place prior to creating the JVM. This can be accomplished by setting a runtime option (XPLINK(ON)), or recompiling the launching application code to be XPLINK. IBM's Redbook SG245991 *XPLink: OS/390 Extra Performance Linkage* has these options and a description of the XPLINK technology.

Performance suggestions for JVM runtime environment

If you are running high-volume Java applications on OS/390, you need to be aware of the tuning you can do to improve performance and to support more Java threads. The following recommendations apply to a Java application needing more than 100 threads:

1 Check your system-wide thread limit in member BPXPRM*xx* in SYS1.PARMLIB. It contains system-wide options, **MAXTHREADS** and **MAXTHREADTASKS**, which control the number of threads that can be started in a process. These values must be set high enough to support the user who needs the most concurrent threads.



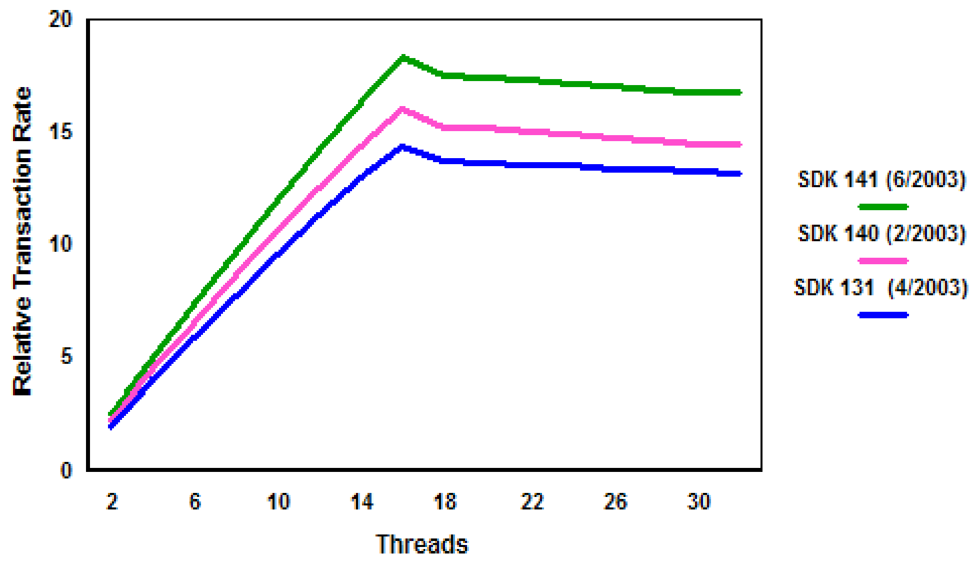


Figure 6: Application performance comparisons

2 LE runtime options may need to be adjusted, if you are running more than 400 threads in an address space.

3 Java runtime storage options can be used to control the storage used by your application. The key option for controlling the number of Java threads is **-ss**.

4 Another parameter that may benefit from tuning is minimum

native stack size for any Java thread. Current levels of the OS/390 JVM allocate this minimum stack size for each thread started, but many threads don't need a 256KB native stack. You can fit more threads in a smaller region by setting this value to one lower than the default. Please beware that, if you start and stop threads frequently, setting this value too low can cause lots of extra GETMAINs and FREEMAINs, so the parameter setting depends on the application.

5 Region size limits the virtual storage that a user address space (process) can consume. Since each Java thread requires some virtual storage, setting the region size too low in an address space running a Java application limits the number of concurrent threads that can run in that application. In general, region limits are inherited from a parent process to a child process. When a TSO user invokes the OMVS command to access USS, forked children will have the same region limits as the TSO address space. This region limit is set in the SIZE option on the TSO LOGON panel.

HISTORY OF PERFORMANCE ENHANCEMENTS

To get a real appreciation for performance enhancements for Java for OS/390, Figure 6 shows application performance comparisons made by IBM, starting with SDK 1.1.6 through SDK 1.4.

As you can see, Java for OS/390 has come a long way – it is now able to process much higher throughput, while using fewer resources.

CONCLUSION

In today's competitive and customer-oriented world, selecting an application server that provides good performance and scalability is crucial to the success of e-business applications.

Java has become a language of choice for most new application development. Using Java reduces programming costs and gives developers the freedom to deploy their applications on any platform, and improves time to market by exploiting Java technologies and allowing developers to use the most advanced application development tools.

Today developers have a choice of developing new Java applications on distributed platforms and then deploying them on OS/390 for production usage. Java technology on the OS/390 continues to improve and offers developers a platform with great availability, scalability, and much improved performance and resource usage.

There are still a lot challenges on OS/390 – managing and trouble-shooting Java applications will be the subject of my next article.

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

System programming C facilities

A little-known feature of the z/OS C programming language is System Programming C Facilities (SPC). As the name implies, this is a C-language subset that gives access to low-level functionality, eg registers. This allows programs written with SPC to be used in many situations as replacements for Assembler programs – an important consideration given the increasing difficulty in obtaining experienced Assembler programmers. Because SPC functions run without Language Environment, SPC programs perform better than normal C programs.

Major features (and restrictions) of the SPC functions include:

- Only C (not C++) functionality. However, this satisfies the requirement for high-performance.
- No Language Environment functionality. This means LE

programs (eg COBOL programs) cannot be invoked directly; the **system()** function must be used to invoke such programs.

- No CICS capability.
- No HFS support.
- No XPLINK support.
- **argc** and **argv** cannot be used to reference the passed arguments (the arguments must be fetched from the passed argument list).

APPLICATION AREAS

There are two main application areas for SPC programs:

- As subprograms (which includes operating system exits, etc).
- As a main program (so-called freestanding program), although the advantages over the complete C language capabilities have less weight here.

An SPC subprogram (exit) is subject only to the following environment restrictions when it is invoked:

1 R13: address of a 72-byte register save area.

2 R15: entry address.

3 R14: return address.

The **__xregs()** function can be used to obtain the register contents on entry. If the standard calling convention is used, register 1 can be used to obtain the arguments.

COMPILER OPTIONS

The NOSTART compiler option must be set. This precludes CEESTART from being used for the initialization.

COMPILER STATEMENTS

The following compiler statements can be used:

- #pragma environment(*functionname[*,nolib*]*) – the *#pragma* environment statement specifies the environment for a program that will run without the C environment. The *nolib* option specifies that the C library will not be used. *functionname* is the call name.

Note: SPC as standard supplies the following C functions. The first group is provided as built-in functions:

```
abs(), fabs()
memchr(), memcmp(), memcpy(), memset(), cds(), cs()
strcat(), strchr(), strcmp(), strcpy(), strlen(), strrchr()
```

The appropriate linkage editor INCLUDE member (eg EDCXMEM) must be specified for the functions of the second group:

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

LINKEDIT INCLUDES (FROM THE SCEESPC LIBRARY)

The appropriate INCLUDE statement (as a CEESTART replacement) must be specified for a freestanding application; this name must also be specified for the ENTRY statement:

- EDCXSTRT – do not provide z/OS C library functions.
- EDCXSTRL– provide z/OS C library functions.

Because the library member for the subprogram prologue code is fetched implicitly, the INCLUDE name does not need to be specified explicitly.

LINKEDIT ENTRY

The appropriate ENTRY statement must be specified:

- EDCXSTRT or EDCXSTRL for a freestanding program.
- functionname for a subprogram.

PROGRAM ENTRY

The usual parameter convention for C programs cannot be used.

PROGRAM EXIT

- Freestanding program – numeric value can be set return or **exit()**.
- Subprogram – no restriction. The value must be set using the **exit()** function.

SPC FUNCTIONS

An spc.h header file is required:

```
__xregs()
```

Gets register contents on entry to the program:

```
int __xregs(int register)
```

where *register* = register number (0 ... 15), for example:

```
__xregs(1); /* get register 1 contents */
```

SAMPLE PROGRAM 1 (SUBPROGRAM)

The TEST1 Assembler program calls TESTSPC (an SPC program) with a parameter. The function value is used as the program return code.

Calling program

```
TEST1    CSECT
TEST1    AMODE 31
TEST1    RMODE 24
* initialize addressing
         STM   14,12,12(13)    save registers
         BASR  12,Ø            load base register
         USING *,12            set base register
         LA    15,SA           address of save area for called subpgm
         ST    13,4(15)        backward ptr
         ST    15,8(13)        forward ptr
         LR    13,15           address of new save area
         CALL  TESTSPC,(P1)    invoke as external program
```

```
* R15: program return code
         L     13,4(13)         restore the address of the old save area
         RETURN (14,12),RC=(15)
* data areas
SA       DS    18F              save area
P1       DC    H'4',CL4'DXXT'   parameter
         END
```

Called subprogram

Task – display the passed argument and use the argument length as return value:

```
#pragma environment(TESTSPC)

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <spc.h>

int len;
char parm[1Ø1];

int TESTSPC()
{
  struct parm {
    short parmlen;
    char  parmdata[1];
  };

  typedef struct parm EXECPARM;
  EXECPARM *pparm, **ppparm;

  ppparm = (EXECPARM **)__xregs(1); /* get R1 */
  pparm = *ppparm;
  /* Get length of EXEC parameter */
  len = pparm->parmlen;
  if (pparm->parmlen > sizeof(parm)-1) {
    puts("Parameter too long. Job terminated");
    printf("Parmlen: %hd\n",pparm->parmlen);
    exit(-8);
  }
  if (pparm->parmlen == Ø) {
    puts("Parameter omitted. Job terminated");
    exit(-12);
  }
  /* move Exec parameter data to work area */
  memcpy(parm,pparm->parmdata,pparm->parmlen);
  parm[pparm->parmlen] = ØxØØ;
  printf("Parmdata: %s\n",parm);

  exit(len); /* OK: normal end */
```

Corresponding linkage editor statements:

```
 INCLUDE SYSLIB(EDCXEXIT) SPC(exit())
 ENTRY TESTSPC
 NAME TESTSPC5(R)
```

SAMPLE PROGRAM 2 (FREESTANDING PROGRAM)

Sample program 2 is a trivial program. What is important is the associated linkage editor statements:

```
#include <stdio.h>

int main()
{

  puts("TESTSPC5 start");
  return 1ØØ;

}
```

Corresponding linkage editor statements:

```
 INCLUDE SYSLIB(EDCXSTRL)
 ENTRY EDCXSTRL
 NAME TESTSPC6(R)
```

PRACTICAL EXAMPLE

The example is a panel exit for an ISPF application. The argument list passed to the program (DMPGM03C) contains eight entries.

The program displays the field names and associated values of the passed variables (assumed to be two, PNAME and PNO, respectively). The PNO field is checked to make sure that its value is a multiple of three; the DMMMM001 message is issued if this is not the case (passed back as the fourth argument).

Note: to avoid over-complication, the field lengths are not checked for overflow.

```
#pragma environment(DMPGMØ3C)
/* Panel exit routine:
   display the field names and contents,
   check that the personnel number (first data field)
   is a multiple of 3. */

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <spc.h>

DMPGMØ3C() {
  int **parme;
  char *p1; /* A(EXDATA) */
  char *p2; /* panel name (CL8) */
  char *p3; /* panel section (CL1) */
  char *p4; /* message-id (CL8) */
  int  *p5; /* dimension of VARNAME and VARLEN (=n) */
  char *p6; /* vector of dialogue variable names (nCL8) */
  int  *p7; /* array of variable lengths (nFL4) */
  char *p8; /* vector of dialogue variable values */

  int len;
  int pno, rem;
  int rc;
  char name[9];
  char *ptr;
  char temp[256];

  parme = (void *)__xregs(1);
  p1 = (char *)parme[Ø];
  p4 = (char *)parme[3];
  p6 = (char *)parme[5];
  p7 = (int *)parme[6];
  p8 = (char *)parme[7];

  /* display field names */
  memcpy(name,p6,8);
  name[8] = ØxØØ;
  puts(name);
  p6 += 8;
  memcpy(name,p6,8);
  name[8] = ØxØØ;
  puts(name);

  /* display field contents */
  ptr = p8; /* set pointer to start of vector */
  len = p7[Ø]; /* field length */
  memcpy(temp,ptr,len);
  temp[len] = ØxØØ;
  pno = atoi(temp);
  printf("pno:%d\n",pno);

  ptr += len;
  len = p7[1];
  memcpy(temp,ptr,len);
  temp[len] = Øx00;
  printf("name:%s\n",temp);

  /* validate <pno> field */
  rem = pno%3;   /* remainder */
  if (rem == Ø) { rc = Ø; } /* zeroise <rc> */
  else {
    memcpy(p4,"DMMMMØØ1",8);
    rc = 8;
  }

  exit(rc);
}
```

The associated ISPF panel definition:

```
)BODY
%---------- Display Panel ----------+
%COMMAND ===>_ZCMD
+
+Persno. ===>_PNO +
+Name     ===>_NAME              +
+
)PROC
IF (.RESP EQ ENTER)
   PANEXIT ((PNO,NAME),LOAD,DMPGMØ3)
)END
```

Systems Programmer
(Germany) © Xephon 2003

Catalogued non-existent datasets

Sharing DASD volumes without sharing catalogs can have some unexpected and unwelcome results – even when only one volume is shared, such as an IODF volume, as in this example.

RESEARCHING THE ERROR MESSAGE

One of the nightly back-up production jobs failed, as it had done for the past three nights. But this was the first I had heard about it. I was given a screen print of the console log with the following message circled:

```
ADR38ØE (ØØ1)-DTDSC(Ø4), DATA SET SYS1.RACF192.MASTER.OLD NOT PROCESSED,
13-ØØØ8
```

A quick look at the JCL confirmed that ADR messages are from DFSMSdss (program ADRDSSU):

```
//STEP4  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//OUT1     DD DSN=OPRP.ONSITE.DAILY.RACFBCK(+1),
//            UNIT=VTAPE,DATACLAS=DCVTS,
//            DISP=(,CATLG,DELETE),DCB=(SYS1.MODEL)
//OUT2     DD DSN=OPRP.OFFSITE.DAILY.RACFBCK(+1),UNIT=ATL,
//            DISP=(,CATLG,DELETE),DCB=(SYS1.MODEL),
//            VOL=(,RETAIN,REF=*.STEP1.OUT1),LABEL=2,
//            DATACLAS=DC359Ø
//SYSIN    DD *
   DUMP OUTDD(OUT1,OUT2) -
      DATASET(INCLUDE(SYS1.RACF*.**,SWFP.RACF*.**))
```

Looking at the relevant manual, *z/OS MVS System Messages*, under message ADR380E with reason code 13-xxxx, the explanation reads:

Retrieving the extents from the VTOC failed. xxxx is the obtain error code. See z/OS DFSMSdfp Advanced Services for these codes.

The *Application Programmer Response* section gives the ubiquitous ‘Contact your system programmer’ reply. A single System Programmer Response is given for all reason codes:

On a RESTORE, dump the VTOC track records on the input that are at the beginning of the file.

The 0008 in 13-0008 is referred to as the ‘obtain error code’, so Obtain is looked up in the Index of *DFSMSdfp Advanced Services* and one of the two entries is to a page with a figure entitled *DADSM OBTAIN Return Codes*. The description for 8 reads:

The format-1 DSCB was not found in the VTOC of the specified volume.

What? That is just a fancy way of saying that the dataset is not listed in the VTOC for the DASD volume indicated in the catalog. In its simplest form, the dataset does not exist on disk even though it is listed in the catalog.

FURTHER INVESTIGATION

Using ISPF option 3.4 (DSLIST), the offending dataset is listed in default view, with dataset name and volume. Typing an I next to it and hitting *Enter* displays *Dataset Not Found* in the upper-right corner of the screen. Hitting PF1 for further explanation, a box appears, reading:

```
'SYS1.RACF192.MASTER.OLD' not on volume 'IODFØ1'.
```

Hitting *Enter*, the box disappears and the ‘I’ remains to the left of the dataset name. To prove that the dataset really does not exist, hit PF3 to get back to ISPF 3.4, where the original Dsname Level specification still appears, and simply enter the DASD VOLSER, IODF01 in this case, into the volume serial field, and hit *Enter*.

It may look like the same process, but it is not. In the first use of ISPF 3.4, without the volume serial field filled in, the search for datasets was made using the catalog. But this time, with volume serial specified, the VTOC of the specified DASD volume is searched. And the dataset in question does not appear.

UNCATALOGUING

At this point, you can go back to the original DSLIST, without volume serial specified, and use the U line command next to the catalogued, but non-existent, dataset, to uncatalogue it. This time, the message in the upper-right corner reads *Data Set Uncatalogued*.

If you make a mistake and uncatalogue the wrong dataset, no harm is done. Simply use the C command to re-catalogue the dataset. But using the C command on the non-existent dataset (while you can still see it, before refreshing the DSLIST) will give you the same *Data set does not exist* message in the upper-right corner.

One caveat: this method will not work with VSAM files. Instead, when no VSAM data or index component exists on DASD, you can delete the catalogue entry for a cluster, alternate index, or page space by using IDCAMS DELETE NOSCRATCH.

WHY?

As mentioned at the outset, this problem occurs because multiple z/OS systems (LPARs) share the same IODF volume. But not all of those systems share the same catalog.

In theory, there should never be a problem, because when the dataset is first allocated, it will only be in the catalog used by the system where the allocation occurred. That should make the dataset invisible to any system that does not share the same catalog. Unless, of course, you regularly run a routine that checks for uncatalogued datasets.

Beyond that, as shown by the example at the beginning of the article, 'the impossible' did happen and the dataset ended up still catalogued after it had been deleted. If your response to all this is "I thought System-Managed Storage (SMS) was supposed to solve all this uncatalogued dataset stuff", I should point out that this site is running z/OS 1.2 and has been running SMS for about 15 years. But the troublesome IODF volume is not SMS-managed.

It can be a real eye opener to see how many critical DASD volumes are not SMS-managed in a typical SMS environment. To find out for yourself, try using ISMF option 2.1 Volume - DASD. On the Volume Selection Entry Panel, specify:

- Select Source to Generate Volume List – 2 for New List
- Specify Source of the New List – 1 for Physical
- Type of Volume List – 1 for Online
- Volume Serial Number – * for all
- Acquire Space Data – Y for Yes.

It may take a while to display, but when the Volume List appears, scroll right (PF11) until you see the PHYSICAL STATUS field, usually column 22. A value of NONSMS is displayed for all DASD volumes that are not managed by SMS. CONVERT means the volume is SMS-managed.

A TYPICAL SCENARIO

The truth is: a volume shared between systems that do not share the same master catalog is a magnet for all sorts of problems, including 'the impossible' mentioned above. Although FTP is a viable alternative, it is certainly easier to use a shared volume to transfer a dataset between the two systems. From one system, copy the dataset to the shared volume, and copy it off the shared volume to another volume on the other system. As a shortcut, you can easily see someone cataloguing the dataset on the second system and accessing it directly, without copying it to another volume. Delete the dataset later from the first system and you have 'the impossible' situation described at the beginning of the article.

By the way, if you do plan to use this approach to transfer datasets between systems, you should be aware that it does not work for all dataset types, most notably VSAM and HFS, unless you go to the effort of building the correct catalog entries on the second system. A far easier approach is to use DFSMSdss DUMP to create a sequential dataset, and RESTORE from it on the second system to recreate the original dataset, including the catalog entries. An additional advantage of DFSMSdss is that you can store multiple datasets in a single dss DUMP sequential dataset.

CONFLICTING INFORMATION

One final note. When investigating this type of problem, you may see multiple entries for a single dataset in ISPF option 3.4. In general, multiple entries in 3.4 do not necessarily indicate a catalog problem. IDCAMS LISTCAT is the only reliable source of catalog information, other than dumping and interpreting the master and user catalogs. Given that IBM stopped publishing the internal format of the catalog in the late ’80s, a catalog dump is no longer a viable alternative.

Jon E Pearkins
Adiant (Canada)

Listing online volumes

The following program is used to create a listing (or dataset) containing all the currently-online volumes on a system. The listing contains address, devicetype, status (allocated or just online), number of cylinders, mount attribute, whether the volume is marked as sharable or not, and optional information such as controller name/type that is held in a table in the program. This information can then be used to remove unwanted volumes (see rexx ‘mvoffln’ at the end), reconcile volumes and addresses against control files, create d/r volume lists, and many other DASD management tasks.

It currently runs on an OS/390 2.10 system.

ONLNDASD PROGRAM

```
***********************************************************************
* HOUSEKEEPING...                                                     *
```

```
***********************************************************************
ONLNDASD CSECT
ONLNDASD AMODE 31
ONLNDASD RMODE 24
         BAKR  R14,Ø                   SAVE CALLER DATA ON STACK
         LR    R12,R15                 GET ENTRY POINT
         LA    R11,2Ø48(R12)           LOAD SECOND BASE
         LA    R11,2Ø48(R11)           LOAD SECOND BASE
         USING ONLNDASD,R12,R11        ADDRESSABILITY
         L     R2,Ø(R1)                GET ADDR OF PARM
         OPEN  (REPORT,(OUTPUT))
***********************************************************************
* SCAN THROUGH THE UCB'S, LOOKING FOR THE TYPE WE WANT...             *
***********************************************************************
SCANUCBS DS    ØH
         USING UCBOB,R4                ADDRESSABILITY TO UCB
         LA    R4,UCBAREA              +POINT TO UCB STORAGE AREA
         LA    R3,MSG3+1               POINT TO FIRST MSG FIELD
         XC    UCBWORK,UCBWORK         +INITIALIZE UCBSCAN WORKAREA
UCBLOOP  DS    ØH
*
         UCBSCAN COPY,                                                 X
               WORKAREA=UCBWORK,                                       X
               UCBAREA=UCBAREA,                                        X
               DCEAREA=NONE,                                           X
               DCELEN=Ø,                                               X
               VOLSER=NONE,            DON'T SELECT BY VOLSER          X
               DEVN=Ø,                 START WITH FIRST UCB            X
               DYNAMIC=YES,            INCLUDE DYNAMICALLY ADDED UCBS X
               RANGE=ALL,              4 AND 3-DIGIT UCBS              X
               NONBASE=NO,             NOT SURE WHAT THIS DOES         X
               DEVCLASS=DASD,          SELECT DASD UCBS ONLY           X
               DEVCID=Ø,               DON'T SELECT BY DEVICE CHAR.    X
               IOCTOKEN=NONE,          NO IODEVICE TABLE TOKEN         X
               LINKAGE=SYSTEM,         USE PC CALL                     X
               PLISTVER=MAX
*
         LTR   R15,R15                 GOT UCB OK?
         BZ    UCBCHECK                YES..CHECK IT
         C     R15,=F'4'               END OF UCBS?
         BE    ENDUCBS                 YES..CLEAN UP, ETC
         B     BADCALL                 NO...SHOW RETURN/REASON CODES
UCBCHECK DS    ØH
         TM    UCBSTAT,UCBONLI         IS THIS ONE ONLINE?
         BO    GETINFO                 YES..
         B     UCBLOOP                 NO...IGNORE IT
***********************************************************************
* NB AS WE GET THE FORMAT4 FROM THE VTOC THERE IS SOMETIMES AN        *
* EXTRA (CE?) CYLINDER ADDED ON- WE WILL TAKE THIS OFF IF WE CAN      *
* RECOGNIZE THAT IT IS THERE (EG 886 ON A 338Ø 'D').                  *
```

```
***********************************************************************
GETINFO  DS    ØH
         MVC   27(8,R3),=C'(?????) '    SET UP DEFAULT SIZE
*
         LSPACE UCB=(R4),               GET THE FORMAT4 DSCB...        X
               F4DSCB=F4DSCB,                                          X
               MSG=F4ERRMSG             PLACE POSSIBLE ERRMSG IN HERE
*
         LTR   R15,R15                  LSPACE WORKED OK?
         BZ    CHKCYLS                  YES..
         ST    R15,FWORD                NO...SAVE RETURN CODE
         TM    SWITCH,MSGSENT           ALREADY DISPLAYED ERROR TEXT?
         BO    SHOWRC                   YES..
         OI    SWITCH,MSGSENT           NO...SET SO WE DON'T REPEAT IT
         PUT   REPORT,F4ERRMSG          DISPLAY ERROR TEXT
SHOWRC   DS    ØH
         UNPK  DWORD(3),FWORD+3(2)      UNPACK RETURN CODE + 1 BYTE
         TR    DWORD(2),HEXTAB-24Ø      XLATE TO PRINTABLE HEX
         MVC   26(5,R3),=C'RC=X"'       SET UP CONSTANT
         MVC   31(2,R3),DWORD           MOVE IN RC
         MVI   33(R3),C'"'
         B     CARRYON                  IGNORE REST OF THIS BIT
CHKCYLS  DS    ØH
         MVC   27(7,R3),=X'4Ø2Ø2Ø6B2Ø212Ø'  YES..MOVE IN EDIT PATTERN
         LH    R1,F4DSCB+18             GET NUMBER OF CYLINDERS
         CH    R1,=H'886'               886 CYLS (338Ø 'D')?
         BE    TAKE1OFF                 YES..
         CH    R1,=H'1771'              1771 CYLS (338Ø 'E')?
         BE    TAKE1OFF                 YES..
         CH    R1,=H'2656'              2656 CYLS (338Ø 'K')?
         BE    TAKE1OFF                 YES..
         CH    R1,=H'1114'              1114 CYLS (339Ø M1)?
         BE    TAKE1OFF                 YES..
         CH    R1,=H'2227'              2227 CYLS (339Ø M2)?
         BE    TAKE1OFF                 YES..
         CH    R1,=H'334Ø'              334Ø CYLS (339Ø M3)?
         BE    TAKE1OFF                 YES..
         CH    R1,=H'1ØØ18'             1ØØ18 CYLS (339Ø M9)?
         BE    TAKE1OFF                 YES..
         B     GETCYLS                  NO...
TAKE1OFF DS    ØH
         BCTR  R1,Ø                     THERE IT GOES...
GETCYLS  DS    ØH
         CVD   R1,DWORD                 CONVERT TO DECIMAL
         ED    27(7,R3),DWORD+5         EDIT IN NUMBER OF CYLINDERS
         MVI   27(R3),C'('              MAKE IT LOOK PRETTY
         MVI   34(R3),C')'
CARRYON  DS    ØH
         TM    UCBFL1,UCBBOX            BOXED?
         BO    ITSBOXED                 YES..
         TM    UCBSTAT,UCBALOC           ALLOCATED?
         BO    ITSALLOC                  YES..
         TM    UCBSTAT,UCBONLI           ONLINE?
         BO    ITSONLIN                  YES..
         B     ITSOFLIN                  NO...LET'S CALL IT OFFLINE
ITSONLIN DS    ØH
         MVC   1Ø(7,R3),ONLINE           SET UP MSG
         B     GETVOLID                  GO AND GET VOLID
ITSOFLIN DS    ØH
         MVC   1Ø(7,R3),OFFLINE          SET UP MSG
         B     GETVOLID                  GO AND GET VOLID
ITSBOXED DS    ØH
         MVC   1Ø(7,R3),BOXED            SET UP MSG
         B     GETDEVTP                  GO AND GET DEVICE TYPE
ITSALLOC DS    ØH
         MVC   1Ø(7,R3),ALLOC            SET UP MSG
         B     GETVOLID                  GO AND GET THE VOLID
GETVOLID DS    ØH
         MVC   18(6,R3),UCBVOLI          MOVE VOLID TO MSG LINE
GETDEVTP DS    ØH
         MVC   5(4,R3),QUERIES           SET UP UNKNOWN DEVTYPE
         CLI   UCBTBYT4,X'ØE'            338Ø? DASD
         BE    SET338Ø                   YES..
         CLI   UCBTBYT4,X'ØF'            339Ø? DASD
         BE    SET339Ø                   YES..
         B     CHKUCBS                   NO...LEAVE AS '????'
SET338Ø  DS    ØH
         MVC   5(4,R3),=C'338Ø'
         B     CHKUCBS                   YES..
SET339Ø  DS    ØH
         MVC   5(4,R3),=C'339Ø'
CHKUCBS  DS    ØH                        YES..
         MVC   52(13,R3),NONSHARE        DEFAULT TO NON-SHAREABLE
         TM    UCBTBYT2,UCBRR            SHAREABLE?
         BNO   CHECKPRI                  NO...
         MVC   52(13,R3),SHARE           YES..SET TO SHAREABLE
CHECKPRI DS    ØH
         TM    UCBSTAB,UCBBPRV           PRIVATE?
         BNO   CHECKPUB                  NO...
         MVC   37(7,R3),PRIVATE          YES..SHOW THAT IN MSG
         BAL   R9,LOCUCB                 SEE WHERE UCB IS...
         B     CHECKCTL
CHECKPUB DS    ØH
         TM    UCBSTAB,UCBBPUB           PUBLIC?
         BNO   CHECKSTR                  NO...
         MVC   37(7,R3),PUBLIC           YES..SHOW THAT IN MSG
         BAL   R9,LOCUCB                 SEE WHERE UCB IS...
         B     CHECKCTL
CHECKSTR DS    ØH
         TM    UCBSTAB,UCBBSTR           STORAGE?
         BNO   CHECKCTL
         MVC   37(7,R3),STORAGE         YES..SHOW THAT IN MSG
         BAL   R9,LOCUCB                SEE WHERE UCB IS...
CHECKCTL DS    ØH
         XC    FWORD2,FWORD2            CLEAR WORK REG
         MVC   FWORD2+2(2),UCBCHAN      DEVICE ADDRESS TO LOOK FOR
         MVC   CTLUNIT,CTLNFND          SET DEFAULT CTLUNIT
         BAL   R9,FINDCTL               GO AND FIND CTLUNIT
         MVC   67(8,R3),CTLUNIT         SET CTLUNIT
         UNPK  UNPKFLD(5),UCBCHAN(3)    UNPACK HEX CUU + 1 CHAR
         TR    UNPKFLD(4),TRTAB2-24Ø    MAKE PRINTABLE HEX
         CLI   UNPKFLD,C'Ø'             LEADING ZERO?
         BNE   DISPLAY2                 NO...
         MVI   UNPKFLD,C' '             YES..BLANK OUT
DISPLAY2 DS    ØH
         MVC   Ø(4,R3),UNPKFLD          MOVE CUU TO MSG LINE
         PUT   REPORT,MSG3              DISPLAY INFO
         MVI   MSG3,C' '                YES..CLEAR OUT LINE
         MVC   MSG3+1(MSG3L-1),MSG3
         LA    R3,MSG3+1
         B     UCBLOOP                  AND GET NEXT UCB
ENDUCBS  DS    ØH
***********************************************************************
* RETURN TO CALLER WITH RELEVANT RETURN CODE...                      *
***********************************************************************
RETURN   DS    ØH
         L     R15,RETC                 LOAD RETURN CODE
         PR    ,                        RESTORE CALLER DATA, RETURN
***********************************************************************
* BAD RETURN CODE FROM CALL TO 'UCBSCAN'...                          *
***********************************************************************
BADCALL  DS    ØH
         ST    R15,RETCD                SAVE RETURN CODE FROM UCBSCAN
         ST    RØ,REASN                 SAVE REASON CODE FROM UCBSCAN
         UNPK  UNPKFLD(3),RETCD+3(2)    UNPK RETURN CODE + 1 BYTE
         TR    UNPKFLD(2),TRTAB2-24Ø    XLATE TO PRINTABLE HEX
         MVC   MSGBTXT1,UNPKFLD         MOVE RETURN CODE TO MSG AREA
         UNPK  UNPKFLD(3),REASN+3(2)    UNPK REASON CODE + 1 BYTE
         TR    UNPKFLD(2),TRTAB2-24Ø    XLATE TO PRINTABLE HEX
         MVC   MSGBTXT2,UNPKFLD         MOVE REASON CODE TO MSG AREA
         PUT   REPORT,MSGB              SHOW CODES...
         MVC   RETC,=F'8'               SET RC=8
         B     RETURN
***********************************************************************
*                    + + S U B R O U T I N E + + +                   *
* CONVERT CHARACTER CUU (EG 'Ø94F') INTO ITS BINARY EQUIVALENT. THIS  *
* IS SO THAT VALID RANGE COMPARISONS CAN BE MADE IF A RANGE OF CUUS   *
* HAS BEEN REQUESTED. THE ROUTINE USES 4-DIGIT ADDRESSES, PADDED WITH *
* A LEADING 'Ø', IF REQUIRED.                                         *
***********************************************************************
```

```
CONVCUU  DS    ØH
         TR    FWORD(4),TRTAB          CONV. C'A->F' INTO X'A->F'
         XC    DWORD,DWORD             CLEAR OUT WORKAREA
         PACK  DWORD+4(4),FWORD(5)     REMOVE ZONES
         L     R8,DWORD+4              LOAD 'ØØCCUUØØ'
         SRL   R8,8                    SHIFT OUT TRAILING 'ØØ'
         ST    R8,FWORD                SAVE BINARY CUU VALUE
         BR    R9                      RETURN FROM SUBROUTINE
***********************************************************************
*                    + + S U B R O U T I N E + + +                    *
* SEE IF UCB IS ABOVE ('A') OR BELOW ('B') THE 16MEG LINE. NOTE THAT  *
* THERE IS ONLY AN EXTENSION FOR UCBS IF THEY ARE 'BELOW THE LINE'.   *
***********************************************************************
LOCUCB   DS    ØH
         MVI   48(R3),C'?'             DEFAULT IS "DON'T KNOW"
         MODESET MF=(E,SUPMODE)        ENTER SUPERVISOR MODE
*
         UCBLOOK DEVN=UCBCHAN,         LOOK BY DEVICE ADDRESS         X
               UCBPTR=FWORD,           TO HOLD A(UCB COMMON SEGMENT)  X
               DYNAMIC=YES,            INCLUDE DYNAMIC UCBS           X
               RANGE=ALL,              3 AND 4 DIGIT UCBS             X
               NOPIN,                  DON'T PIN UCB                  X
               LOC=ANY                 ABOVE AND BELOW THE LINE
*
         LTR   R15,R15                 SUCCESSFUL?
         BNZ   RESET                   NO...LEAVE AS DEFAULT
         MODESET MF=(E,PROBMODE)       RETURN TO PROBLEM MODE
         MVI   48(R3),C'A'             DEFAULT IS 'A'BOVE
         L     R1,FWORD                GET UCB ADDRESS
         C     R1,=F'16777216'         ABOVE 16M?
         BHR   R9                      YES..RETURN
         MVI   48(R3),C'B'             NO...MAKE IT 'B'ELOW
         BR    R9                      RETURN FROM ROUTINE
RESET    DS    ØH
         MODESET MF=(E,PROBMODE)       RETURN TO PROBLEM MODE
         BR    R9                      RETURN FROM ROUTINE
***********************************************************************
*                    + + S U B R O U T I N E + + +                    *
* FIND WHICH CTLUNIT THE ADDRESS IS ON: HDS, SVA, ESS, ETC...         *
***********************************************************************
FINDCTL  DS    ØH
         LA    R1,CTLRTAB              LOCATE CTLUNIT TABLE
         LA    R2,CTLENTS              NUMBER OF ENTRIES
         L     R1Ø,FWORD2              GET BINARY CUU VALUE
FINDLOOP DS    ØH
         L     R6,Ø(R1)                GET LOW RANGE ADDRESS
         CR    R1Ø,R6                  CUU EQUAL?
         BL    FINDBUMP                LOW - NOT IN RANGE, TRY NEXT
         L     R6,4(R1)                GET HIGH RANGE ADDRESS
         CR    R1Ø,R6                  CUU EQUAL?
         BH    FINDBUMP                  HIGH - NOT IN RANGE, TRY NEXT
         MVC   CTLUNIT,8(R1)             IN RANGE - SAVE CTLUNIT NAME
         BR    R9                        RETURN FROM ROUTINE
FINDBUMP DS    ØH
         LA    R1,16(R1)                 BUMP TO NEXT ENTRY
         BCT   R2,FINDLOOP               KEEP LOOKING
         BR    R9                        RETURN FROM ROUTINE
         EJECT
*---------------------------------------------------------------------*
*
         LTORG                           LITERAL POOL
*
OFFLINE  DC    CL7'OFFLINE'
ONLINE   DC    CL7'ONLINE '
BOXED    DC    CL7'BOXED  '
ALLOC    DC    CL7'ALLOC  '
PRIVATE  DC    CL7'PRIVATE'
PUBLIC   DC    CL7'PUBLIC '
STORAGE  DC    CL7'STORAGE'
SHARE    DC    CL13'SHAREABLE'
NONSHARE DC    CL13'NON-SHAREABLE'
HEXTAB   DC    C'Ø123456789ABCDEF'
LETTERS  DC    X'ØAØBØCØDØEØF'
NUMBERS  DC    X'ØØØ1Ø2Ø3Ø4Ø5Ø6Ø7Ø8Ø9'
FWORD    DS    F
FWORD2   DS    F
DWORD    DS    D
F4DSCB   DS    CL96
F4ERRMSG DS    CL3Ø
QUERIES  DC    CL4'????'
CUU      DS    CL4
VOLID    DS    CL6
SWITCH   DC    X'ØØ'                     SWITCH FIELD
FOUND1   EQU   X'Ø1'
GENERIC  EQU   X'Ø2'
RANGE    EQU   X'Ø4'
GETDASD  EQU   X'Ø8'
MSGSENT  EQU   X'Ø4'
ALLUCBS  EQU   X'1Ø'
ONECUU   EQU   X'2Ø'
ONEVOLID EQU   X'4Ø'
ONLIN    EQU   X'8Ø'
RETC     DS    F
UCBAREA  DS    XL48                      HOLDS UCB COMMON & DEV SEGS
UCBWORK  DS    XL1ØØ                     UCBSCAN WORKAREA
UNPKFLD  DS    CL5
RETCD    DS    F
REASN    DS    F
SUPMODE  MODESET KEY=ZERO,MODE=SUP,MF=L
PROBMODE MODESET KEY=NZERO,MODE=PROB,MF=L
```

```
*
*                Ø 1 2 3 4 5 6 7 8 9 A B C D E F
TRTAB    DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' Ø
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 1
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 2
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 3
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 4
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 5
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 6
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 7
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 8
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 9
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' A
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' B
         DC    X'FFØØØØØØØØØØØØFFFFFFFFFFFFFFFFFF' C  (ABCDEF)
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' D
         DC    X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' E
         DC    X'ØØØØØØØØØØØØØØØØØØØØFFFFFFFFFFFF' F  (Ø123456789)
*
* THIS TABLE HOLDS THE RANGE OF ADDRESSES FOR EACH CONTROL UNIT (IN
* ITS DECIMAL EQUIVALENT).
*
TRTAB2   DC    CL16'Ø123456789ABCDEF'
*
CTLRTAB  DS    ØF
         DC    F'Ø1Ø24',F'Ø1279',C' H.D.S. '  Ø4ØØ-Ø4FF HDS
         DC    F'Ø1792',F'Ø2Ø47',C'  RVA1  '  Ø7ØØ-Ø7FF RVA1
         DC    F'Ø4Ø96',F'Ø8191',C'  SVA1  '  1ØØØ-13FF SVA1
         DC    F'Ø8192',F'12287',C'  SVA2  '  2ØØØ-23FF SVA2
         DC    F'12288',F'13311',C'  SVA3  '  3ØØØ-33FF SVA3
         DC    F'16384',F'174Ø7',C'  SVA4  '  4ØØØ-43FF SVA4
CTLENTS  EQU   (*-CTLRTAB)/16                  NUMBER OF TABLE ENTRIES
CTLNFND  DC    CL8'????????'
CTLUNIT  DC    CL8' '
*
MSG3     DC    CL8Ø' '
MSG3L    EQU   *-MSG3
*
MSGB     DC    C'>>> ERROR IN CALL TO "UCBSCAN"...RC=X''..'', RS=X''..'X
               '.'
MSGBTXT1 EQU   MSGB+38,2
MSGBTXT2 EQU   MSGB+48,2
MSGBL    EQU   *-MSGB
*-------------------------------------------------------------------*
* REPORT DCB...                                                     *
*-------------------------------------------------------------------*
REPORT   DCB   DDNAME=REPORT,DSORG=PS,LRECL=8Ø,MACRF=PM,BLKSIZE=8Ø
*-------------------------------------------------------------------*
* REGISTERS EQUATES, ETC...                                         *
*-------------------------------------------------------------------*
```

```
         YREGS
         PRINT ON,GEN
UCBDEF   DSECT
         IEFUCBOB
         PRINT NOGEN
         CVT   DSECT=YES
*
         END                     , END OF PROGRAM
```

MVOFFLN REXX

```
/* REXX */
/* --------------------------------------------------------------- */
/* MVOFFLN : READ A LIST OF VOLIDS AND ADDRESSES (CREATED BY THE    */
/*           'ONLNDASD' PROGRAM) AND ISSUE 'VARY OFF' COMMANDS      */
/*           AGAINST THOSE THAT ARE 'SPARES' (VOLID BEGINS 'MV').  */
/*           THE ROUTINE CALLS 2 OTHER PROGRAMS:                    */
/*           1 MVSCMD    ISSUE A COMMAND PASSED AS A PARM           */
/*           2 WAIT      WAIT FOR SECONDS PASSED AS A PARM          */
/* --------------------------------------------------------------- */
PARSE UPPER ARG SYSID .

"EXECIO * DISKR DISKS (STEM DISKS. FINIS"
OFFS = Ø

DO AA = 1 TO DISKS.Ø
  PARSE VALUE DISKS.AA WITH ADD1 . . VOL1 .
  IF LEFT(VOL1,2) = "MV" THEN DO
    CMD = "V "ADD1",OFFLINE"
    "CALL 'SYSG.LINKLIB(MVSCMD)'" "'"CMD"'"   /* ISSUE VARY OFF */
    "CALL 'SYSG.LINKLIB(WAIT)'" "'1'"         /* WAIT 1 SECOND  */
    OFFS = OFFS + 1
  END
END
SAY "OFFLINE COMMANDS ISSUED = "OFFS
```

MVOFFLN REXX

```
//JOBCARD
//*
//* ISSUE 'VARY OFFLINE' COMMANDS FOR ANY ONLINE MV- DISKS...
//*
//STEPØ1  EXEC PGM=ONLNDASD
//SYSPRINT DD SYSOUT=*
//REPORT   DD DSN=&&ONDASD,DISP=(MOD,PASS),RECFM=F,
//         SPACE=(CYL,2),UNIT=SYSDA,LRECL=8Ø
//STEPØ2  EXEC PGM=IKJEFTØ1,DYNAMNBR=5Ø,PARM='%MVOFFLN'
//SYSPRINT DD SYSOUT=*
```

```
//SYSTSPRT DD SYSOUT=*
//SYSEXEC  DD DISP=SHR,DSN=SYS3.GEN.ELIB   <== CONTAINS MVOFFLN
//DISKS    DD DSN=&&ONDASD,DISP=(OLD,PASS)
//SYSTSIN  DD DUMMY
//*
```

Grant Carson
Systems Programmer (UK)

Using CSI to identify VSAM datasets defined with IMBED, REPLICATE, and KEYRANGE – revisited

In the August *MVS Update*, issue 203, the author of the article *Using CSI to identify VSAM datasets defined with IMBED, REPLICATE, and KEYRANGE* stated that the above mentioned attributes will not be supported in z/OS 1.4. This statement can lead to false conclusions. The attributes will be ignored if present; changes to the datasets are not required, but are recommended.

The *z/OS V1R3.0-V1R4.0 DFSMS Migration* manual (GC26-7398-03), Section 2.4.7 *IDCAMS changes for DEFINE CLUSTER with IMBED or REPLICATE*, which is at http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/dgt2mn12/2.4.7, states:

With advances in DASD and control unit hardware, particularly with cache control units, the IMBED and REPLICATE options of the IDCAMS DEFINE CLUSTER command no longer offer performance advantages. Support for these IDCAMS options is now limited to existing data sets only.

If you define a new VSAM key-sequenced data set with IMBED or REPLICATE, IDCAMS ignores these options. If you have data sets that are already cataloged with these options, processing is not affected.

William K Mongan (Germany)

Possible rounding errors in COBOL on the mainframe – update

I recently reported on rounding errors we have been experiencing on the mainframe (*Possible rounding errors in COBOL on the mainframe, MVS Update*, issue 203, August 2003). At the time I submitted the problem to IBM as a PMR (Problem Management Report) and have now received information from IBM regarding the problem.

To recap, the problem was a rounding error with COMPUTE ROUNDED.

Problem:

```
Ø1 A  PIC S9(1Ø)V9(8)
Ø1 B  PIC S9(1Ø)V9(8)
Ø1 C  PIC S99V9
Ø1 D  PIC S99

MOVE 9.975 TO A
MOVE 6 TO C
MOVE 6 TO D
```

Case 1:

```
COMPUTE B ROUNDED = A / (1Ø ** C)
```

Case 2 :

```
COMPUTE B ROUNDED = A / (1Ø ** D)
```

Result:

```
case 1 +Ø.ØØØØØ997
case 2 +Ø.ØØØØØ998
```

When the exponential factor contains a variable with a defined decimal point, the rounding error occurs.

Here’s what IBM had to say (excerpts from the PMR):

I reviewed the results with a COBOL Runtime developer.

If the exponent contains decimal places, then floating point routines are used.

Action Taken: I found that I get consistent results if I break the COMPUTE into two parts:

```
 COMPUTE INT = (10 ** D)
 COMPUTE B ROUNDED = A / INT.
```

where 01 INT PIC S9(10)V9(8).

The answer is .00000998 signed in all cases.

Appendix A of the COBOL/OS390 Programming Guide *under* Fixed-point Data and Intermediate Results *(just under the box table, the title varies with the manual release) states: "Exponentiation is represented by the expression op1 ** op2." and the first bullet a few lines later says: "When op2 is expressed with decimals, floating point instructions are used".*

Please see very old APARs PP74213 for FCOBOL under VSE and PP43923 for OS/COBOL 2.3 on MVS. These explain that low-order precision may vary. The most recent explanation is in APAR PQ55320 for the LE/OS390 C Library, but the same principles apply.

The response from COBOL Development said:

We looked at this PMR in some detail today. Here is what we found:

The difference is that the calculations are being done in floating-point arithmetic instead of fixed-point arithmetic and the problem is that not all fixed-point decimal numbers with a fractional part can be exactly represented in a floating-point number.

*When using COMPUTE B ROUNDED = A / (10 ** C) with A = 9.975 and C = 6 and C is defined to contain decimal places, the calculation is being done in floating point and the result cannot be exactly represented as a floating point value (any decimal number with a fractional part of the number which is represented in base 10 may not be exactly representable as a floating point value which is represented in base 16).*

According to our COBOL Programming Guide, Appendix A, Intermediate Results and Arithmetic Precision, *floating point* *instructions are used to compute the result if an exponent contains decimal places.*

In this case, the value returned by COBOL/OS390 is 3CA75A4C7CB054C7 (in hex floating point) which is 0.00000997499999999999999 ... (in decimal notation). In this case, COBOL for OS/390 gives an answer that is slightly lower than the exact answer. This would explain why the result is only 0.00000997 instead of 0.00000998 when rounded since 0.00000997499999999999999 … would be rounded to 0.00000997.

As with any decimal number that has a fractional part, we cannot always exactly represent the number in floating point notation. Hence, we give the closest answer to the exact answer in this case. In some cases, the answer will be slightly above the exact answer and in some cases it will be slightly below the exact answer. In this situation, when the closest answer to the exact answer is slightly below the exact answer, the rounding may not give the desired result.

So, as it is, it looks like the customer has two options:

1 *Either use exponents that do not contain decimals (so that the calculation is done in fixed-point arithmetic).*

2 *Calculate the exponentiation in a separate COMPUTE statement (so that the division will be done in fixed-point).*

CONCLUSION

Works as designed!

This is not really the solution we were looking for but we can understand how it has come about. Our next step is to request IBM to update the COBOL compiler to at least report this type of calculation as a ‘Compiler Warning Message’.

Rolf Parker
Systems Programmer (Germany)

System-wide member search utility

INTRODUCTION

While working under TSO, we often wish there was an easier way to locate a specific member or all members with similar names among ever-increasing partitioned datasets. Sometimes we have difficulty in remembering the location of members, even though they are in our own libraries! Or maybe what we need is just any IEBCOPY job and we use unnecessary time in navigating libraries to look for any members whose name starts with IEBC. All these examples lead us to think that searching members is a time-consuming and laborious task.

I came up with a utility for these kinds of situation. The utility not only gives us the facility to find members easily, but also permits us to browse, edit, print, copy, or even delete those members matching our selection criterion entered on the Query input panel. In addition, if there are several copies of the same member distributed among several libraries, we can easily figure out which one is of interest to us, since navigating between those libraries is an easy task (it is just question of issuing several browse commands one after another on the Query Output panel).

THE ALGORITHM USED IN THE ARTICLE

The principal part of the article is to build an inventory dataset. The job JCL1 is in charge of this task, which is supposed to be executed as frequently as possible to keep the inventory up to date. By the way, it's the first of two jobs the utility comes with. The second job, JCL2, is used to search for members in batch.

In our data centre we schedule this job on a daily basis early in the morning. JCL1 does some housekeeping jobs for the utility, executes a DCOLLECT, and then calls the REXX EXEC REXX1 to build the inventory VSAM dataset by handling the DCOLLECT output, which is executed against the selected disk volumes.

After building the inventory, the rest makes queries against it under ISPF. This interactive part of the utility is realized by the REXX program REXX2.

This REXX EXEC presents the user with a Query Input Panel, which is PANELIN, asking them to enter the member they want to search for, then scans the inventory, fetches all related records, and lastly presents all found members and their locations on the Query Output panel, which is PANELOUT.

From this point on, the user is given the opportunity to browse, edit, print, and delete any member found in the inventory as a result of the query. If a user wants, they can copy the member to another library or get more information on any specific member in a real-time manner. In this latter case, an LMMLIST command is issued against the member, and its real-time statistics are fetched and presented to the user on a different panel.

VSAM INVENTORY DATASET

The record layout of the inventory dataset

The inventory dataset is the backbone part of the utility. Except for two kinds of special record, the inventory contains one standard record type, whose layout is given below (column range then field):

- 1-8 – member name
- 10-53 – dataset name
- 55-60 – disk volser
- 62-69 – creation date of member
- 71-78 – last change date of member
- 80-86 – user who changed the member last.

When a query is made, all those members meeting the user’s selection criterion are presented on the Query Output panel with all of the above fields fetched from the inventory. However, if a user wants to get more information about one of these members, an LMMLIST command is issued in real-time and the result is displayed for the user.

Actually, the utility first constructs the sequential inventory dataset during the execution of the job JCL1, but it's just a temporary dataset. The main inventory on which the utility relies is the VSAM inventory dataset. It's a KSDS.

Key composition for the inventory

Because it's a KSDS, we have to come up with a key for it. For designing a proper key for it, three fields are taken into account – member, dataset where the member is, and volser where the dataset is. All three fields are concatenated and one unique key is formed.

Normally, if we're talking about a data centre that is totally SMS-managed, we wouldn't need the volser component in building the key because all datasets are catalogued datasets and there won't be two datasets with the same name. In this case the volume component can be eliminated from the key since dataset name+member will make a unique key for the inventory. This calls for fewer characters in building the key, and consequently requires less VSAM space. So it's good for space usage. However, to extend the capabilities of the utility over uncatalogued datasets as well, member+dataset+volser key structure will be the optimum solution.

To explain the key structure, let's think about the member MEM1. If we have three catalogued datasets, DS1, DS2, and DS3, and each of them has a member with the name MEM1, then a key of member+dataset would be sufficient since each MEM1DS1, MEM1DS2, or MEM1DS3 combination is a unique value. On the other hand, if we have an identical copy of the DS3 on another volume, and if it's uncatalogued, this approach building the key will not work because there will be two MEM1DS3 records, which prevents us from building the VSAM dataset properly. Adding the third field, which is volser, will overcome this obstacle. In the example, let's imagine that DS1, DS2 and catalogued version of DS3 reside in the volume VL1, and the uncatalogued version of DS3 resides in VL2. So all these records will be unique records within the inventory dataset:

```
 MEM1DS1VL1, MEM1DS2VL1, MEM1DS3VL1 and MEM1DS3VL2,
```

SPECIAL RECORDS OF THE INVENTORY

The inventory dataset is a VSAM KSDS, which will be used for the READ operations by the utility. That's why the FREESPACE(0 0) is coded in its cluster definition. As stated earlier, the inventory hosts two special record types for internal use. The first kind of record is the one that starts with the $ (seven blank characters plus one dollar sign). It is called 'Inventory descriptive record'. This special record gives some basic characteristics of the inventory, such as number of datasets, number of members, and number of disk volumes that that inventory is composed of. In addition, build date of the inventory is also found here. There's just one record of this type in the inventory. For example, the following special record can be interpreted like this: the inventory has collected all 65,196 member names of 678 partitioned datasets, which are spread out across 12 disk volumes:

```
     $ DSET NUM = 678 - MEMBER NUM = 65196 - BUILD DATE =
     16/Ø6/Ø3  - VOL.NUM = 12
```

The other special record is called 'volume record', which starts with the $ (six blank characters plus one dollar sign plus one more blank character). Each inventory has at least one record of this type. The wider the scope of the utility, the more of this kind of record there will be. For example, the inventory given above would have a volume record something like the following:

```
"      $  DWR3Ø1DWR3Ø2DWR3Ø3DWR3Ø4DWR3Ø5DWR3Ø6DWR3Ø7DLB3Ø1DLB3Ø2
DLB3Ø3DLB3Ø4DLB3Ø5"
```

There are exactly 12 volume serial numbers concatenated together.

Note that the first type of record is interpreted and then presented on both Query panels (Query Input and Query Output). However, the second type of record is retrieved on demand basis on the Query Output panel.

ON INVENTORY DISK USAGE AND INVENTORY BUILD-UP ELAPSED TIME

Just to give you an idea of the inventory build-up elapsed time and possible inventory size, here are some values taken from our data centre where this utility was originally implemented.

In our production LPAR, we have around 2,720 partitioned datasets, which all house more than 450,000 members. To accommodate all these member entries and their associated information, our inventory dataset takes up 140 cylinders of disk space.

Note that, in the early steps of the job JCL1, one sequential inventory dataset is created, which is temporary. This sequential dataset uses just 45 cylinders and is deleted at the end of the job.

As for the duration time for building the inventory dataset, it's about 15 minutes.

HOW TO MAKE THE UTILITY RUN

Put all these members into a library – let's say that it's called EXP.MEMSRCH.CNTL. Then make sure to update the SYSPROC statement in the JCL1 to point out this library, eg:

```
//SYSPROC  DD   DISP=SHR,DSN=EXP.MEMSRCH.CNTL
```

With a job scheduler utility, execute the job JCL1 on a daily basis to build up the INVENTORY.

To run the utility from TSO, call the REXX EXEC REXX2, which is the foreground part of the utility:

```
TSO ex 'EXP.MEMSRCH.CNTL(REXX2)'
```

Or better yet, put the REXX2 into any SYSPROC library and then the utility can be accessed by an option in the ISPF application menu:

```
...

%          I  +CTT/I       - Control T Reports
%          H  +HSM         - HSM Management
%          M  +MemSRCH     - Member Search utility
. . .

)PROC
  &ZSEL = TRANS( &ZCMD

                I,'CMD(%mainctt)'
                H,'CMD(%mainhsm)'
                M,'CMD(%Rexx2)'
              ' ',' '
                X,'EXIT'
```

UTILITY COMPONENTS

The utility is composed of two REXX programs, two JCL, and nine ISPF panels. Their names are given below:

- JCL – Jcl1, Jcl2.
- REXX EXECs – Rexx1, Rexx2, Rexx3.
- ISPF panels – Panelin, Panelout, Panel1, Panel2, Panel3, Panel4, Panel5, Panel6, Panel7.

SAMPLE QUERY INPUT (PANELIN) AND QUERY OUTPUT PANEL (PANELOUT)

Here, a sub-string search is being realized. What we want is to scan the inventory and fetch all those members containing the letters ‘JCL’ in their names:

```
------------------------ MEMBER SEARCH UTILITY: D31Ø ------------------


 CHOOSE SEARCH TYPE ===> T  ENTER MEMBER NAME / SUB-STRING / MASK-FIELD)

 (G) Generic      ------
 (S) Specific          | ==> JCL
 (T) Sub-string   ------


 (M) Mask                ==> -M-A-S-K
 For SUB-STRING search, enter any string of 1 to 8 char.long. It is used
 to search all member names that have any character string IN COMMON.

 Example: To search all members where the string “JCL” is found in the
 member name, enter the word “JCL” into the Member field.

 Inventory is built on 17/06/03 with 23173 members of 378 datasets,
 which are spread out to 7 disk volumes.
```

The utility may return an output table something like this:

```
------------------ MEMBER SEARCH UTILITY: D31Ø ------ Row 1 to 8 of 134
Command ===>                              Number of members found : 134

Inventory is built on 17/Ø6/Ø3 with 23173    members of 378    datasets.

Sort Commands: SORT < Dset/Member/Volume/Chgdt/Uid/Crtdt >   (On Command
line )
Action Characters:
(Issue it on the member you want to Browse/Edit/Delete/Print/Get info)

 B  Browse   E  Edit   D  Delete   P  Print   G  Get more info   C  Copy

 * To get a listing of this screen and then print it, issue "LSTNG"
command.
 * To see the scope of the Inventory, (disk volumes), issue "AVOLS"
command.


  Member   Dataset Name                 Crtdt    Volume Chgdt    Uid
  -------- ---------------------------- -------- ------ -------- ------
-
$TRASJCL DES.GNI.TRAB-FAC.JCL          Ø1/11/Ø7 DWR3Ø5 Ø3/Ø3/27 INDXAØ9
$TRASJCL INDXA11.SOURCE.JCL            99/Ø7/Ø6 DWR3Ø7 Ø3/Ø6/1Ø INDXA11
$JCLNEWH INEXØØ7.JOBS.JCL              Ø1/1Ø/19 DWR3Ø5 Ø2/Ø5/13 INEXØØ8
$JCLNEWH INEXØØ8.JCL                   Ø1/1Ø/19 DWR3Ø3 Ø3/Ø3/Ø7 INEXØØ8
GPJCLRAM DES.GNI.TRAB-COB.JCL          Ø3/Ø3/28 DWR3Ø4 Ø3/Ø4/Ø1 INDXMXD
GRANTJCL INSXGØ4.DB2.SQL               ØØ/11/Ø6 DWR3Ø1 ØØ/11/Ø6 INSXGØ4
HSMJCLBM EXP.HSM.REXX                    N/A      DWR3Ø6 N/A      N/A
************************************************************************
```

From this point on, we can realize all basic member operations on any members displayed on the panel, such as browse, edit, delete, print, or copy.

We are also given the opportunity to take a look at the actual status of the member in real-time with the ‘Get more info’ option.

(The member we have chosen may have updated since the inventory dataset was built.) If this is the case, what we see on the screen about the last change date or last changed userid would be wrong. So the G action is applied on the member and the real-time statistics are fetched and presented to us using the LMOPEN/LMMLIST commands. On the other hand, if a member is deleted from its library after the inventory is built, the utility will still keep its entry in the inventory. In this case, if a query is made to display that member, no action will be available to the user, since the utility will detect that the member is already deleted and will notify the user about it.

In addition to the actions on members, we have many more things to do on the Query Output panel:

- SORT command – we can sort the query result table by any field such as volser, dataset name, member name, member creation date, member last-update date, or userid that changed the member last.
- LSTNG command – with this command, we can get a listing of the query output panel and we can print it.
- AVOLS command – this command provides us with a list of volume serial numbers in a panel, which form the scope of the utility set by DCOLLECT control statement used in the job JCL1.

For example, If the DCOLLECT statement has the VOLUMES(*) keyword, the scope will have all disk volumes visible to the operating system. Of course these volumes must have at least one partitioned dataset containing at least one member; otherwise, they are not taken into account in the scope.

However, if the DCOLLECT has VOLUMES(DWR*), the scope will have just disk volsers containing DWR in the first three positions.

THE SEARCH TYPES THAT CAN BE REALIZED BY THE UTILITY

There are four different kinds of search that can be realized by the user on the Query Input panel. They are generic, specific, sub-string, and mask:

- Generic search – with this search, the user can find all members in the system starting with any character string. If a user needs a model IEBCOPY job, he enters IEBC and makes a generic search. He is most likely to find at least one member that executes the IEBCOPY program among all members whose names start with the characters IEBC.

  For example, if the user enters DGT and chooses G (generic search), all members with the prefix of DGT will be found and the user will know where they are located. As it is a known prefix in the DFSMS terminology, those members which are in the ISMF panel, message, and REXX libraries will also be included on the Query Output panel, since all these ISPF libraries have members starting with DGT.

  The PANELOUT panel would contain some members such as:

```
Member   Dataset Name
-------- -------------
DGTQCATL SYS1.DGTCLIB
DGTQCATL SYS1.DGTCLIB
DGTQALØ1 SYS1.DGTCLIB
DGTTTVAØ SYS1.DGTLLIB
DGTTTVAØ SYS1.DGTLLIB
DGTTSGF1 SYS1.DGTLLIB
```

- Specific search – this search is very useful if we don't remember the location of a specific member or if we want to find out all those partitioned datasets which have a specific member in common in their member listings. The action character to choose on the Query Input panel is S (specific search).

- Sub-string search – sometimes we want a list of all members having a common character string in their names. For example, if a user would like to list all those members whose name includes the word TAPE, the user chooses the T option (sub-string search) and the utility fetches all members meeting the user's criterion:

```
Member   Dataset Name
-------- --------------------
COPYTAPE INSXØØ9.JOBS.JCL
DELTAPE  EXP.SEX.JCL
DELTAPE1 EXP.SEX.JCL
DITFTAPE DIT.V1R3MØ.SDITPLIB
TAPEINIT INSXØ14.SMS.CNTL
XTAPERMS TCPIP.SEZAXTLB
```

- Mask search – in this search, a user enters a mask-member, consisting of a combination of constants and the mask-character. The mask character is a per cent sign (%). The action character to choose on the Query Input panel is M (mask search).

  A user can specify a mask member of eight characters, which may have one or more mask characters: % represents exactly one non-blank character, while %%% represents three character positions. According to this nomenclature, IMS%%%%% means all eight-character names beginning IMS, and %%WK%%%T means all names where the third and fourth characters of the name are WK and the last character is T.

  The records in the inventory dataset are scanned and those members meeting the user-specified mask are found and presented to the user. For example, query of the mask-member %%NEL%%% would give output something like this:

```
Member   Dataset Name
-------- -----------------
PANELIN  EXP.MEMSRCH.CNTL
PANEL7   EXP.MEMSRCH.CNTL
PANEL6   EXP.MEMSRCH.CNTL
PANEL5   EXP.MEMSRCH.CNTL
PANELFTP INEXØ11.REXX.CLIST
```

DATASETS USED IN THE UTILITY

The following list shows dataset name, dsorg, recfm, lrecl, and description:

- EXP.MEMSRCH.DCOLLECT PS VB 644 DCOLLECT dataset.
- EXP.MEMSRCH.JOURNAL PS FB 90 journal for deleted members.
- EXP.MEMSRCH.OUTPUT PS FB 86 dataset containing query results.
- EXP.MEMSRCH.SEQ PS FB 86 inventory (sequential, temporary).
- EXP.MEMSRCH.VSAM VS 86 inventory (VSAM, permanent).
- EXP.MEMSRCH.BATCH PS FB 50 batch query input file.

USING THE UTILITY IN BATCH

To be able to use the utility in batch, REXX3 is provided. In order to run this batch utility, all queries must be entered into a plain sequential dataset (batch query input dataset). Then the job JCL2 is submitted in which REXX3 will be run in the background. The query results will be routed to SYSTSPRT sysout dataset, which is the sysprint of IKJEFT01 program.

Batch member search is very useful if a user wants to make several queries at a time and get the results altogether in a single output file.

THE RECORD LAYOUT OF THE QUERY INPUT FILE

The record layout of the query input file showing column range and field is:

- 1-8 – member name
- 10 – search type.

SAMPLE QUERY INPUT FILE FOR THE BATCH MEMBER SEARCH

```
BROWSE    EXP.MEMSRCH.BATCH
 Command ===>
```

```
******************************** Top of Data ******
*----------------------------------------------
*  To execute 'Member Search utility' in batch,
*  this dataset is used to enter the query
*  input records.
*----------------------------------------------
* MEMBER SEARCH TYPE
IKJ      G
REXX1    S
HSMR     T
AD%%%%11 M
******************************** Bottom of Data ****
```

SAMPLE OUTPUT OF THE BATCH MEMBER SEARCH

```
QUERY : 1
Member name.......: IKJ
Search type.......: GENERIC
Number of records.: 17

Member   Data-set-Name                          Volume Crtdte   Chgdte
======== ====================================== ====== ======== ========
IKJEFF1Ø SISTEMES.CBT135.LOAD                   DSI3Ø3 ØØ/12/Ø5 Ø1/Ø3/Ø1
IKJEFF53 SISTEMES.CBT135.LOAD                   DSI3Ø3 ØØ/12/Ø5 Ø1/Ø3/Ø1
IKJEFTØ1 INSXCØ2.MY.REXX                        DWR3Ø2 N/A N/A N/A
IKJTSO   SIS.PROCLIB.OLD                        DSI3Ø1 96/Ø5/22 98/Ø1/Ø9
IKJTSOØØ SIS.PARMLIB                            DSI3Ø2 97/1Ø/17 98/1Ø/22
IKJT9FI  SISTEMES.CBT135.LOAD                   DSI3Ø3 ØØ/12/Ø5 Ø1/Ø3/Ø1
IKJT9LB  SISTEMES.CBT135.LOAD                   DSI3Ø3 ØØ/12/Ø5 Ø1/Ø3/Ø1
-----------------------------------------------------------------------
QUERY : 2
Member name.......: REXX1
Search type.......: SPECIFIC
Number of records.: 1

Member   Data-set-Name                          Volume Crtdte   Chgdte
======== ====================================== ====== ======== ========
REXX1    EXP.MEMSRCH.CNTL                       DWR3Ø3 N/A N/A N/A

-----------------------------------------------------------------------
QUERY : 3
Member name.......: HSMR
Search Type.......: SUB-STRING
Number of records.: 15

Member   Data-set-Name                          Volume Crtdte   Chgdte
======== ====================================== ====== ======== ========
DSNHSMR1 DB271Ø.ADSNLOAD                        DSI3Ø3 98/Ø1/13 98/Ø1/13
HSMREXBM EXP.HSM.REXX                           DWR3Ø6 N/A N/A N/A
HSMREXBS EXP.HSM.REXX                           DWR3Ø6 Ø3/Ø4/24 Ø3/Ø4/3Ø
```

```
HSMREXLG EXP.HSM.REXX                                DWR3Ø6 N/A N/A N/A

------------------------------------------------------------------------
QUERY : 4
Member name.......: AD%%%%11
Search Type.......: MASK
Number of records.: 3

Member   Data-set-Name                               Volume Crtdte   Chgdte
======== ======================================= ====== ======== ========
ADRDSS11 INEXØØ7.JOBS.JCL                            DWR3Ø5 99/Ø6/Ø2 99/Ø6/Ø2
ADRDSS11 INSXGØ4.JOBS.JCL                            DWR3Ø7 ØØ/Ø4/17 ØØ/Ø5/2Ø
AD36BØ11 ENPRUØ.AD.LOAD                              DSI3Ø1 ØØ/1Ø/17 ØØ/1Ø/24
```

ADDITIONAL NOTES ON THE UTILITY

The job JCL1 has to be scheduled and executed by an authorized started task or user so that the inventory dataset can have as many members as the MVS system would have. This will make the inventory more versatile. If the inventory is built by a user who has limited security access, the inventory may not be able to catch all the members in the system. Since the inventory will not represent the actual member status in the system, queries by users against the inventory would produce an incomplete result, which misleads those users.

DCOLLECT output is the source dataset of the inventory. The job JCL1 of the utility has the following DCOLLECT control statement:

```
//SYSIN    DD   *
 DCOLLECT OUTFILE(DCOUT) VOLUMES(*)
```

All volumes that DCOLLECT reaches are involved here. However, if you want to reduce the scope of the inventory, you can change it. For example, if your concern is to be able to make searches only in the partitioned datasets located on disk volumes DWR* and DSR*, we would code this DCOLLECT statement in the following way:

```
//SYSIN    DD   *
 DCOLLECT OUTFILE(DCOUT) VOLUMES(DWR*,DSR*)
```

The utility also supports uncatalogued datasets, which tend to be non-SMS datasets. So the query output may contain some members whose owning datasets are uncatalogued.

The utility maintains a journal dataset for members which are deleted through the Query Output panel. Every deleted member occupies one record in the journal. The journal dataset is created automatically. Here is a sample contents for this special dataset:

```
BROWSE    EXP.MEMSRCH.JOURNAL
 Command ===>

********************************* Top of Data ****************

         JOURNAL LOG FOR DELETED MEMBERS


USERID  DELETION DATE & TIME VOLUME DATASET & MEMBER NAME

======= ==================== ====== ======================

INEXØØ4 1 Jul 2ØØ3 11:ØØ:ØØ DWR3Ø1 EXP.HSM.CNTL(DENIS)

INSXØ14 2 Jul 2ØØ3 Ø9:Ø1:33 DWR3Ø3 EXP.GNI.CNTL(ZEYCAN)

SDIAGAS 3 Jul 2ØØ3 2Ø:Ø8:42 DSR3Ø5 SDIAGAS.USER.JCL(DIVINA)

USER1   3 Jul 2ØØ3 2Ø:Ø8:51 DWR3Ø1 SIS.SMSM.CNTL(MARTAPV)

******************************** Bottom of Data **************
```

JCL1

```
//INSXØ141 JOB MSGCLASS=X,MSGLEVEL=(1,1),CLASS=E
//*-----------------------------------------------------------------*
//* JCL NAME : Jcl1                                                  *
//* FUNCTION :                                                       *
//* Build the Member Inventory dataset, which is a Vsam Ksds dataset.*
//*                                                                  *
//* DATASETS USED :                                                 *
//* Exp.Memsrch.Dcollect     (In) -> Dcollect dataset.              *
//* Exp.Memsrch.Seq  (*)    (Out) -> Member inventory temp.dset (SEQL) *
//* Exp.Memsrch.Vsam        (Out) -> Member inventory dataset  (VSAM) *
//*                                                                  *
//* ( * ) : This dataset is temporary and will be deleted at the end  *
//*         of this job.                                             *
//*-----------------------------------------------------------------*
//* Clean-up Dcollect and Inventory datasets.                       *
//*------------------------------------------------------------------
//PASOØ    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
```

```
 DELETE EXP.MEMSRCH.DCOLLECT
 DELETE EXP.MEMSRCH.SEQ
 DELETE EXP.MEMSRCH.VSAM
//*------------------------------------------------------------------
//* Execute Dcollect against all volumes in the system.             *
//* Note that, you can limit the Dcollect scope by filtering volumes.  *
//* For example, to allow member search to be done in the PEX* disks,  *
//* code the following DCOLLECT statement:                          *
//*   DCOLLECT OUTFILE(DCOUT) VOLUMES(PEX*)                         *
//*------------------------------------------------------------------
//PASO1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//DCOUT    DD   DSN=EXP.MEMSRCH.DCOLLECT,DISP=(,CATLG,DELETE),
// SPACE=(CYL,(14,2),RLSE),DSORG=PS,RECFM=VB,LRECL=644
//SYSIN    DD   *
 DCOLLECT OUTFILE(DCOUT) VOLUMES(*)
 SET MAXCC=Ø
//*------------------------------------------------------------------
//* Allocate Inventory dataset (SEQL).                             *
//*------------------------------------------------------------------
//PASO2    EXEC PGM=IEFBR14
//SYSPRINT DD   SYSOUT=*
//ALLOC    DD   DSN=EXP.MEMSRCH.SEQ,DISP=(,CATLG,DELETE),
// SPACE=(CYL,(5Ø,1Ø),RLSE),DSORG=PS,RECFM=FB,LRECL=86
//SYSIN    DD   DUMMY
//*------------------------------------------------------------------
//* Build Inventory dataset (SEQL) by executing the Rexx  "REXX1".    *
//*------------------------------------------------------------------
//PASO3    EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø,REGION=9ØM
//SYSPROC  DD   DISP=SHR,DSN=EXP.MEMSRCH.CNTL
//         DD   DISP=SHR,DSN=SIS.EXEC
//DCOLIN   DD   DISP=SHR,DSN=EXP.MEMSRCH.DCOLLECT
//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(%REXX1) +
 BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(99999999)
//*------------------------------------------------------------------
//* Allocate Inventory dataset (VSAM).                             *
//*------------------------------------------------------------------
//PASO4    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
```

```
 DEFINE  CLUSTER                         -
          (                              -
          NAME (EXP.MEMSRCH.VSAM)        -
          CYLINDERS(1ØØ 2Ø)              -
          SHAREOPTIONS (3)               -
          RECORDSIZE (86 86)             -
          INDEXED                        -
          NOREUSE                        -
          VOLUMES(DWR3Ø1)                -
          KEYS(6Ø Ø)                     -
          FREESPACE (Ø Ø))
//*-------------------------------------------------------------------
//* Build the Inventory dataset (VSAM) by IDCAMS Repro.             *
//*-------------------------------------------------------------------
//PASO4    EXEC PGM=IDCAMS
//IN       DD   DISP=SHR,DSN=EXP.MEMSRCH.SEQ
//OUT      DD   DISP=SHR,DSN=EXP.MEMSRCH.VSAM
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
  REPRO INFILE(IN) OUTFILE(OUT)
//*-------------------------------------------------------------------
//* Delete DCOLLECT and sequential INVENTORY datasets.              *
//*-------------------------------------------------------------------
//PASO5    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
 DELETE EXP.MEMSRCH.DCOLLECT
 DELETE EXP.MEMSRCH.SEQ
```

JCL2

```
//INSXØ141 JOB MSGCLASS=X,MSGLEVEL=(1,1),CLASS=E
//*------------------------------------------------------------------*
//* JCL NAME : Jcl2                                                  *
//* FUNCTION : Executes the Member Search Utility IN BATCH.          *
//*------------------------------------------------------------------*
//PASO3    EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø,REGION=9ØM
//SYSPROC  DD   DISP=SHR,DSN=EXP.MEMSRCH.CNTL
//         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(%REXX3) +
 BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(99999999)
//*
```

PANEL1

```
)ATTR
 % TYPE(TEXT)   COLOR(WHITE) CAPS(OFF) HILITE(USCORE) INTENS(LOW)
 # TYPE(TEXT)   COLOR(BLUE)  CAPS(OFF)
 @ TYPE(TEXT)   COLOR(RED)   CAPS(OFF)
 + TYPE(TEXT)   COLOR(TURQ)  CAPS(OFF) JUST(LEFT)
 $ TYPE(OUTPUT) COLOR(YELLOW)
 [ TYPE(OUTPUT) COLOR(PINK)  CAPS(OFF)
)BODY WINDOW(47,19)
+
+#Data-Set-Name+
+$DSET                                       +
+
+#Member:$MEM     #On Volume:$VOL   +
+
+ MEMBER STATISTICS
+ Creation Date..........:[ZLCDATE +
+ Date Modified..........:[ZLMDATE +
+ Time Modified..........:[ZLMTIME +
+ Current No. Lines......:[ZLCNORC +
+ Initial No. Lines......:[ZLINORC +
+ No. Modified Lines.....:[ZLMNORC +
+ Who Created / Updated..:[ZLUSER  +
+ Version Number.........:[ZLVERS  +
+ Modification Level.....:[ZLMOD   +
+ Updated by SCLM (Y/N)..:[ZSCLM   +
 %Press@<Enter>%to exit.+
)INIT
Vget (MD,CD) Profile
)PROC
&Spfkey = .Pfkey
Vput Spfkey Profile
)END
```

PANEL2

```
)ATTR
 % TYPE(TEXT)   COLOR(WHITE) CAPS(OFF) HILITE(USCORE) INTENS(LOW)
 # TYPE(TEXT)   COLOR(BLUE)  CAPS(OFF)
 @ TYPE(TEXT)   COLOR(RED)   CAPS(OFF)
 + TYPE(TEXT)   COLOR(TURQ)  CAPS(OFF) JUST(LEFT)
 $ TYPE(OUTPUT) COLOR(YELLOW)
 [ TYPE(OUTPUT) COLOR(PINK)  CAPS(OFF)
```

```
)BODY WINDOW(47,16)
+
+#Data-Set-Name+
+$DSET                                   +
+
+#Member:$MEM     #On Volume:$VOL   +
+
+ MEMBER STATISTICS
+ Load module size (hex).:[ZLSIZE  +
+ TTR of the member......:[ZLTTR   +
+ Alias Name.............:[ZLALIAS +
+ Authorization Code.....:[ZLAC    +
+ AMODE..................:[ZLAMODE +
+ RMODE..................:[ZLRMODE +
+ Load Module Attributes.:[ZLATTR  +
 %Press@<Enter>%to exit.+
)INIT
)PROC
&Spfkey = .Pfkey
Vput Spfkey Profile
)END
```

PANEL3

```
)ATTR
 # TYPE(TEXT)   COLOR(BLUE)  CAPS(OFF)
 + TYPE(TEXT)   COLOR(GREEN) CAPS(OFF)
 $ TYPE(INPUT)  COLOR(RED) HILITE(USCORE)
 @ TYPE(OUTPUT) COLOR(TURQ)
 { TYPE(OUTPUT) COLOR(PINK)
)BODY WINDOW(53,3)
#Want to delete the member{K1      $Z+ ..?(Enter Y/N)
+ Dataset : @K2
+ Volume   : @K3
)INIT
.Zvars    = '(Answer)'
)PROC
&Spfkey = .Pfkey
Vput Spfkey Profile
Ver (&Answer,LIST,Y,N)
)END
```

PANEL4

```
)ATTR
 @ TYPE(OUTPUT) COLOR(TURQ) CAPS(OFF)
)BODY WINDOW(74,2)
@MESAJ
```

```
@MESAJ2
)INIT
)PROC
&Spfkey = .Pfkey
)END
```

PANEL5

```
)ATTR
 % TYPE(TEXT)  COLOR(WHITE) CAPS(OFF) HILITE(USCORE) INTENS(LOW)
 @ TYPE(TEXT)  COLOR(RED)   CAPS(OFF) HILITE(USCORE)
 + TYPE(TEXT)  COLOR(TURQ)  CAPS(OFF) JUST(LEFT)
 $ TYPE(TEXT)  COLOR(GREEN) HILITE(REVERSE)
 [ TYPE(INPUT) COLOR(PINK)  CAPS(ON)  HILITE(REVERSE)
)BODY WINDOW(32,9)
+
$        DO YOU WANT TO PRINT
$          REPORT / MEMBER
+
+ Yes(Y) / No(N).....:[Z+
+ Destination........:[Z       +
+ Output Class.......:[Z+
+
% Hit@<Enter>%to exit / print.+
)INIT
.Zvars = '( Resp Dest Class)'
&Resp   = 'N'
&Dest   = 'LOCAL'
&Class  = 'A'
.Cursor = RESP

)PROC
&Spfkey = .Pfkey
   Vput Spfkey Profile

Ver (&Resp,Nonblank)
Ver (&Resp,List,Y,N)
                                        /*---------------------------*/
If (&Resp = 'Y')                        /* If user wants to print the */
 Ver (&Dest,Nonblank)                   /* report, check the values   */
 Ver (&Dest,Include,Alphab,Num)         /* entered for the Destination*/
 &V = Trunc(&Dest,1)                    /* and Class fields.          */
 Ver (&V,Alphab)                        /*---------------------------*/

 Ver (&Class,Nonblank)
 Ver (&Class,Include,Alphab,Num)
 Vput (Resp Dest Class) Profile
)END
```

PANEL6

```
)ATTR
 + TYPE(TEXT)   COLOR(TURQ)  CAPS(OFF) JUST(LEFT)
 $ TYPE(TEXT)   COLOR(GREEN)
 [ TYPE(OUTPUT) COLOR(PINK)  CAPS(ON)
)BODY WINDOW(74,12)
$        ALL VOLUME SERIAL NUMBERS WITHIN DCOLLECT SCOPE
+
[R1     [R2     [R3     [R4     [R5     [R6     [R7     [R8      [R9     [R1Ø
[R11    [R12    [R13    [R14    [R15    [R16    [R17    [R18     [R19    [R2Ø
[R21    [R22    [R23    [R24    [R25    [R26    [R27    [R28     [R29    [R3Ø
[R31    [R32    [R33    [R34    [R35    [R36    [R37    [R38     [R39    [R4Ø
[R41    [R42    [R43    [R44    [R45    [R46    [R47    [R48     [R49    [R5Ø
[R51    [R52    [R53    [R54    [R55    [R56    [R57    [R58     [R59    [R6Ø
[R61    [R62    [R63    [R64    [R65    [R66    [R67    [R68     [R69    [R7Ø
[R71    [R72    [R73    [R74    [R75    [R76    [R77    [R78     [R79    [R8Ø
[R81    [R82    [R83    [R84    [R85    [R86    [R87    [R88     [R89    [R9Ø
[R91    [R92    [R93    [R94    [R95    [R96    [R97    [R98     [R99    [R1ØØ
)INIT
)PROC
)END
```

PANEL7

```
)ATTR
 # TYPE(TEXT)  COLOR(PINK)  CAPS(OFF)
 + TYPE(TEXT)  COLOR(GREEN) CAPS(OFF)
 @ TYPE(INPUT) COLOR(TURQ)
)BODY WINDOW(6Ø,4)
#Enter TARGET dataset and member :
+
+Dataset :@Tdset                                 +
+Member   :@Tmem    +
)INIT
)PROC
&Spfkey = .Pfkey
Vput Spfkey Profile

Ver (&Tdset,Nonblank,Dsname)
Ver (&Tmem,Nonblank,Name)
)END
```

PANELIN

```
)ATTR
    % TYPE(TEXT)   INTENS(HIGH) COLOR(TURQ)   CAPS(OFF)
    + TYPE(TEXT)   INTENS(LOW)
    ] TYPE(TEXT)   INTENS(HIGH) COLOR(GREEN)
    [ TYPE(TEXT)   INTENS(HIGH) COLOR(GREEN)  HILITE(REVERSE)
    @ TYPE(OUTPUT) INTENS(HIGH) COLOR(GREEN)
    ! TYPE(OUTPUT) INTENS(HIGH) COLOR(BLUE)   HILITE(REVERSE) CAPS(OFF)
    _ TYPE(INPUT)  INTENS(HIGH) COLOR(WHITE)  HILITE(USCORE)
    $ TYPE(OUTPUT) INTENS(HIGH) COLOR(YELLOW) HILITE(REVERSE) JUST(ASIS)
    # TYPE(OUTPUT) INTENS(HIGH) COLOR(WHITE)  CAPS(OFF)
    } TYPE(OUTPUT) INTENS(LOW)  COLOR(PINK)   HILITE(USCORE)
)BODY
%-------------------[MEMBER SEARCH UTILITY:$ENVR%----------------------
%
%
%CHOOSE SEARCH TYPE ===>_Z%  ENTER MEMBER NAME / SUB-STRING / MASK-
FIELD)
%
+(G) Generic      ------
+(S) Specific           | ==>}Z       +
+(T) Sub-string   ------
+
+
+(M) Mask               ==>}Z       + #StØ1
+                                     #StØ2
+
+
#St1
#St2
+
#St3
#St4
+
+Inventory is built on@Z       +with@Z       +members of@Z   +datasets,
+which are spread out to @Z   +disk volumes.
+
]                Press%PFØ3]or%PFØ4] to exit from this panel.
)INIT
.Zvars = '(Type,Membrec,Mask,Blddate,Nummemb,Numdset,Numvol)'
&Mask    = '-M-A-S-K'
&Membrec = '-MEMBER-'
.Cursor = Type
&MeØ='XXXXXXXX'

&Me1='For GENERIC search, enter one or more characters into the Member'
&Me2='  field.                                                        '
&Me2='For example, to search all member names where their first 3     '
&Me4='characters are "SMS", enter the word "SMS" into the Member field.'

&Me5='For SPECIFIC search, enter the exact member name into the Member'
&Me6='  field.                                                        '
&Me7='For example, to search all member names called "IDCAMS", enter '
&Me8='the word "IDCAMS" into the Member field.                       '
```

```
&Mea='For MASK search, enter a string of 8 characters. It may have one '
&Meb='or more "%" character, which is called "Mask character".         '
&Mec='Example: To search all members where fourth, fifth and sixth
characters'
&Med='are "JCL" and the last character is "1", use this mask:
"%%%JCL%1".    '

&Mea1='For SUB-STRING search, enter any string of 1 to 8 char.long. It
is used'
&Meb1='to search all member names that have a character string IN
COMMON.    '
&Mec1='Example: To search all members where the string "JCL" is found '
&Med1='in the member name, enter the word "JCL" into the Member field. '

Vget (Envr) Profile

)REINIT
 Refresh(StØ1,StØ2,St1,St2,St3,St4,Membrec,Mask)
)PROC
&Spfkey = .Pfkey
Ver  (&Type,Nonblank)                        /* Search Type control. */
Ver  (&Type,List,S,G,T,M)
If (&Type='G')            /*--------------------------------*/
    &St1= &Me1            /* Generic    Search explanation. */
    &St2= &Me2            /*--------------------------------*/
    &St3= &Me3
    &St4= &Me4
If (&Type='S')            /*--------------------------------*/
    &St1= &Me5            /* Specific   Search explanation. */
    &St2= &Me6            /*--------------------------------*/
    &St3= &Me7
    &St4= &Me8
If (&Type='M')            /*--------------------------------*/
    &St1= &Mea            /* Masking    Search explanation. */
    &St2= &Meb            /*--------------------------------*/
    &St3= &Mec
    &St4= &Med
If (&Type='T')            /*--------------------------------*/
    &St1= &Mea1           /* Sub-string Search explanation. */
    &St2= &Meb1           /*--------------------------------*/
    &St3= &Mec1
    &St4= &Med1

If (&Type = M )
     .Attr (Mask)    = ' Type(Input) Color(Green) '
     Ver  (&Mask,Nonblank,NameF)
     Ver  (&Mask,Len,EQ,8)
     &StØ1 = 'Enter a mask-field of 8 characters long.'
     &StØ2 = '( Mask character is "%".)'
```

```
If (&Type = S,G,T)
     .Attr (Membrec) = ' Type(Input) Color(Green) '
     Ver  (&Membrec,Nonblank,Name)
     &StØ1 = &Z
     &StØ2 = &Z

Vput (Spfkey,Membrec,Mask,Type) Profile
)END
```

PANELOUT

```
)ATTR DEFAULT(%+_)
     # TYPE(INPUT)  INTENS(HIGH) COLOR(PINK) HILITE(USCORE) CAPS(ON)
     @ TYPE(OUTPUT) INTENS(HIGH) COLOR(GREEN)
     [ TYPE(TEXT)   INTENS(HIGH) COLOR(GREEN) HILITE(REVERSE)
     % TYPE(TEXT)   INTENS(HIGH) COLOR(TURQ) CAPS(OFF)
     $ TYPE(OUTPUT) INTENS(HIGH) JUST(ASIS) COLOR(YELLOW)
HILITE(REVERSE)
     | TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(RED)
     } TYPE(TEXT)   INTENS(HIGH) COLOR(PINK) HILITE(REVERSE)
     { TYPE(TEXT)   INTENS(HIGH) COLOR(YELLOW)
)Body
%----------------------[MEMBER SEARCH UTILITY:$ENVR%--------------------
%Command ===>_Pcmd                         +Number of members found :@qc
+
+Inventory is built on@Z      +with@Z      +members of@Z    +datasets.
+
{Sort Commands:+SORT <|Dset/Member/Volume/Chgdt/Uid/Crtdt+>   (On
Command line )
{Action Characters:
+(Issue it on the member you want to Browse/Edit/Delete/Print/Get info).
+
} B + Browse  } E +Edit  } D +Delete  } P +Print  } G +Get more info  }
C +Copy
+
{ * +To get a listing of this screen and then print it,
issue%"LSTNG"+command.
{ * +To see the scope of the Inventory, (disk volumes),
issue%"AVOLS"+command.
+
+ %Member   Dataset Name                      Crtdt    %Volume%Chgdt
%Uid
+ %--------%----------------------------------%--------%------%--------
%-------
)Model
#L@K1     @K2                                @K4      @K3    @K5
@K6
)INIT
.Zvars    = '(Blddate,Nummemb,Numdset)'
```

3 Some useful ISPF utilities - 140z
9 A fresh look at Java for OS/390 – enhancements in functionality, scalability, and performance - 142
25 System programming C facilities - 128 ~~Anthony Rudd~~
32 Catalogued non-existent datasets - 115
36 Listing online volumes - 119
45 Using CSI to identify VSAM datasets defined with IMBED, REPLICATE, and KEYRANGE – revisited
46 Possible rounding errors in COBOL on the mainframe – update - 137
49 System-wide member search utility - 129

106 <claude.dunand@free.fr> - 77
111 Records with duplicate fields in a file - 82
118 NOTIFYEXTENT Catalog Option - 99
126 Analysing HSM dump volumes - 112
134 Cloning an MVS system to another workshop - 113
140 Analysing Data-in-Virtual statistics - 118
156 WAIT4FTP - 120
165 Contingency planning and problem resolution - m111 hold
171 A peek into SMF30 data - 126
192 Splitting a file across several files - 127
263 An IPCS VERBEXIT Routine for Displaying NAME/TOKEN Lists - 130
292 Space Abend reporter - 130
303 DFHSM Data Set Backup Version Record Inventory - 132b
370 HFS files at glance - 133
377 Data conversion - 134
391 NOTIFYEXTENT Catalog Option - 135
394 PCGLCMDX – z/OS Commands Installation Exit - 136
410 DUPLA and DEDUPLA - 138
435 Some Useful ISPF Utilities - 140z
442 New Z990 Channel Subsystem - Patrick Renard

```
&L        = &Z
&Pcmd     = &Z
&Ztdsels = Ø
Vget (Envr) Profile
)REINIT
If (.Msg = ' ')
    &L  = &Z
    Refresh(L)
)PROC
&Spfkey = .Pfkey
Vput Spfkey Profile
Ver (&L,LIST,B,E,D,G,P,C)
                               /*-----------------------------------*/
If (&Ztdsels NE ØØØØ)          /* Without completing the operation on */
  Ver(&L,Nonblank)             /* the current row, any other row can  */
Vput (Pcmd L) Profile          /* not be chosen.                      */
)END                           /*-----------------------------------*/
```

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

Atalay Gul
Systems Programmer
Azertia SA (Spain)

MVS news

Mainstar Software has announced the latest version of Mirroring Solutions/Volume Conflict Rename (MS/VCR). The product is designed to make access to cloned data easier, even with multiple sites or where data is moved frequently.

MS/VCR provides access to point-in-time copies made from 'cloned' data. The product renames and catalogs the cloned data, fixes VTOC, VTOCIX, and VVDS conflicts, and can be used to update DB2 internal control information.

MS/VCR now offers a new command, VOLOPTIONS (LIST | CLIP | UPDATE(NEWTARGETS)), which is specifically designed for situations when the MS/VCR COPY step is run at one site (SITEA) and the MS/VCR RENAME step is run at another site (SITEB) or for data that has been moved multiple times at the primary site.

For further information contact:
Mainstar Software, PO BOX 4132, Bellevue, WA 98009-4132, USA.
Tel: (425) 455 3589.
URL: http://www.mainstar.com/products/msvcr/index.asp.

* * *

Micro Focus has announced that its Mainframe Express, a workstation-based development environment for IBM mainframe business applications, has been certified CA smart with AllFusion Endevor Change Manager, an automated mainframe software configuration management tool, from Computer Associates.

Mainframe Express and AllFusion Endevor Change Manager provide an end-to-end application development environment for z/OS. Developers now have the flexibility and productivity provided by the ability to carry out the development lifecycle on a Windows PC using Mainframe Express instead of using mainframe host resources, while making use of the security and scalability of CA's host-based source control management systems.

AllFusion Endevor Change Manager enables organizations to control all software management tasks associated with the mainframe development environment through its automated transformation functionality, module relationship management, parallel development management, and release automation. It eliminates the manual steps that slow down the software development process and ensures greater efficiency with fewer errors.

For further information contact:
Information Builders, Two Penn Plaza, New York, NY 10121-2898, USA.
Tel: (212) 736 4433.
URL: http://www.informationbuilders.com/products/index.html.

* * *

Catalyst Systems has announced Version 6.2 of application build management product Openmake. This new release of Openmake focuses on redefining the way developers build and deploy Java applications of all kinds. Openmake Version 6.2 supports Enterprise Builds on z/OS and USS.

For further information contact:
Catalyst Systems, PO Box 556, Glencoe, IL 60022, USA.
Tel: (847) 835 6106.
URL: http://www.openmake.com/press/om62.html.