166 MVS update

July 2000

In this issue

3 Concurrent copy

8 Return-code special register

10 Implementing shared HFS on OS/390 Version 2 Release 9

27 Keeping track of load module changes

48 Sorting a PDS list

49 A Java client/server application on OS/390 – part 2

66 Reference modification

68 Shifting text and data in ISPF/PDF

70 OS/390 Version 2 Release 10 directions

72 MVS news

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 33598
From USA: 01144 1635 33598
E-mail: Jaimek@xephon.com

North American office
Xephon/QNA
PO Box 350100,
Westminster, CO 80035-0100
USA
Telephone: (303) 410 9344
Fax: (303) 438 0290

Contributions
Articles published in *MVS Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*, or you can download a copy from www.xephon.com/contnote.html.

Editor
Jaime Kaminski

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

MVS Update on-line
Code from *MVS Update* can be downloaded from our Web site at http://www.xephon.com/mvsupdate.html; you will need the user-id shown on your address label.

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 1992 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.

Printed in England.

Concurrent copy

INTRODUCTION

How do you carry out your mainframe disaster recovery? In this world of e-commerce and ever-increasing demands of existing OS/390 systems, it is easy to get the impression from the press and from advertising that if you have OS/390 then you probably have mirrored DASD and maybe even a hot site. Would that this were the case. Many sites cannot justify the cost for such systems, yet still wish to improve their availability and minimize their down time for functions such as back-up. Having been involved with such a site for a while now, I would like to discuss the merits of exploiting what is an old, yet seemingly not widely-used technology, that nearly all of us probably have and which could help reduce down time. This is concurrent copy.

WHAT IS CONCURRENT COPY?

Concurrent copy is, at its most basic, a tool that can be invoked during DFDSS DUMP (or COPY) processing that *virtually* eliminates the processing time involved with doing the DUMP. What happens is this:

- DFDSS initiates a concurrent copy session with 3990 compatible controllers. At this point, a track map of all selected data is taken within the controller so that a complete location map exists of all the disk areas that will be transferred.
- Once this map is built, the dump processing actually begins, and, most importantly, all the ENQs on the data are released leaving the data free for use by other systems. Furthermore a log message is issued enabling automation tools to release applications back to the system.
- Should one of the systems which then gets access to the data attempt to update one of the datasets being dumped, the controller will detect from its track map that a change is to be made. As a result the 'about to be updated' track is uploaded into an area of the controller's storage known as a sidefile (thus protecting the original data from being lost), and the update can then take place.

This sidefile occupies about 1MB of the controller storage at most, so if the amount of data being updated exceeds this sidefile, then the data is transferred into expanded storage and then subsequently to paging areas for safe keeping. This data is associated with an address space known as ANTMAIN. The size of this paging requirement ebbs and flows depending on how widely DFDSS is able to get the data transferred to tape (or the copy disk).

The net result is that your DFDSS dumps are now reduced to the time taken to complete the mapping process the duration of which is highly dependent upon how much work DFDSS has to do to identify all the relevant datasets to dump. It will, however, be considerably less than that for the whole dump!

FREE LUNCH?

Unfortunately, concurrent copy comes under the category of 'no such thing as a free lunch'. It may seem from the above that concurrent copy is the ideal solution for the smaller site, but it does need to be implemented with care, and with understanding of the potential risks involved with using concurrent copy. The ultimate risk associated with this facility (and one that may have put people off its use to some extent) is the fact that it is a non-restartable operation. Should there be a failure during the dump process, and should any of the data being processed have been updated since the copy was initiated, it is not possible to start the copy again from its original position.

Having said that, we have used this tool at our site for somewhere around seven years now and in itself it is very reliable. However, there are a number of issues to look out for which can affect its use as follows:

- Ensure that you have spare page datasets available (or a large amount of free space in your existing ones) since if there is a large amount of updating going on, the updated tracks will find their way on to your page areas, and this could lead to auxiliary storage shortages.
- If there is a large amount of activity, ANTMAIN may well 'swamp' expanded storage as well, so try to ensure that the dumps are not done at peak times.

- Always make sure that the output part of your dump does not become ‘blocked’ (ie if you have manual tape drives, do not let the system wait for tapes because this may cause more data to be sent to ANTMAIN). If you have a robot, do not put it into pause mode or manual mode because this will cause ANTMAIN to clog the system as well.

Concurrent copy in itself rarely seems to fail. What does compromise it is its potential impact on the system through its use of memory and auxiliary storage. With care therefore it can be safe to use.

Should it still fail despite your best efforts, for example because of a tape error or perhaps because ultimately the job had to be cancelled because of its use of resources, the question then is, is it really that much of a problem? If it is possible to take a new copy quickly, as it can be with this technique, is it possible to rerun it anyway at the point it has reached, and deal with any data inconsistencies through operational methods? That question can only be asked when you have decided on how you may want to exploit this tool in your processing schedules.

HOW IS IT INVOKED?

Although it was stated earlier in this article that concurrent copy is invoked under DFDSS, it can also be activated through HSM. Having said that, ultimately, it is DFDSS that acts as the data mover for HSM and that is the real invoker of concurrent copy.

If you wish to use HSM as your method of exploiting concurrent copy, then you can either do it through the use of VOLUMEDUMP(CC) on a SETSYS command for volume dumps, or through the use of management class specifications. There are two specifications within the management class – ‘back-up copy technique’ and ‘abackup copy technique’. Each can be specified as CONCURRENT REQUIRED (ie must use concurrent copy), CONCURRENT PREFERRED (ie use it if on a suitable device), and STANDARD (no concurrent copy), The back-up copy technique covers ordinary HSM incremental back-ups, while the ABACKUP category covers ABARS (aggregate back-up and restore). In order for the management class to be effective, the data must reside on concurrent copy capable devices. This may be assisted

by setting the storage class 'accessibility' option to 'continuous'(or CONT PREFD if you have a mix of DASD). Having said that, it is probable by now that the majority of sites will have concurrent copy capable controllers, so the storage class may well be irrelevant.

Use of DFDSS with concurrent copy is the easiest and most direct approach to using this facility. To show how easy it is to do a concurrent copy for back-up through DFDSS, the following is an example of a set of DFDSS parameters that are in use at our site for daily incremental back-up. I have included all of them because the overall set-up has been optimized for back-up speed and may therefore be of interest generally.

```
DUMP -
ADMIN -
CANCELERROR -
CONCURRENT -
DATASET(INCLUDE(PROD1.**,PROD2.**) -
        EXCLUDE(%%%.DSN*.**, -
       SYS1.VVDS.**) -
        BY((DSCHA EQ 1))) -
INCAT(CATALOG.PROD1,CATALOG.PROD2) -
ONLYINCAT -
OPTIMIZE(4) -
OUTDDNAME(TAPEOUT1) -
SHARE -
SPHERE -
TOL(ENQF) -
VALID -
WAIT(0,0)
```

Assuming this works satisfactorily, the following style of messages should be present in your job log:

```
ADR801I (001)-DTDSC(01), DATASET FILTERING IS COMPLETE. 4171 OF 4171
DATA SETS WERE SELECTED: 0 FAILED SERIALIZATION AND 0 FAILED FOR OTHER
REASONS.
ADR734I (001)-DTDSC(01), 2000.133 04:28:11 CONCURRENT COPY
INITIALIZATION SUCCESSFUL FOR 4171 OF 4171 SELECTED DATA SETS.
SERIALIZATION FOR THIS DATA IS RELEASED IF DFSMSDSS HELD IT. THE
INTERMEDIATE RETURN CODE IS 0004.
```

The ADR734I message will also be present on SYSLOG to allow your automation routines to restart systems and applications after concurrent copy completion. As an example of the time taken to process, the above job took just over six minutes for concurrent copy completion.

ADDITIONAL CONSIDERATIONS FOR USE

The following points should also be considered when using concurrent copy:

- Concurrent cannot be used with the DELETE and UNCATALOG DFDSS keywords. Effectively you cannot use concurrent copy where you want DFDSS to carry out a change to the data to be copied. What you can do though, if you want to take a copy and do a delete, start the delete job upon completion of concurrent copy initialization.
- By default you cannot use the RESET DFDSS keyword to setoff change flags when using concurrent copy. However this can be altered through a simple ZAP to the DFDSS patch area. Note this is a standard technique for controlling DFDSS operation and is not a 'dodgy tweak' to the product. The required ZAP is included below.
- IBM's and STK's devices with snapshot capability simulate concurrent copy through a snap file, rather than through the track map. However, the effect is the same, but the potential impact of concurrent copy on the paging areas is moved outboard to the device.

ALLOWING RESET WITH CONCURRENT COPY

All that is required to achieve RESET with concurrent copy is to set the byte at offset X'18' in ADRPATCH to a non-zero value. The ZAP deck shown below should be all that is required to make the necessary change.

```
//your job card
//STEPA  EXEC  PGM=AMASPZAP
//SYSPRINT DD SYSOUT=*
//SYSLIB DD DSN=library.where.adrdssu.resides,DISP=SHR
//SYSIN    DD *
    NAME ADRDSSU ADRPATCH
    VER   18      ØØ
    REP   18      FF
```

CONCLUSIONS

As with all functionality, it is necessary to decide if this tool suits the requirements of your site. You can easily check whether it works by simply trying the concurrent keyword in a simple DFDSS job. Then all that is required is a risk management exercise to determine if you can safely accept the possibility of using a non-restartable mechanism for your back-up. Which leads me to one final suggestion for using this facility. If you take two copies of data for security reasons and feel that concurrent copy is not for you, there may still be one possibility for its use. If one copy of data is taken in standard form, you could initiate the second copy upon completion of the first as a concurrent copy. Although the reads to the disk are potentially doubled, the actual elapsed time between completion of the first copy and concurrent copy completion of the second will be around 35% less than taking two simultaneous copies.

Return-code special register

INTRODUCTION

The RETURN-CODE special register can be used to pass a number between separately-compiled programs. This is an IBM extension to the COBOL language, so do not expect to see it in every version of ANSI COBOL that you might work with. It has the implicit definition as follows:

```
01 RETURN-CODE  GLOBAL  PIC S9999  COMP.
```

You do not have to define it in your data division.

You may use it to pass a 'return code' to the calling program or the system by setting RETURN-CODE before executing an 'EXIT program'or GOBACK statement.

The first time a program is called, the compiler initializes the field to zero (normal return code for a successful program). Under normal conditions, the register will not be reset on subsequent calls to the program. There are some rare exceptions to this rule, so look them up for your specific compiler.

When control returns to the operating system, return-code's contents are returned to the operating system as a 'user return code'.

EXAMPLE

Starting with the calling program:

```
PROCEDURE DIVISION.

    CALL 'ABSØ1Ø' USING ARGUMENT-1.
    IF RETURN-CODE = ZERO
       PERFORM MAIN-PROCESSING-ROUTINE
    ELSE
       PERFORM SUBROUTINE-ERROR.
```

Now consider the subroutine ABS010:

```
PROCEDURE DIVISION USING ARG-X.
    PERFORM DO-A-LOT-OF-STUFF.
    IF ERROR-LEVEL > 1Ø
       MOVE 8 TO RETURN-CODE.
    GOBACK.
```

When the calling program returns control to the operating system, the user return code will be '8' if ABS010 experienced an error. Many shops use '8' to indicate a serious error. '16' is usually reserved for total melt-downs. This pretty much follows the IBM usage, which is similar to the following:

```
 Ø - Successful operation
 4 - Warning ("oops")
 8 - Error
16 - Major unrecoverable error.
```

Allan Kalar
Systems Programmer (USA)

Implementing shared HFS on OS/390 Version 2 Release 9

INTRODUCTION

With Release 9, OS/390 enables Sysplex users to share read/write access to data stored in Hierarchical File Systems (HFS) among the MVS images of the Sysplex.

Before Release 9, users had to log onto one system in a Sysplex and had write access only to the file systems associated with their own system. With shared HFS, all file systems are available, in read/write access, to all systems participating in the shared HFS support. A user who is logged on to any system can make changes to shared file systems and those changes are visible to all systems of the Sysplex.

In this article we will explain how HFS Sysplex sharing works. Then we will describe how to implement it step by step.

PHYSICAL SHARED HFS IMPLEMENTATION DESIGN

XCF message services are used to communicate between systems in the Sysplex. This allows metadata, such as mount structure for example, to be maintained in a couple dataset to allow each system to know where it must communicate to access a file.

Recovery is provided, such that if one system fails another system in the Sysplex will take over the file sharing responsibilities for the failing system. This will be invisible to the application.

XCF Transfer Protocol

Every HFS dataset that is shared read/write in the Sysplex has a central owning system that manages it on behalf of other systems in the Sysplex. The owning system gets read or write requests from other systems and performs these operations on the HFS datasets.

A messaging protocol via XCF services is used to transfer data around the Sysplex from the central owner. This is like a client/server environment.

OS/390 Unix System Services Version 2 Release 9 implemented shared HFS support by using function shipping. Function shipping means that one system performs as a server to the other sharing systems (clients).

A client system ships its requests to the related server(s) for each read/write shared HFS dataset.

Only the server system issues read and write requests for both types of data (files and metadata) to an HFS dataset on a volume. The buffering and cacheing is also done within by the system owning the file system.

By default, the system that mounts the HFS dataset the first time automatically becomes the owning system.

The XCF group that is used by shared HFS is SYSBPX.

Mount requests are Sysplex wide.

The communication for the shared HFS support is implemented by using:

- XCF messaging services to inform all systems when a system joins or leaves the SYSBPX group, and to notify other systems that an update occurred in the system-wide mount table when a mount, unmount, chmount, quiesce, or unquiesce request was issued.
- A coupling dataset to track information about the participating systems and to have a Sysplex-wide mount table. Note that a coupling dataset is used, not a coupling facility structure.

Recovery

Dead system recovery is provided to dynamically recover file systems owned by a system that has left the Sysplex for any reason. Those files will be randomly moved to surviving systems in the Sysplex, providing enhanced file system availability.

LOGICAL SHARED HFS IMPLEMENTATION DESIGN

This paragraph introduces new terms and concepts that are used when

implementing shared HFS, such as Sysplex root, system-specific HFS, and Version HFS.

Sysplex root

This new HFS, which will be allocated during Shared HFS implementation, is unique in the Sysplex.

It contains only directories and symbolic links that allow redirection to system-specific HFS directories and HFS directories.

The structure of the Sysplex root HFS looks like:

Directory	/		
Symlink	/bin	–	$VERSION/bin
Symlink	/usr	–	$VERSION/usr
Symlink	/lib	–	$VERSION/lib
Symlink	/opt	–	$VERSION/opt
Symlink	/samples	–	$VERSION/samples
Symlink	$SYSNAME	–	$SYSNAME/
Symlink	$VERSION	–	$VERSION/
Symlink	/dev	–	$SYSNAME/dev
Symlink	/tmp	–	$SYSNAME/tmp
Symlink	/var	–	$SYSNAME/var
Symlink	/etc	–	$SYSNAME/etc
Directory	/u	–	automount managed.

$VERSION and $SYSNAME are system variables which will be instantiated using BPXPRM*xx* parameters (root version) and the local MVS system name.

System-specific HFS

This new HFS will also be allocated during shared HFS implementation.

There is one system-specific HFS for each MVS image participating in the Sysplex. It contains only directories and symbolic links that allow redirection to system specific directories (/dev, /tmp, /var and /etc). The structure of the system-specific file system looks like:

Symlink	/bin	–	/bin
Symlink	/usr	–	/usr
Symlink	/lib	–	/lib
Symlink	/opt	–	/opt
Symlink	/samples	–	/samples
Directory	/dev	mount point for system-specific HFS for /dev	
Directory	/tmp	mount point for system-specific HFS for /tmp	
Directory	/var	mount point for system-specific HFS for /var	
Directory	/etc	mount point for system-specific HFS for /etc.	

Version HFS

The version HFS is the IBM-supplied root HFS dataset containing files and executables for OS/390 elements.

IBM supplies this HFS in the ServerPac – it corresponds to the 'old' root HFS. It contains system code including /bin, /usr, /lib, /opt, /samples, etc. In the Sysplex environment, each system could be on a different OS/390 release or service level. You should use the VERSION statement of BPXPRM*xx* member to specify the version of the HFS.

Global shared HFS structure overview

In order to describe the global HFS structure when implementing Shared HFS in a Sysplex, we will use a sample Sysplex (WXCF), composed of two MVS images (AMVS and BMVS). After implementing Shared HFS, the HFS structure for AMVS will look like:

- the root Version will be instantiated with the SYSRES volser (&SYSR1 = RES$01). So, the symbolic link /bin will point to the directory /bin of the Version HFS RES$01.

- The symbolic link /tmp will point to /AMVS/tmp, which corresponds to the /tmp directory of the System-specific HFS for AMVS. A specific HFS for /tmp (ABASEPRO.BASEPRHF. OMVS.TMP) is mounted on that directory.

A similar structure will be implemented for the other MVS image BMVS.

IMPLEMENTING SHARED HFS

First, you should install OS/390 Version 2 Release 9 on all the systems that will participate in HFS sharing (these systems should be set up in a Sysplex).

A new chapter in *OS/390 Unix System Services Planning* contains information about how to set up shared HFS in a Sysplex, including how to create the Sysplex root HFS dataset and the system-specific HFS dataset, and how to format an OS/390 Unix Couple Dataset (CDS).

It also describes how to update BPXPRM*xx*, move file systems, control security, and tune performance in a Sysplex.

Creating the Sysplex root HFS dataset

The Sysplex root is an HFS dataset that is used as the Sysplex-wide root. This HFS dataset, which is very small, must be mounted read/ write. Only one Sysplex root is allowed for all systems participating in shared HFS.

The Sysplex root is created by invoking the BPXISYSR sample job in SYS1.SAMPLIB shown below:

```
//*
//IKJEFT1A  EXEC PGM=IKJEFT1A,PARM='BPXISYS1'
//*
//ROOTSYSP  DD DSNAME=WBASEPRO.BASEPRHF.OMVS.SYSPLEX.ROOT,
//          DISP=(,CATLG),
//          DSNTYPE=HFS,
//          SPACE=(CYL,(1,Ø,1))
//*
//SYSEXEC   DD DSN=SYS1.SAMPLIB,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
//*
```

You would need superuser authority to run this job.

Creating the system-specific HFS datasets

Directories in the system-specific HFS dataset are used as mount points, specifically for /etc, /var, /tmp, and /dev.

It is highly recommended to use a standardized naming convention for this dataset uses the MVS system name (&SYSNAME).

To create the system-specific HFS, run the BPXISYSS sample job in SYS1.SAMPLIB:

```
//*
//IKJEFT1A  EXEC PGM=IKJEFT1A,PARM='BPXISYS2'
//*
//HFSSYSTS  DD DSNAME=WBASEPRO.BASEPRHF.OMVS.AMVS.HFS,
//          DISP=(,CATLG),
//          DSNTYPE=HFS,
//          SPACE=(CYL,(1,Ø,1))
//*
//SYSEXEC   DD DSN=SYS1.SAMPLIB,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
//*
```

BPXISYSS should be run multiple times, with different dataset names, in order to create one system-specific HFS for every MVS image in the Sysplex. The mount point for the system-specific HFS dataset (/AMVS) will be created dynamically during OMVS start-up.

Creating OMVS Couple Datasets

The Couple Dataset (CDS) contains the Sysplex-wide mount table and information about all participating systems, and all mounted file systems in the Sysplex. CDS are used in the following manner:

- The first system that enters the Sysplex with SYSPLEX(YES) initializes an OMVS CDS. The CDS controls shared HFS mounts and will eventually contain information about all systems participating in shared HFS.

- This system processes its BPXPRM*xx* parmlib member, including all its ROOT and MOUNT statement information. It is also the designated owner of the byte range lock manager for the participating group. The MOUNT and ROOT information are logged in the CDS so that other systems that eventually join the participating group can read data about systems that are already using shared HFS.
- Subsequent systems joining the participating group will read what is already logged in the CDS and will perform all mounts. Any new BPXPRM*xx* mounts are processed and logged into the CDS. Systems already in the participating group will then process the new mounts added to the CDS.

To allocate and format a CDS invoke the BPXISCDS sample job in SYS1.SAMPLIB. It specifies the number of mount records that are supported by the CDS. For example:

```
//STEP1Ø   EXEC PGM=IXCL1DSU
//STEPLIB  DD   DSN=SYS1.MIGLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
  /* Begin definition for OMVS couple dataset (1)                  */
     DEFINEDS SYSPLEX(WXCF)           /* Name of the Sysplex in
                                         which the OMVS couple
                                         dataset is to be used.    */
  DSN(SYSW.WXCF.OMVSØ1) VOLSER(SYSWA1) /* The name and
                                         volume for the OMVS
                                         couple dataset. The
                                         utility will allocate a
                                         new dataset by the name
                                         specified on the volume
                                         specified.                */
              MAXSYSTEM(2)            /* Number of systems in the
                                         Sysplex to be supported by
                                         this couple dataset. Default
                                         value is eight.       @Ø1A*/
              CATALOG                 /* Default is not to CATALOG.
                                                               @Ø1C*/
           DATA TYPE(BPXMCDS)         /* The type of data in the
                                         dataset being created is
                                         for OMVS. BPXMCDS is the
                                         TYPE for OMVS.            */
               ITEM NAME(MOUNTS) NUMBER(Ø3Ø) /* Specifies the number of
                                         MOUNTS that can be supported
                                         by OMVS.
```

```
                                    Default =   100
                                    Minimum =     1
                                    Maximum = 50000        @D1C*/
              ITEM NAME(AMTRULES) NUMBER(50) /* Specifies the number
                                    of automount rules that can
                                    be supported by OMVS.
                                    Default =    50
                                    Minimum =    50
                                    Maximum =  1000        @D1A*/

//*
```

When you create this dataset you need to specify the maximum number of HFS mounts and automount rules you want to support under OMVS. This is the total number Sysplex wide.

You should set this value right for your installation. If you set it too high and more than what you really need, the HFS performance may be affected.

You should define two couple datasets – one as primary and the other as secondary. Define these on two different volumes for availability reasons.

Updating COUPLExx parmlib member

Next you should tell XCF the names of the primary and secondary couple datasets you have just defined. This is done by updating the COUPLE*xx* member in your parmlib dataset. The COUPLExx member contains information on all couple datasets used by your Sysplex.

You should update that COUPLE*xx* member to define the CDS to XCF.

```
/*                                                                 */
 COUPLE SYSPLEX(WXCF)
        PCOUPLE(SYSW.WXCF.COUPLE01,SYSWA1)
        ACOUPLE(SYSW.WXCF.COUPLE02,SYSWA2)
/*                                                                 */
/*                                                                 */
DATA  TYPE(CFRM)
      PCOUPLE(SYSW.WXCF.CFRM01,SYSWA1)
      ACOUPLE(SYSW.WXCF.CFRM02,SYSWA2)
/*                                                                 */
DATA  TYPE(WLM)
      PCOUPLE(SYSW.WXCF.WLM01,SYSWA1)
```

```
      ACOUPLE(SYSW.WXCF.WLMØ2,SYSWA2)
/*                                                                  */
DATA  TYPE(LOGR)
      PCOUPLE(SYSW.WXCF.LOGRØ1,SYSWA1)
      ACOUPLE(SYSW.WXCF.LOGRØ2,SYSWA2)
/*                                                                  */
DATA  TYPE(ARM)
      PCOUPLE(SYSW.WXCF.ARMØ1,SYSWA1)
      ACOUPLE(SYSW.WXCF.ARMØ2,SYSWA2)
/*                                                                  */
DATA  TYPE(SFM)
      PCOUPLE(SYSW.WXCF.SFMØ1,SYSWA1)
      ACOUPLE(SYSW.WXCF.SFMØ2,SYSWA2)
/*                                                                  */
/*                                                                  */
DATA  TYPE(BPXMCDS)
/* Á OMVS CDS */
      PCOUPLE(SYSW.WXCF.OMVSØ1,SYSWA1)
      ACOUPLE(SYSW.WXCF.OMVSØ2,SYSWA2)
/*                                                                  */
/* PATH AMVS TO BMVS                                                */
/*                                                                  */
 PATHIN  DEVICE(C49)
 PATHOUT DEVICE(C48)
/*                                                                  */
/*                                                                  */
/* PATH BMVS TO AMVS                                                */
/*                                                                  */
  PATHIN  DEVICE(C18)
  PATHOUT DEVICE(C19)
 /*                                                                 */
 /* PATH VIA COUPLING FACILITY                                      */
 /*                                                                 */
  PATHIN  STRNAME(IXC_PATH1)
  PATHOUT STRNAME(IXC_PATH1)
 /*                                                                 */
  PATHIN  STRNAME(IXC_PATH2)
  PATHOUT STRNAME(IXC_PATH2)
 /*                                                                 */
 /* TRANSPORT CLASS                                                 */
 /*                                                                 */
  CLASSDEF CLASS(TCØ4Ø96Ø) CLASSLEN(4Ø96Ø)
```

Customizing BPXPRM*xx* for Shared HFS

You specify the various parameters to control the file system in the BPXPRMxx parmlib member.

This member also contains information to control the OMVS set up and processing.

New keywords of BPXPRM*xx*

In order to implement shared HFS, there are new keywords available in BPXPRM*xx*. They are:

- SYSPLEX(YES|NO).
- VERSION('xxxx').
- ROOT and MOUNT new keywords: SYSNAME(sysname), AUTOMOVE|NOAUTOMOVE.
- SYSPLEX(YES) indicates whether a system is to be initialized in a Sysplex environment or operate in local mode. The first system entering the Sysplex with SYSPLEX(yes) initializes a Couple Dataset (CDS), which controls shared HFS mounts. The value of this parameter cannot be changed dynamically.
- VERSION('xxxx') indicates the release or version of root HFS. This statement will dynamically create a mount point at the time of IPL to mount the Version HFS file. The version HFS is the IBM-supplied root HFS dataset. You should specify a version parameter that identifies your OS/390 system release level. The system residence volser might be an appropriate name you can use. Different MVS systems in the Sysplex can specify different VERSION parameters in their BPXPRMxx member to allow different releases or service levels of root HFS.
- SYSNAME(sysname) declares the system that is the 'owner' of the file system (by default, it is the system where the mount is issued).
- AUTOMOVE|NOAUTOMOVE parameters indicate whether, if the specified root file system owner goes down, the root file system can be automatically moved to another system, which then becomes the owner of that root.

Sample BPXPRM00 member

You can use a common BPXPRM*xx* member for all the systems in the Sysplex:

```
FILESYSTYPE TYPE(HFS)
            ENTRYPOINT(GFUAINIT)
```

```
            PARM(" ")

/****************************/
/* shared HFS               */
/****************************/

SYSPLEX(YES)
VERSION("&SYSR1.")

/************************/
/* Sysplex root HFS    */
/************************/

ROOT  FILESYSTEM("WBASEPRO.BASEPRHF.OMVS.SYSPLEX.ROOT")
      TYPE(HFS)
      MODE(RDWR)

/****************************/
/* version HFS              */
/****************************/

MOUNT FILESYSTEM("SYS1.ROOT")
      TYPE(HFS)
      MODE(READ)
      MOUNTPOINT("/$VERSION")

/************************/
/* system-specific HFS */
/************************/

MOUNT FILESYSTEM("WBASEPRO.BASEPRHF.OMVS.&SYSNAME..HFS")
      TYPE(HFS)
      MODE(RDWR)
      NOAUTOMOVE
      MOUNTPOINT("/&SYSNAME.")

MOUNT FILESYSTEM("&SYSCLONE.BASEPRO.BASEPRHF.OMVS.TMP")
      TYPE(HFS)
      MODE(RDWR)
      NOAUTOMOVE
      MOUNTPOINT("/&SYSNAME./tmp")

MOUNT FILESYSTEM("&SYSCLONE.BASEPRO.BASEPRHF.OMVS.ETC")
      TYPE(HFS)
      MODE(RDWR)
      NOAUTOMOVE
      MOUNTPOINT("/&SYSNAME./etc")

MOUNT FILESYSTEM("&SYSCLONE.BASEPRO.BASEPRHF.OMVS.DEV")
      TYPE(HFS)
```

```
MODE(RDWR)
NOAUTOMOVE
MOUNTPOINT("/&SYSNAME./dev")
```

You should notice the NOAUTOMOVE options on the MOUNT statements. These tell the mount table not to move this HFS dataset to another system when the MVS 'owner' system goes down (default, if not specified, is AUTOMOVE).

IPL the systems of the Sysplex

This is to activate the COUPLE*xx* and BPXPRM*xx* changes. Now you are ready to share HFS datasets across the Sysplex.

Automount policies

The automount facility gives you the flexibility to mount files as and when they are required rather than keeping them mounted all the time. The mount point directories are created internally as they are required. Later, when the file system is no longer in use, the mount point directories are deleted. You specify the details on how to and when to mount and unmount the file in a policy file.

Before Release 9, your automount policy most likely resided in the /etc/auto.master and /etc/u.map files. For those using shared HFS, each participating system has a separate /etc file system.

In order for the automount policy to be consistent across participating systems, the same copy of the automount policy must exist in every system's /etc/auto.master and /etc/u.map files.

For example both AMVS and BMVS have the following configuration files:

```
/u                      /etc/auto_u.map
```

and

```
name            *
type            HFS
filesystem      WBASEPRO.BASEPRHF.OMVS.U.<uc_name>
mode            rdwr
duration        60
delay           60
```

When the automount daemon initializes on AMVS, it will read its local /etc/auto.master file to identify what directories to manage; in this case, it is /u.

Next, the automount daemon will use the policy specified in the local /etc/u.map file to mount file systems with the specified naming convention under /u. The automount daemon on BMVS will perform similar actions.

Because all mounted file systems are available to all participating systems in the Sysplex, your automount policy must be consistent. This is true for the file system name specified in /etc/u.map and the values for other parameters in /etc/u.map and /etc/auto.master.

Different behaviour

The biggest change generated by the introduction of Shared HFS is the fallout from changing directories to symbolic links.

Things like the 'ls' command will now require you to specify:

```
ls /etc/
```

You should notice the trailing slash, which is now mandatory.

Displaying enhanced mount information

Changes were made to display commands to support Shared HFS. For example, D OMVS,F:

```
D OMVS,F.
D OMVS,FILE
BPXO045I 15.35.05 DISPLAY OMVS 256
OMVS     000F ACTIVE          OMVS=(00)
TYPENAME   DEVICE ————STATUS———— MODE
AUTOMNT         7 ACTIVE                      RDWR
  NAME=*AMD/u
  PATH=/u
  OWNER=AMVS     AUTOMOVE=Y CLIENT=N
HFS            11 ACTIVE                      RDWR
  NAME=BBASEPRO.BASEPRHF.OMVS.DEV
  PATH=/BMVS/dev
  OWNER=BMVS     AUTOMOVE=N CLIENT=Y
HFS            10 ACTIVE                      RDWR
  NAME=BBASEPRO.BASEPRHF.OMVS.ETC
  PATH=/BMVS/etc
```

```
  OWNER=BMVS     AUTOMOVE=N CLIENT=Y
HFS             9 ACTIVE                        RDWR
  NAME=BBASEPRO.BASEPRHF.OMVS.TMP
  PATH=/BMVS/tmp
  OWNER=BMVS     AUTOMOVE=N CLIENT=Y
HFS             8 ACTIVE                        RDWR
  NAME=WBASEPRO.BASEPRHF.OMVS.BMVS.HFS
  PATH=/BMVS
  OWNER=BMVS     AUTOMOVE=N CLIENT=Y
HFS             6 ACTIVE                        RDWR
  NAME=ABASEPRO.BASEPRHF.OMVS.DEV
  PATH=/AMVS/dev
  OWNER=AMVS     AUTOMOVE=N CLIENT=N
HFS             5 ACTIVE                        RDWR
  NAME=ABASEPRO.BASEPRHF.OMVS.ETC
  PATH=/AMVS/etc
  OWNER=AMVS     AUTOMOVE=N CLIENT=N
HFS             4 ACTIVE                        RDWR
  NAME=ABASEPRO.BASEPRHF.OMVS.TMP
  PATH=/AMVS/tmp
  OWNER=AMVS     AUTOMOVE=N CLIENT=N
HFS             3 ACTIVE                        RDWR
  NAME=WBASEPRO.BASEPRHF.OMVS.AMVS.HFS
  PATH=/AMVS
  OWNER=AMVS     AUTOMOVE=N CLIENT=N
HFS             2 ACTIVE                        READ
  NAME=SYS1.ROOT
  PATH=/RES$Ø1
  OWNER=AMVS     AUTOMOVE=Y CLIENT=N
HFS             1 ACTIVE                        RDWR
  NAME=WBASEPRO.BASEPRHF.OMVS.SYSPLEX.ROOT
  PATH=/
  OWNER=AMVS     AUTOMOVE=Y CLIENT=N
```

For D OMVS,O, the command has been updated to show new parmlib SYSPLEX and VERSION keywords.

```
D OMVS,OPTIONS
BPXOØ43I 17.54.57 DISPLAY OMVS 282
OMVS     ØØØF ACTIVE          OMVS=(ØØ)
OS/39Ø Unix CURRENT CONFIGURATION SETTINGS:
MAXPROCSYS      =        2ØØ    MAXPROCUSER     =         25
MAXFILEPROC     =         64    MAXFILESIZE     = NOLIMIT
MAXCPUTIME      =       1ØØØ    MAXUIDS         =         5Ø
MAXRTYS         =        256    MAXPTYS         =        256
MAXMMAPAREA     =       4Ø96    MAXASSIZE       =   41943Ø4Ø
MAXTHREADS      =        2ØØ    MAXTHREADTASKS  =         5Ø
MAXCORESIZE     =    4194Ø4    MAXSHAREPAGES   =     131Ø72
IPCMSGQBYTES    =     262144    IPCMSGQMNUM     =      1ØØØØ
IPCMSGNIDS      =        5ØØ    IPCSEMNIDS      =        5ØØ
```

```
IPCSEMNOPS      =         25    IPCSEMNSEMS     =         25
IPCSHMMPAGES    =        256    IPCSHMNIDS      =        500
IPCSHMNSEGS     =         10    IPCSHMSPAGES    =     262144
SUPERUSER       = BPXROOT       FORKCOPY        = COW
STEPLIBLIST     =
USERIDALIASTABLE=
PRIORITYPG VALUES: NONE
PRIORITYGOAL VALUES: NONE
MAXQUEUEDSIGS   =       1000    SHRLIBRGNSIZE   =   67108864
SHRLIBMAXPAGES  =       4096    VERSION         = RES$01
SYSCALL COUNTS  = NO            TTYGROUP        = TTY
SYSPLEX         = YES           BRLM SERVER     = AMVS
```

PERFORMANCE ISSUES

There is a performance cost associated with the introduction of the shared HFS function. Specifically, a degradation in file I/O and mount times can be noticed.

XCF overhead

There is no I/O performance reduction for an HFS successfully mounted R/O on all systems.

However, the intersystem communication required for shared HFS does affect response time and throughput on R/W file systems being shared in a Sysplex.

File I/O to a shared file system from a client that does not own the mount has additional pathlength plus latency involved in the XCF messaging function. For this reason, it is recommended that files be mounted on the system where they are most heavily used.

Increased XCF message traffic to support shared HFS can contribute to system degradation if not monitored and controlled.

Workloads that use large file buffer sizes will give you an increased number of large messages; workloads with small file buffer sizes will you give you an increased number of small messages.

To control XCF message traffic, closely monitor the number and size of message buffers and the number of message paths within the Sysplex.

It is likely that you will need to define additional XCF paths and increase the number of XCF message buffers above the minimum default.

Impacts in HFS mount times

You should be aware that because of I/O operations to the mount table CDS, every mount request requires additional system overhead.

Mount time increases as a function of the following three parameters – the number of mounts, the number of members in the Sysplex, and the size of your CDS.

Number of mounts

When a system joins the Sysplex, information concerning its mounts must be written into the CDS. This takes time since mounts are performed serially, one at a time.

Also, every time a system joins the Sysplex, it must first read the CDS and perform the mounts already listed there. Once this is done, the new system can perform its own mounts and write that information to the CDS. As each new mount is performed, any other systems in the Sysplex must then read what was just written into the CDS and perform the same mount (catch-up). Conducting these catch-ups affects the system on which the new mounts are initiated, ie the next mount cannot be written into the CDS until the other systems have read and mounted the last mount recorded in the CDS.

Thus, the more mounts that need to be performed, the more time is expended in the reading of and writing to the CDS.

Number of members in the Sysplex

As pointed out above, when a new system enters a Sysplex, it must perform all the mounts listed in the CDS before it can perform its own mounts. The more systems in the Sysplex, the more mounts will be recorded in the CDS, and thus the more time it will take for this new system to read what is already in the CDS and perform the mounts.

Also, as already discussed, a catch-up function must be performed

when there are multiple systems in the Sysplex. The more systems in the Sysplex, the more systems will be competing to read what is added to the CDS and perform the new mounts listed there. This competition, as discussed above, impedes the mount response time on the system that is writing new mounts to the CDS.

Size of the CDS

When you format your CDS, the number of mounts specified in the JCL determines the size of the CDS.

A value of 1000 gives you a smaller CDS than a value of 5000.

When a system reads the CDS for mount information, it must read the entire CDS. Thus, even if you are only performing 600 mounts, if you specify a mount number of 1000, all members in the Sysplex will read the CDS as if 1000 mounts must be performed. In other words, whatever amount of space the CDS will use to hold information on 1000 mounts will be read by all the systems in the Sysplex.

For example, even if only 600 mounts are listed in the CDS, if the mount number in the JCL is 1000, a new system joining the Sysplex will read the entire CDS, not just the space that contains the information on the 600 mounts. Also, those systems performing catch-up will not only read the new information added, but will read through the entire CDS to get to the new information along with whatever space is left over (in our example, the additional space used up by the 400 mounts defined in the JCL).

For this reason, IBM recommends that you specify an appropriate number of mounts in the JCL; this way you will not end up with a CDS whose size is unnecessarily large.

Keeping track of load module changes

INTRODUCTION

This article is based on *Determining the LMOD date of load modules* in *MVS Update,* March 1999, Issue 150.

There is no user program that never needs to be modified to reflect end-user change requests or to fix a problem. These changes may happen only every few years, or they may be much more frequent – sometimes even every few days. In an MVS environment (and especially a production one), we need to know, for change management purposes, how many times a program, be it Assembler, PL/I, or COBOL, has been changed since it was first link-edited.

I was installing a CICS program (CEMT I PROG command) when I suddenly realized that it was the fifth end-user install request in 24 hours for the same program. So I came up with the following way to keep track of load module changes. Just to give an example of what the utility does: if a program called GUVEN has been changed five times within a given period, this utility enables you to see all the change dates in a dataset associated with that load library, as follows:

```
GUVEN   00005   91212  92355  92365  96003  98200
```

ENVIRONMENT

We use MVS 5.2.2 running under ES/9000 580 mainframe. We also use ISPF Version 4 Release 2, DFSORT Release 13, PL/I Version 2 Release 3, and JES2 5.2.

CUSTOMIZATION

This utility allows you to use as many load libraries to track as you like. Only two load libraries are used to track their members' change dates. These libraries are SDIAGAS.USER.LOAD and SDIAGAS.AGMPV.LOADLIB. In the following list, therefore, *n* must be taken as '2'. If you wish to use more than two load libraries to track, you must reflect this in the utility programs.

The datasets you need in order to get the job done are shown in Figure 1, which also refers to Figures 2-11.

MISCELLANEOUS NOTES

- The notation *n* used at the end of the dataset name means that there are *n* datasets associated for that dataset group. For example, SDID.MVS.LIB.CHANGE*n* represents both SDID.MVS.LIB.CHANGE1 and SDID.MVS.LIB.CHANGE2 in this utility. That is why it is '2' here.
- In the utility, LMOD DATE word is used as the link-edit date for a given load module.
- To keep load module history records, a variable VSAM dataset structure was considered the best for this type of design. Some load modules might have just one change date, while others might have a variable number of change dates (between two and 1001 changes). This explains why a variable VSAM dataset is used in the utility; if it was not, disk space would be wasted.

  If the maximum of 1001 changes seems to be too high for your organization, you can decrease it to by customizing the VSAM define jobs and PL/I programs.

HOW IT WORKS

This utility can be used for libraries from a variety of applications. First, you need to determine the load libraries of importance to you. Depending on the number you decide on, some minor changes have to be made to the JCL and programs, such as repeating some job steps for each additional load libraries.

Once the theoretical phase of the project is completed, the rest is quite easy. The jobs to be run are listed below:

- First, run ALLOC1 and ALLOC2. These two jobs allocate the necessary VSAM and sequential datasets.
- Second, NULLIFY is a separate job that helps you redefine sequential datasets without having to delete them.

- Then, use the EXTRACT job to get the LMOD dates of load libraries.
- Finally, run INITHIST. This builds the VSAM history datasets. This is the last step for the setting-up process, after which we have all the datasets the utility requires. All VSAM history datasets are then set at their initial status – that is, all members have a change count of 1.

After set-up, you simply need to run the EXTRACT, HJOB1, and HJOB2 daily. These three jobs detect changes and reflect them into the history datasets. Note that, because of the way in which the utility is designed, these jobs must be run the day after set-up. At our site, they run automatically at the end of the day.

When the utility jobs are run late at night they not only make updates but also insertions into the history datasets. As the utility encounters a newly-created load module, it inserts a record into the history dataset concerning the new load module saying 'count = 1'.

The following PL/I programs are used in this utility:

- LMODPLI in Job EXTRACT
- LMODPLI2 in Job INITHIST
- LMODPLI3 in Job HJOB2
- LMODPLI4 in Job HJOB1.

AUTOMATING THE UTILITY JOBS

We use CA-OPERA as a scheduler product, but any other automation tool, such as AF/Operator or CA-7, could also be used. At our organization, CA-OPERA schedules the utility jobs for 2:00 am; this means that by next morning all the utility datasets are ready for the TSO users.

PRESENTING THE RESULTS

ISPF panels are used to show the user the history and LMOD datasets.

SDINLIB1 is the main panel of the utility. Option 1 enables you to have a look at the LMOD dates of a load library selected from the list. Option 2 shows the history records of a load module library. Option 3 is for browsing the load modules link-edited the day before. By using option 4, you can even get the LMOD dates of any load library you want just by entering its name.

FINAL WORD

The efficient use of this utility can even lead to a badly written program being re-written as error-free as possible. Consider the following scenario: if, while browsing history records, you see a high number of changes for a given production load module, there can be two explanations for this. Either there are too many end-user change requests, or the program is not well-written and has been altered over and over again every time it caused an abend or error.

You do not need to worry if the first explanation is the correct one, and changes have been made to accommodate end-user requests. For example, in a banking-related program, interest rate changes may need to be reflected. But action should be taken to recode the program if the second explanation is correct.

This utility is coded according to the Julian date format, with two digits for the year and 3 for the day, because the AMBLIST utility which I have used uses Julian date. Once Year 2000 support is provided for all IBM program products, VSAM history datasets and programs using them should be re-designed to accommodate the digit increase.

SOURCE

JCL ALLOC1

```
//SDIAGAS1 JOB (SDIAGAS),MSGCLASS=X,CLASS=A,MSGLEVEL=(1,1),
// NOTIFY=SDIAGAS
//*
//*************************************************************
//*
//* Allocate required VSAM HISTORY datasets.
//*
```



```
//* In case there are more than 2 load libraries to track,
//* its related VSAM datasets must be added here.
//*
//*  For example, for the three load libraries:
//*
//*  1 - Sdid.Mvs.Lib.Histv1            must be defined.
//*  2 - Sdid.Mvs.Lib.Histv2
//*  3 - Sdid.Mvs.Lib.Histv3
//*
//************************************************************
//*
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
       DELETE  SDID.MVS.LIB.HISTV1 CLUSTER PURGE
       DELETE  SDID.MVS.LIB.HISTV2 CLUSTER PURGE

       DEFINE   CLUSTER                                    -
                (                                          -
                NAME (SDID.MVS.LIB.HISTV1)                 -
                CYLINDERS(1Ø 1)                            -
                INDEXED                                    -
                KEYS (8 Ø)                                 -
                RECORDSIZE (16Ø 6Ø21)                      -
                SHR (2)                                    -
                NOREUSE                                    -
                SPEED                                      -
                WRITECHECK                                 -
                VOLUMES (SDIDØ1)                           -
                )                                          -
               DATA                                        -
                (                                          -
                NAME (SDID.MVS.LIB.HISTV1.DATA)            -
                CISZ (6144)                                -
                FREESPACE (2Ø 1Ø)                          -
                )                                          -
               INDEX                                       -
                (                                          -
                NAME (SDID.MVS.LIB.HISTV1.INDEX)           -
                )
//*
//STEP2    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
  DEFINE CLUSTER(NAME(SDID.MVS.LIB.HISTV2)                  -
                 MODEL(SDID.MVS.LIB.HISTV1))
//*
```

JCL ALLOC2

```
//SDIAGAS2 JOB (SDIAGAS),MSGCLASS=X,CLASS=A,MSGLEVEL=(1,1),
// NOTIFY=SDIAGAS
```

```
//*
//*************************************************************
//*
//* Allocate  all required sequential datasets.
//*
//* In case there are more than 2 load libraries to track,
//* its related  sequential datasets must be added here.
//*
//*  For example, for the third additional load library:
//*
//*  1 - Sdid.Mvs.Lib.Change3
//*  2 - Sdid.Mvs.Lib2.Change3
//*  3 - Sdid.Mvs.Lib.Hists3         must be allocated.
//*
//*************************************************************
//*
//ALLOC1       PROC BIR=
//STEP         EXEC PGM=IEFBR14
//SYSPRINT     DD   SYSOUT=*
//ALLOC1       DD   DISP=(,CATLG),DSN=&BIR,
// SPACE=(TRK,(3,1),RLSE),VOL=SER=SDIDØ1,UNIT=339Ø,
// DCB=(DSORG=PS,RECFM=FB,LRECL=79,BLKSIZE=Ø)
//              PEND
//*
//ALLOC2       PROC IKI=
//STEP         EXEC PGM=IEFBR14
//SYSPRINT     DD   SYSOUT=*
//ALLOC1       DD   DISP=(,CATLG),DSN=&IKI,
// SPACE=(TRK,(3,1),RLSE),VOL=SER=SDIDØ1,UNIT=339Ø,
// DCB=(DSORG=PS,RECFM=VB,LRECL=6Ø27,BLKSIZE=Ø)
//              PEND
//*
//CAGIR1       EXEC ALLOC1,BIR='SDID.MVS.LIB.CHANGE1'
//CAGIR2       EXEC ALLOC1,BIR='SDID.MVS.LIB.CHANGE2'
//CAGIR3       EXEC ALLOC1,BIR='SDID.MVS.LIB.CHANGEX'
//CAGIR4       EXEC ALLOC1,BIR='SDID.MVS.LIB2.CHANGE1'
//CAGIR5       EXEC ALLOC1,BIR='SDID.MVS.LIB2.CHANGE2'
//CAGIR5       EXEC ALLOC1,BIR='SDID.MVS.LIB.SUM'
//*
//CAGIR6       EXEC ALLOC2,IKI='SDID.MVS.LIB.HISTS1'
//CAGIR7       EXEC ALLOC2,IKI='SDID.MVS.LIB.HISTS2'
//*
```

JCL EXTRACT

```
//SDIAGAS3 JOB (SDIAGAS),MSGCLASS=X,CLASS=A,MSGLEVEL=(1,1),
// NOTIFY=SDIAGAS
//*
//*********************************************************************
//*
//* Extract the LMOD dates of members for load libraries and write
//* them into the LMOD datasets.
```

```
//*
//* Sdid.Mvs.Lib.Change1  holds the LMOD DATES of Sdiagas.User.Load
//* Sdid.Mvs.Lib.Change2  holds the LMOD DATES of Sdiagas.Agmpv.Loadlib
//*
//* In case there are more than 2 load libraries to track,
//* this job must be updated. That is, VILLAR3 step must be
//* coded for the third load library.
//*
//*********************************************************************
//*
//VILLAR1   EXEC LMODPROC,OUT='SDID.MVS.LIB.CHANGE1',
// IN='SDIAGAS.USER.LOAD'
//*
//VILLAR2   EXEC LMODPROC,OUT='SDID.MVS.LIB.CHANGE2',
// IN='SDIAGAS.AGMPV.LOADLIB'

//*********************************************************************
//*
//* Sort Sdid.Mvs.Lib.Change1 along with Sdid.Mvs.Lib.Change2 into
//* the MERGE dataset Sdid.Mvs.Lib.ChangeX by 'Julian date.'
//*
//* This dataset will be input to the Clist SDICLIBV called at
//* PERNAS4.
//*
//*********************************************************************
//*
//PERNAS1   EXEC PGM=SORT
//SYSOUT    DD   SYSOUT=*
//SORTIN    DD   DISP=SHR,DSN=SDID.MVS.LIB.CHANGE1
//          DD   DISP=SHR,DSN=SDID.MVS.LIB.CHANGE2
//SORTOUT   DD   DISP=SHR,DSN=SDID.MVS.LIB.CHANGEX
//SORTWKØ1  DD   UNIT=SYSDA,SPACE=(CYL,2Ø)
//SORTWKØ2  DD   UNIT=SYSDA,SPACE=(CYL,2Ø)
//SYSIN     DD   DISP=SHR,DSN=SDID.MVS.LIB.DATA(CTRL3)
//*
//*********************************************************************
//*
//* Sort Sdid.Mvs.Lib.ChangeN into Sdid.Mvs.Lib2.ChangeN by the field
//* 'member-name'.  This dataset will be input to the PL/I program
//* LMODPLI2 later to build the VSAM HISTORY datasets. ( N = 1, 2)
//*
//*********************************************************************
//*
//PERNAS2      EXEC PGM=SORT
//SYSOUT       DD   SYSOUT=*
//SORTIN       DD   DISP=SHR,DSN=SDID.MVS.LIB.CHANGE1
//SORTOUT      DD   DISP=SHR,DSN=SDID.MVS.LIB2.CHANGE1
//SORTWKØ1     DD   UNIT=SYSDA,SPACE=(CYL,2Ø)
//SORTWKØ2     DD   UNIT=SYSDA,SPACE=(CYL,2Ø)
//SYSIN        DD   DISP=SHR,DSN=SDID.MVS.LIB.DATA(CTRL4)
```

```
//*
//PERNAS3      EXEC PGM=SORT
//SYSOUT       DD   SYSOUT=*
//SORTIN       DD   DISP=SHR,DSN=SDID.MVS.LIB.CHANGE2
//SORTOUT      DD   DISP=SHR,DSN=SDID.MVS.LIB2.CHANGE2
//SORTWKØ1     DD   UNIT=SYSDA,SPACE=(CYL,2Ø)
//SORTWKØ2     DD   UNIT=SYSDA,SPACE=(CYL,2Ø)
//SYSIN        DD   DISP=SHR,DSN=SDID.MVS.LIB.DATA(CTRL4)
//*
//************************************************************
//*
//* By using the Clist SDICLIB2, extract the last link edited
//* (changed) modules' names  from the Sdid.Mvs.Lib.ChangeX
//* and write them into TRANSACTION dataset Sdid.Mvs.Lib.Sum.
//*
//************************************************************
//*
//PERNAS4   EXEC PGM=IKJEFTØ1
//SYSTSPRT  DD   SYSOUT=*
//SYSTSIN   DD   *
EX 'SYSP.ISPF.CLIB(SDICLIB2)'
```

JCL INITHIST

```
//SDIAGAS4 JOB (SDIAGAS),MSGCLASS=X,CLASS=A,MSGLEVEL=(1,1),
// NOTIFY=SDIAGAS
//*
//*****************************************************************
//*
//* Build the VSAM HISTORY datasets  Sdid.Mvs.Lib.HistvN
//* from the LMOD datasets  Sdid.Mvs.Lib2.ChangeN. (N = 1, 2)
//*
//* PS: this job must be run only once to set up VSAM datasets.
//*
//* In case there are more than 2 load libraries to track, this
//* job must be updated. That is, CAGIR3 step must be inserted
//* for the third library.
//*
//*
//*  Procedure parameters:
//*                    1- BIR  = LMOD dataset name .
//*                    2- IKI  = Vsam dataset name.
//*
//*****************************************************************
//*
//HISTORY     PROC IKI=,BIR=
//*
//STEP         EXEC PGM=LMODPLI2
//STEPLIB      DD   DISP=SHR,DSN=SDIAGAS.USER.LOAD
```

```
//SYSPRINT     DD   SYSOUT=*
//GO.INFILE    DD   DISP=SHR,DSN=&BIR
//GO.OUTFILE   DD   DISP=SHR,DSN=&IKI
//             PEND
//*
//*
//CAGIR1       EXEC HISTORY,IKI='SDID.MVS.LIB.HISTV1',
// BIR='SDID.MVS.LIB2.CHANGE1'
//*
//CAGIR2       EXEC HISTORY,IKI='SDID.MVS.LIB.HISTV2',
// BIR='SDID.MVS.LIB2.CHANGE2'
//*
```

JCL HJOB1

```
//SDIAGAS5 JOB (SDIAGAS),MSGCLASS=X,TIME=144Ø,CLASS=P,
// NOTIFY=SDIAGAS
//*
//*******************************************************************
//*
//* Update the Sdid.Mvs.Lib.HistvN VSAM HISTORY datasets by
//* using the TRANSACTION dataset Sdid.Mvs.Lib.Sum. (N = 1, 2)
//*
//* The associated VSAM HISTORY datasets are updated for
//* the new load modules or the load modules changed lately.
//*
//*******************************************************************
//*
//STEP1        EXEC PGM=SORT
//*
//*******************************************************************
//*
//*  Sort the TRANSACTION dataset by 'member-name' field.
//*
//*******************************************************************
//*
//SYSOUT       DD   SYSOUT=*
//SORTIN       DD   DISP=SHR,DSN=SDID.MVS.LIB.SUM
//SORTOUT      DD   DISP=SHR,DSN=SDID.MVS.LIB.SUM
//SORTWKØ1     DD   UNIT=SYSDA,SPACE=(CYL,2Ø)
//SORTWKØ2     DD   UNIT=SYSDA,SPACE=(CYL,2Ø)
//SYSIN        DD   DISP=SHR,DSN=SDID.MVS.LIB.DATA(CTRL4)
//*
//*
//STEP2        EXEC PGM=LMODPLI4
//STEPLIB      DD   DISP=SHR,DSN=SDIAGAS.USER.LOAD
//SYSPRINT     DD   SYSOUT=*
//GO.INFILE    DD   DISP=SHR,DSN=SDID.MVS.LIB.SUM
//GO.FIL1      DD   DISP=OLD,DSN=SDID.MVS.LIB.HISTV1
//GO.FIL2      DD   DISP=OLD,DSN=SDID.MVS.LIB.HISTV2
```

```
//*
```

JCL HJOB2

```
//SDIAGAS6 JOB (SDIAGAS),MSGCLASS=X,TIME=144Ø,CLASS=P,
// NOTIFY=SDIAGAS
//*
//*************************************************************
//*
//* Build the HISTORY sequential datasets from HISTORY VSAM
//* datasets.
//*
//*  Procedure parameters:
//*                            A) ILK  = HISTORY (VSAM)  dataset
//*                            B) SON  = HISTORY (Seql.) dataset
//*
//*
//* In case there are more than 2 load libraries to track,
//* this job must be updated. That is, CAGIR3 and GUVEN3
//* steps must be coded for the third load library.
//*
//*
//*************************************************************
//*
//HISTSEQ      PROC ILK=,SON=
//STEP         EXEC PGM=LMODPLI3
//STEPLIB      DD   DISP=SHR,DSN=SDIAGAS.USER.LOAD
//SYSPRINT     DD   SYSOUT=*
//GO.INFILE    DD   DISP=SHR,DSN=&ILK
//GO.KUT       DD   DISP=SHR,DSN=&SON
//             PEND
//*
//CAGIR1       EXEC HISTSEQ,ILK='SDID.MVS.LIB.HISTV1',
// SON='SDID.MVS.LIB.HISTS1'
//*
//CAGIR2       EXEC HISTSEQ,ILK='SDID.MVS.LIB.HISTV2',
// SON='SDID.MVS.LIB.HISTS2'
//*
//*****************************************************************
//*
//* Sort the HISTORY sequential datasets Sdid.Mvs.Lib.HistsN
//* by 'Count' field.
//*
//*****************************************************************
//*
//SORT       PROC SRTFL=
//STEP       EXEC PGM=SORT
//SYSOUT     DD   SYSOUT=*
//SORTIN     DD   DISP=SHR,DSN=&SRTFL
//SORTOUT    DD   DISP=SHR,DSN=&SRTFL
```

```
//SORTWKØ1   DD   UNIT=SYSDA,SPACE=(CYL,2Ø)
//SORTWKØ2   DD   UNIT=SYSDA,SPACE=(CYL,2Ø)
//SYSIN      DD   DISP=SHR,DSN=SDID.MVS.LIB.DATA(CTRL5)
//           PEND
//*
//GUVEN1     EXEC SORT,SRTFL='SDID.MVS.LIB.HISTS1'
//GUVEN2     EXEC SORT,SRTFL='SDID.MVS.LIB.HISTS2'
//*
```

PL/I PROGRAM LMODPLI

```
//SDIAGASW JOB  (SDIAGAS),MSGCLASS=X,MSGLEVEL=(1,1),CLASS=A,
// NOTIFY=SDIAGAS
//*
//GUVEN     EXEC PLIXCL
//SYSPRINT  DD   SYSOUT=*
//PLI.SYSIN DD   *
 SERKAN:PROC(PARM) OPTIONS(MAIN NOEXECOPS)                              ;

 /*********************************************************************/
 /*                                                                   */
 /* Program:  Lmodpli                                                 */
 /* Function: This PL/I program formats the output of the Amblist     */
 /*           utility and builds the LMOD dataset.                    */
 /*                                                                   */
 /*********************************************************************/

 DCL IN  FILE RECORD SEQL INPUT                                       ,
     OUT FILE RECORD SEQL OUTPUT                                      ,
     (SUBSTR,INDEX,ONCODE) BUILTIN                                    ,
     A BIT(1) INIT('1'B)                                              ,
     SAHA CHAR(133)                                                   ,
     (MEMB_NAME,MEMB_DATE_GUN,MEMB_DATE_YIL) CHAR(2Ø) VAR             ,
     PARM CHAR(44) VAR                                                ,
     1 ALAN                                                           ,
       2   FILLER1       CHAR(1) INIT('')                             ,
       2   MEMB_NAME     CHAR(8)                                      ,
       2   FILLER2       CHAR(1) INIT('')                             ,
       2   MEMB_DATE_YIL PIC'(2)X'                                    ,
       2   MEMB_DATE_GUN PIC'(3)X'                                    ,
       2   FILLER3       CHAR(2) INIT('')                             ,
       2   TARIH         CHAR(18) INIT('')                            ,
       2   DSN           CHAR(44)                                     ;

 ON ENDFILE(IN) A='Ø'B                                                ;

 /*********************************************************************/
 /*                                                                   */
 /* Open the files.                                                   */
 /*                                                                   */
```

```
/*****************************************************************/

OPEN FILE(IN)                                                     ;
OPEN FILE(OUT)                                                    ;
READ FILE(IN) INTO(SAHA)                                          ;
DO WHILE(A='1'B)                                                  ;
    IF (INDEX(SAHA,'MEMBER NAME') ¬=Ø) | (INDEX(SAHA,'OF YEAR') ¬=Ø)
       THEN
        DO                                                        ;
           ALAN.MEMB_NAME=SUBSTR(SAHA,3Ø,8)                       ;
           READ FILE(IN) INTO(SAHA)                               ;
           ALAN.MEMB_DATE_GUN=SUBSTR(SAHA,1Ø2,3)                  ;
           ALAN.MEMB_DATE_YIL=SUBSTR(SAHA,114,2)                  ;
           ALAN.DSN=PARM                                          ;
           CALL ALT(ALAN.MEMB_DATE_YIL,ALAN.MEMB_DATE_GUN,ALAN.TARIH);
           WRITE FILE(OUT) FROM(ALAN)                             ;
        END                                                       ;
    READ FILE(IN) INTO(SAHA)                                      ;
END                                                               ;
/*****************************************************************/
/*                                                               */
/* Close the files.                                              */
/*                                                               */
/*****************************************************************/

CLOSE FILE(IN)                                                    ;
CLOSE FILE(OUT)                                                   ;

ALT:PROC(M_DATE_YIL,M_DATE_GUN,YAZI)                              ;

/*****************************************************************/
/*                                                               */
/*  This sub_program converts the Julian date to the normal date.*/
/*  eg Julian date 98Ø67 will be converted to 'Ø8.MARCH.1998'    */
/*                                                               */
/*  NOTE: This conversion doesn't apply the rules for the leap   */
/*        years such as  1ØØ year and 4ØØ year rule. It just     */
/*        applies 'divisible by 4 ' rule. The oldest MVS load    */
/*        module is NO MORE THAN 2Ø YEARS. SO THERE IS NO PROBLEM*/
/*        AT ALL in terms of the leap year.                      */
/*                                                               */
/*****************************************************************/

DCL JDATE PIC'99999'                                              ,
  M_DATE_YIL PIC'XX'                                              ,
  M_DATE_GUN PIC'XXX'                                             ,
  CONV CHAR(5)                                                    ,
  YAZI CHAR(18)                                                   ,
  YEAR PIC'9999'                                                  ,
  DAY PIC'99'                                                     ,
```

```
   MONTH CHAR(9)                                                          ,
   (SUM1,I,SUM2) FIXED BIN(31)                                            ,
   YIL(12) FIXED BIN(31) INIT(31,28,31,3Ø,31,3Ø,31,31,3Ø,31,3Ø,31)        ,
   AY(12) CHAR(9) VAR INIT('JANUARY','FEBRUARY','MARCH','APRIL','MAY'     ,
   'JUNE','JULY','AUGUST','SEPTEMBER','OCTOBER','NOVEMBER','DECEMBER'),
   MOD BUILTIN                                                            ;

 CONV = M_DATE_YIL || M_DATE_GUN                                          ;
 SUM1,SUM2 = Ø                                                            ;
 JDATE = CONV                                                             ;
 YEAR = JDATE/1ØØØ                                                        ;
 JDATE = JDATE - 1ØØØ * YEAR /*Number of days portion of Julian date*/;
 YEAR = 19ØØ + YEAR /* Add 19ØØ to the year portion of Julian date  */;

 /*****************************************************************/
 /*                                                               */
 /* Every year divisible by 4 is a leap year. (MOD builtin function)*/
 /* If it's a leap year then change the array to reflect leap year. */
 /*                                                               */
 /*****************************************************************/

 IF MOD(YEAR,4) =Ø THEN YIL(2) = 29                                       ;

 DO I=1 TO 12                                                             ;
      SUM1 = SUM1 + YIL(I)                                                ;
      IF SUM1     >= JDATE THEN LEAVE                                     ;
      SUM2         = SUM1                                                 ;
 END                                                                      ;

 MONTH = AY(I)                                                            ;

 IF SUM1 = JDATE THEN DAY = YIL(I)                                        ;
                 ELSE DAY = JDATE - SUM2                                  ;

 /*****************************************************************/
 /*                                                               */
 /* At last, the date in normal format is returned to the main      */
 /* program.                                                      */
 /*                                                               */
 /*****************************************************************/

 SUBSTR(YAZI,1,2)  = DAY                                                  ;
 SUBSTR(YAZI,3,1)  = '.'                                                  ;
 SUBSTR(YAZI,4,9)  = MONTH                                                ;
 SUBSTR(YAZI,13,1) = '.'                                                  ;
 SUBSTR(YAZI,14,4) = YEAR                                                 ;
 END ALT                                                                  ;
 END SERKAN                                                               ;
//*
//LKED.SYSLMOD  DD DISP=SHR,DSN=SDIAGAS.USER.LOAD(LMODPLI)
```

```
//*
```

PL/I PROGRAM LMODPLI2

```
//SDIAGASX     JOB  (SDIAGAS),MSGCLASS=X,MSGLEVEL=(1,1),CLASS=A,
// NOTIFY=SDIAGAS
//*
//KONCI        EXEC PLIXCL,PARM.PLI='GOSTMT'
//SYSPRINT     DD   SYSOUT=*
//PLI.SYSIN    DD   *
 SERVET:PROC OPTIONS(MAIN)                                              ;

 /*************************************************************
 *
 * Program:  Lmodpli2
 * Function: Build the VSAM HISTORY datasets Sdid.Mvs.Lib.HistvN
 *           from the LMOD datasets Sdid.Mvs.Lib2.ChangeN (N =1, 2)
 *
 *
 * RECORD FORMAT for VSAM HISTORY datasets:
 *
 * Columns   Length   Description
 * =======   ======   ========================================
 *
 *  1 -  8     8     : Member name (Load module name)
 *  9 -  9     1     : Blank
 * 1Ø - 13     4     : Change count for that member
 * 14 - 14     1     : Blank
 * 15 - 19     5     : 1. change date of that member (Julian date)
 * 2Ø - 2Ø     1     : Blank
 * 21 - 25     5     : 2. change date of that member (If any)
 * 26 - 26     1     : Blank
 * 27 - 31     5     : 3. change date of that member (If any)
 * 32 - 32     1     : Blank
 * .    .              ..........
 * .    .              ..........
 * .    .              ..........
 *
 * VSAM HISTORY datasets are designed as a variable VSAM.
 * It can have as many as 1Ø1 change dates for a given load
 * module. Then;
 *
* Maximum record length = 19 + 6*1ØØØ = 6Ø19 bytes
 *(This value has been coded in ALLOC1 Allocation job.)
 *
 *
 * Minimum record length = 19 bytes
 * (In this case, one member has just one change count and one change
 *  date.
 *
```

```
*******************************************************************/

 DCL ONCODE BUILTIN,A BIT(1) INIT('1'B)                              ,
     OUTFILE FILE RECORD SEQL OUTPUT KEYED ENV(V VSAM)               ,
     INFILE  FILE RECORD                                            ,
       1 INREC                                                      ,
         2   FILLER1       CHAR(1) INIT('')                        ,
         2   MEMBER        CHAR(8)                                 ,
         2   FILLER2       CHAR(1) INIT('')                        ,
         2   MEMB_DATE     PIC'(5)X'                               ,
         2   FILLER3       CHAR(2) INIT('')                        ,
         2   TARIH         CHAR(18) INIT('')                       ,
         2   DSN           CHAR(44)                                ,

       1 OUTREC                                                     ,
         2 UYE    PIC'(8)X'                                        ,
         2 BOS1   PIC'(1)X'                                        ,
         2 COUNT  PIC'9999'                                        ,
         2 BOS2   PIC'(1)X'                                        ,
         2 DATE   PIC'(5)X'                                        ;

 ON ENDFILE(INFILE) A='Ø'B                                          ;

 ON KEY(OUTFILE)  BEGIN                                             ;
                    PUT LIST('Error',ONCODE,INREC)                  ;
                  END                                               ;

 OPEN FILE(OUTFILE) OUTPUT                                          ;
 OPEN FILE(INFILE)  INPUT                                           ;

 READ FILE(INFILE) INTO(INREC)                                      ;

 DO WHILE(A)                                                        ;
     OUTREC.UYE    = INREC.MEMBER                                   ;
     OUTREC.DATE   = INREC.MEMB_DATE                                ;
     OUTREC.COUNT  = 1                                              ;
     OUTREC.BOS1,OUTREC.BOS2 = ' '                                  ;
     WRITE FILE(OUTFILE) FROM(OUTREC) KEYFROM(UYE)                  ;
     READ  FILE(INFILE)  INTO(INREC)                                ;
 END                                                                ;

 CLOSE FILE(INFILE)                                                 ;
 CLOSE FILE(OUTFILE)                                                ;
 END SERVET                                                         ;
//*
//LKED.SYSLIB  DD   DISP=SHR,DSN=PLI.PLIBASE
//             DD   DISP=SHR,DSN=PLI.SIBMBASE
//LKED.SYSLMOD DD   DISP=SHR,DSN=SDIAGAS.USER.LOAD(LMODPLI2)
//*
```

PL/I PROGRAM LMODPLI3

```
//SDIAGASY     JOB  (SDIAGAS),MSGCLASS=X,MSGLEVEL=(1,1),CLASS=A,
// NOTIFY=SDIAGAS
//*
//PEDRO     EXEC PLIXCL
//SYSPRINT  DD   SYSOUT=*
//PLI.SYSIN DD   *
 CEM:PROC OPTIONS(MAIN)                                                ;

 /*****************************************************************
 *
 * Program:  Lmodpli3
 * Function: Build the HISTORY sequential datasets from the
 *           HISTORY VSAM datasets. These HISTORY sequential
 *           datasets will be used for browsing under ISPF.
 *
 *****************************************************************/

 DCL ONCODE BUILTIN,A BIT(1) INIT('1'B)                                ,
     INFILE FILE RECORD  SEQL  KEYED ENV(VSAM)                         ,
     KUT    FILE RECORD  SEQL  OUTPUT                                  ,

       1 ALAN                                                          ,
         2 MEMBER PIC'(8)X'                                            ,
         2 BOS1   PIC'(1)X'                                            ,
         2 COUNT  PIC'(4)9'                                            ,
         2 BOS2   PIC'(1)X'                                            ,
         2 DATE   PIC'(5)X'                                            ,
         2 KALAN  CHAR(6ØØ2) VARYING                                   ,

     (ALANX,ALANY)  CHAR(6Ø21) VARYING                                 ,

       1 SON CTL                                                       ,
         2 UYE    PIC'(8)X'                                          ,
         2 BOS1   PIC'(1)X' INIT(' ')                                  ,
         2 SAYI   PIC'(4)9'                                            ,
         2 BOS2   PIC'(1)X' INIT(' ')                                  ,
         2 DATE2  PIC'(5)X'                                            ,
         2 KALAN2 CHAR(*) VARYING                                      ;

 ON ENDFILE(INFILE) A='Ø'B                                             ;

 OPEN FILE(INFILE)  INPUT                                              ;
 OPEN FILE(KUT)    OUTPUT                                              ;

 /*****************************************************************/
 /*                                                               */
 /*  Write a heading.                                             */
 /*                                                               */
```

```
 /*********************************************************************/

 ALANY = '$MEMBER $COUNT $DATES'                                       ;
 WRITE FILE(KUT)   FROM(ALANY)                                         ;
 ALANY = '$====== $==== $========'                                     ;
 WRITE FILE(KUT)   FROM(ALANY)                                         ;
 READ FILE(INFILE) INTO(ALANX)                                         ;



 DO WHILE(A)                                                           ;
 IF SUBSTR(ALANX,10,4) = '0001' THEN
               ALANY = SUBSTR(ALANX,1,21) || ' ' || SUBSTR(ALANX,23) ;
          ELSE ALANY = SUBSTR(ALANX,1,19) || ' ' || SUBSTR(ALANX,23) ;
 WRITE FILE(KUT)   FROM(ALANY)                                         ;
 READ FILE(INFILE) INTO(ALANX)                                         ;
 END                                                                   ;

 CLOSE FILE(KUT)                                                       ;
 CLOSE FILE(INFILE)                                                    ;

 FREE SON                                                              ;
 END  CEM                                                              ;
//*
//LKED.SYSLMOD DD DISP=SHR,DSN=SDIAGAS.USER.LOAD(LMODPLI3)
//*
```

PL/I PROGRAM LMODPLI4

```
//SDIAGASZ     JOB  (SDIAGAS),MSGCLASS=X,MSGLEVEL=(1,1),CLASS=A,
// NOTIFY=SDIAGAS
//*
//DANIEL       EXEC PLIXCL
//SYSPRINT     DD  SYSOUT=*
//PLI.SYSIN    DD  *
 PATTY:PROC OPTIONS(MAIN)                                          ;

 /*********************************************************************
  *
  * Program:  Lmodpli4
  * Function: Update the VSAM HISTORY datasets Sdid.Mvs.Lib.HistvN
  *           from the TRANSACTION dataset Sdid.Mvs.Lib.Sum.
  *           (N = 1, 2)
  *
  * USED FILES:
  *
  *   A-) Infile : TRANSACTION dataset
  *
  *  It has load modules changed lately. This may contain different
  *  load module names from different load libraries. So, this program
```

```
 *  reads the TRANSACTION dataset and updates associated HISTORY VSAM
 *  dataset.
 *
 *  B -) Fil1   : HISTORY VSAM datasets.
 *       Fil2
 *       In case there are more than 2 load libraries to track,
 *       related FIL statements must be added here. In addition,
 *       for those load libraries, 'WHEN' PL/I statements must be
 *       coded. (eg:  For the third load library, FIL3 variable
 *       here and GO.FIL3 DD statement to JCL HJOB1 must be added.)
 *
 *  C -) Kut    : This is the variable filename in the PL/I program.
 * It is set to one of the HISTORY VSAM datasets inside the program.
 *
 *
 ********************************************************************/

DCL KUT FILE VARIABLE,A BIT(1) INIT('1'B)                              ,

    FIL1 FILE RECORD  DIRECT KEYED BUFFERED ENV(V VSAM)                ,
    FIL2 FILE RECORD  DIRECT KEYED BUFFERED ENV(V VSAM)                ,

    (SUBSTR,LENGTH) BUILTIN, SYSPRINT SYSTEM OUTPUT,I FIXED BIN(31)    ,

      1 INREC                                                          ,
        2 BOS1   CHAR(1) INIT('')                                      ,
        2 MEMBER CHAR(8)                                               ,
        2 BOS2   CHAR(1) INIT('')                                      ,
        2 MEMB_DATE PIC'(5)X'                                          ,
        2 BOS3   CHAR(2) INIT('')                                      ,
        2 TARIH  CHAR(18)   INIT('')                                   ,
        2 DSN    CHAR(44)                                              ,

    INFILE  FILE RECORD                                                ;

ON ENDFILE(INFILE) A='Ø'B                                              ;
OPEN FILE(INFILE)  INPUT                                               ;

READ FILE(INFILE) INTO(INREC)                                          ;

DO WHILE(A)                                                            ;

 SELECT(INREC.DSN)                                                     ;
   WHEN('SDIAGAS.USER.LOAD')     DO                                    ;
                                   KUT = FIL1                          ;
                                   CALL ATALAY(KUT)                    ;
                                 END                                   ;
   WHEN('SDIAGAS.AGMPV.LOADLIB') DO                                    ;
                                   KUT = FIL2                          ;
                                   CALL ATALAY(KUT)                    ;
```

```
                                  END                                    ;
   OTHERWISE                     PUT EDIT ('Dataset',INREC.DSN,'is not
within the scope.') (COL(1),A(8),X(1),A,X(2),A(24))                      ;
 END                                                                      ;
READ FILE(INFILE) INTO(INREC)                                             ;

END   /* Do while end */                                                  ;

CLOSE FILE(INFILE)                                                        ;

ATALAY:PROC(KUT)                                                          ;

DCL FIL1 FILE RECORD  DIRECT KEYED BUFFERED ENV(V VSAM)                   ,
    FIL2 FILE RECORD  DIRECT KEYED BUFFERED ENV(V VSAM)                   ,
    KUT FILE VARIABLE                                                     ,

    ONCODE BUILTIN,UZ FIXED BIN(31)                                       ,
    EKLE CHAR(6) INIT('      ')                                           ,
    FALAN CHAR(6Ø21) VAR                                                  ,

      1 ALAN                                                              ,
        2 UYE    PIC'(8)X'                                                ,
        2 BOS4   PIC'(1)X'                                                ,
        2 COUNT  PIC'(4)9'                                                ,
        2 BOS5   PIC'(1)X'                                                ,
        2 DATE   PIC'(5)X'                                                ,
        2 KALAN  CHAR(6ØØ2) VARYING                                       ,

      1 ALAN3                                                             ,
        2 UYE3    PIC'(8)X'                                               ,
        2 BOS4    PIC'(1)X' INIT(' ')                                     ,
        2 COUNT3  PIC'(4)9'                                               ,
        2 BOS5    PIC'(1)X' INIT(' ')                                     ,
        2 DATE3   PIC'(5)X'                                               ,

      1 ALANZ CTL                                                         ,
        2 UYEZ   PIC'(8)X'                                                ,
        2 BOS4Z  PIC'(1)X'                                                ,
        2 COUNTZ PIC'(4)9'                                                ,
        2 BOS5Z  PIC'(1)X'                                                ,
        2 DATEZ  PIC'(5)X'                                                ,
        2 KALANZ CHAR(*) VAR                                              ;

ON KEY(KUT)
  BEGIN                                                                   ;
     SELECT(ONCODE)                                                       ;
       WHEN(51) DO                                                        ;

                          /*****************************************/
                          /*                                       */
```

```
                          /* This is a newly-created load module   */
                          /* for that load library dataset. So in  */
                          /* the HISTORY VSAM dataset this should  */
                          /* be written with the 'Count = 1'.      */
                          /*                                       */
                          /*****************************************/
                  PUT EDIT('Member ',MEMBER, 'is a newly created load
 module.It has been added to the VSAM file.')(COL(1),A,A,A)           ;
                  ALAN3.UYE3=INREC.MEMBER                             ;
                  ALAN3.COUNT3=1                                      ;
                  ALAN3.DATE3=INREC.MEMB_DATE                         ;
WRITE FILE(KUT)     FROM(ALAN3) KEYFROM(INREC.MEMBER);
                  GO TO ETIK2                                         ;
                  END                                                 ;
        OTHERWISE PUT SKIP LIST('Any other error......',MEMBER)       ;
      END                                                             ;
   END                                                                ;

 ON RECORD(KUT)
   BEGIN                                                              ;
        SELECT(ONCODE)                                                ;
            WHEN(22) DO                                               ;
                     PUT  SKIP LIST('Record Condition',ONCODE)        ;
                     PUT DATA(ALANZ.UYEZ)                             ;
                     GO TO ETIK2                                      ;
                     END                                              ;
            OTHERWISE PUT SKIP LIST('Any other error......')          ;
        END                                                           ;
   END                                                                ;

 OPEN FILE(KUT) UPDATE                                                ;
 READ FILE(KUT) INTO(ALAN) KEY(INREC.MEMBER)                          ;

 FALAN = ALAN.KALAN                                                   ;
 IF (ALAN.COUNT + 1)  > 1001  THEN
              DO                                                      ;
                 PUT SKIP EDIT('For the load module ',ALAN.UYE,
'the count limit 1001 is exceeded.') (COL(1),A,A,A)                   ;
                 GO TO ETIK2                                          ;
              END                                                     ;

 ALAN.COUNT = ALAN.COUNT + 1                                          ;
 SUBSTR(EKLE,1,1) =' '                                                ;
 SUBSTR(EKLE,2,5) = INREC.MEMB_DATE                                   ;
 FALAN = FALAN || EKLE                                                ;
 ALAN.KALAN=FALAN                                                     ;
 I = LENGTH(FALAN)                                                    ;
                          /*****************************************/
                          /*                                       */
                          /* By using CONTROLLED definition of     */
```

```
                        /* field ALANZ, just the necessary amount */
                        /* of storage is reserved. Here is the    */
                        /* ALLOCATE statement for this purpose.   */
                        /*                                        */
                        /******************************************/


 ALLOCATE 1 ALANZ                                                    ,
            2 UYEZ                                                   ,
            2 BOS4Z                                                  ,
            2 COUNTZ                                                 ,
            2 BOS5Z                                                  ,
            2 DATEZ                                                  ,
            2 KALANZ CHAR(I)                                         ;

 ALANZ = ALAN                                                        ;

 REWRITE FILE(KUT)  FROM(ALANZ) KEY(ALAN.UYE)                        ;
 FREE ALANZ                                                          ;

 EKLE='      ';FALAN='';I=Ø;ALAN=''                                  ;
 ETIK2:CLOSE FILE(KUT)                                               ;

 RETURN                                                              ;
 END ATALAY                                                          ;

 END PATTY                                                           ;
//*
//LKED.SYSLMOD DD  DISP=SHR,DSN=SDIAGAS.USER.LOAD(LMODPLI4)
//*
```

Editor's note: this article will be concluded in the next issue.

Recep Atalay Gul
Systems Programmer (Spain)

As a free service to subscribers and to remove the need to rekey the scripts, code in individual articles can be accessed on our Web site. Subscribers need the user-id printed on the envelope containing their *Update* issue. Once they have registered, any code requested will be automatically e-mailed to them.

Sorting a PDS list

THE PROBLEM

You have listed the contents of a library used by everyone in the shop . . . say, by entering '=3.4' on the COMMAND line and entering 'B' next to one of the library names that comes up. Where is that member you are looking for? You cannot quite remember the name but you do know that you created it.

The 'ID' column contains either the logon ID of the person or the Job ID of the job, that created the file. On the command line, type:

```
SORT ID
```

After you press <Enter> the list will be re-sorted by the values in the ID column. Now, all you have to do is type:

```
L mlnabk (substitute your ID for 'mlnabk')
```

. . . and the list will page forward to your stuff. Now you can wade through a shorter list of member names to find the one you want.

This works on most of the columns in most of the member listings (Edit, Browse, etc). Locate ('L') will always search on the column values that are currently 'sorted'. The next time you enter the list, the sort will default to the standard 'Name' column.

Allan Kalar
Systems Programmer (USA)

If you want to contribute an article to *MVS Update*, a copy of our *Notes for contributors* can be downloaded from the Xephon Web site. The URL is: www.xephon.com/contnote.html.

A Java client/server application on OS/390 – part 2

INTRODUCTION

Last month we considered how Java could be used to implement a simple client/server application on OS/390.

The application can be used as an example to detail Java programming concepts, and will allow a Java client on a PC workstation to interact with a Java server located on OS/390.

This example shows how to implement TCP/IP communication in Java. It also describes how to use the Java Native Interface (JNI) to allow Java programs on OS/390 to communicate with C/C++ and Assembler routines.

MVSCMD.MAKE.

```
MAIN = MVScmd

CC = c89 -c -W c,expo,dll -DNEEDSIEEE754 -DNEEDSLONGLONG
CFLAGS := -I. -I/usr/lpp/java/J1.1/include -I/usr/lpp/java/J1.1/include/mvs
LL = c89 -W l,dll
LFLAG1 := /usr/lpp/java/J1.1/lib/mvs/native_threads/libjava.x
LFLAG2 := /usr/lpp/java/J1.1/lib/mvs/native_threads/libJNIConvert.x

$(MAIN): $(MAIN).o ; $(LL) -o libMVScmdJNI.so $(MAIN).o $(LFLAG1) $(LFLAG2)
$(MAIN).o: $(MAIN).c $(MAIN).h ; $(CC) -o $(MAIN).o $(CFLAGS) $(MAIN).c

$(MAIN).class: $(MAIN).java ; javac $(MAIN).java
$(MAIN).h: $(MAIN).class ; javah -jni -o $*.h $(MAIN)
```

To use this makefile (whose name is not ‘Makefile’), you should enter the following command:

```
make -f MVScmd.make
```

COMASO ASSEMBLER PROGRAM

```
COMASO   CSECT
COMASO   AMODE 31
COMASO   RMODE ANY
```

```
***************************
** SAVE REGISTERS *********
***************************
COMMND   STM R14,R12,12(R13)        SAVE CALLERS REGISTERS
         LR   R12,R15               INITIALIZE BASE REGISTER
         USING COMMND,R12           ESTABLISH ADDRESSABILITY
         LA    R14,SAVEAREA         MY SAVEAREA
         ST    R14,8(R13)           LINK TO CALLERS
         ST    R13,SAVEAREA+4       LINK TO MINE
         LA    R13,SAVEAREA         POINT R13 TO MY SAVE AREA
***************************
** COPY COMMAND ***********
***************************
         L     R2,Ø(R1)             LOAD PARAMETER ADDRESS
         MVC   XCMD,Ø(R2)
*
         L     R3,4(R1)
         L     R3,Ø(R3)
*        LA    R3,5
         STH   R3,CMDLEN
*
*        LH    R3,Ø(R2)             LENGTH
*
*        MVC   CMDLEN(2),Ø(R2)      COPY COMMAND LENGTH
*
*        BCTR  R3,Ø                 SUBTRACT 1 FROM LENGTH FOR MOVE
*        EX    R3,MOVE              MOVE COMMAND FROM PARAMETER LIST
***************************
***************************
***************************
***************************
*
         AUTHON               AUTH ON
*
         MODESET KEY=ZERO       SUPERVISOR MODE FOR SVC GENERATION
         SR    RØ,RØ            RØ=Ø
         LA    R2,CMD
         MGCRE MF=(E,LAREA),TEXT=(R2),CONSID=MASTER
         MODESET KEY=NZERO      RESET SUPERVISOR MODE
*
         AUTHOFF              AUTH OFF
*
***************************
** RESTORE REGISTERS ******
***************************
EXIT     DS    ØH                       NORMAL RETURN.
         L     R13,SAVEAREA+4
         LM    R14,R12,12(R13)
         SR    R15,R15
         BR    R14
```

```
***************************
** DATA FIELDS ************
***************************
SAVEAREA DC    18F'0'
***************************
MOVE     MVC   XCMD(0),2(R2)          MOVE COMMAND FROM AN ARGUMENT
         DS    0F
*
CMD      DS    0CL062
CMDLEN   DC    XL2'005'               NO SECURITY TOKEN
XCMD     DC    CL60'D C'              COMMAND
*
FILLER   DC    CL60''
*
LAREA    MGCRE MF=L
*
MASTER   DC    F'00'
*
*
         LTORG
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         END
```

JCL TO START THE SERVER

The server can be started as a job:

```
//I990557A JOB (01808),
//             'SYSTEM TEAM',
//             MSGCLASS=R,
//             MSGLEVEL=(1,1),
//             NOTIFY=&SYSUID,
//             CLASS=4
//*
//STEP01  EXEC PGM=BPXBATCH,
//            PARM='SH /u/i990557/java/jni/pgm04/Server.sh'
```

```
//*
//STDOUT   DD PATH='/tmp/mvs_server.out',
//            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//            PATHMODE=SIRWXU
//*
//STDERR   DD PATH='/tmp/mvs_server.err',
//            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//            PATHMODE=SIRWXU
//*
//STDENV   DD *
CLASSPATH=/u/i990557/java/jni/pgm04:$CLASSPATH
/*
//STEP02   EXEC PGM=IKJEFT01
//HFSOUT   DD PATH='/tmp/mvs_server.out'
//HFSERR   DD PATH='/tmp/mvs_server.err'
//STDOUTL  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=133,BLKSIZE=137)
//STDERRL  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=133,BLKSIZE=137)
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 OCOPY INDD(HFSOUT) OUTDD(STDOUTL)
 OCOPY INDD(HFSERR) OUTDD(STDERRL)
/*
```

The first step of this job starts the server using a shell script. The following script starts the server on port 5001, which is passed as a argument to the Server class:

```
BROWSE — /u/i990557/java/jni/pgm04/Server.sh —— Line 00000000 Col 001 054
 Command ===>                                                  Scroll ===>
HALF
******************************** Top of Data ***********************************
echo '*** Starting Java Server on OS/390 ***'

export STEPLIB=SMAINT.I990557.ASM.LOAD
export CLASSPATH=/u/i990557/java/jni/pgm04/:$CLASSPATH
export LIBPATH=/u/i990557/java/jni/pgm04/:$LIBPATH

java Server 5001

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

You should modify the STEPLIB statement to point to your load library containing the Assembler programs IPLCASO and COMASO.

The second step of the job redirects STDOUT and STDERR to the JES2 spool, so you can control the server output in a familiar environment such as SDSF.

SERVER LOG

When the server is stopped, you can display its log using SDSF :

```
*** Starting Java Server on OS/390 ***
Server started using port 5001
>> Waiting for client...
   Client Socket opened...
   Data from Client : IPLINFO_
   Request type : IPLINFO_
   ->> IPLINFO
>> Waiting for client...
   Client Socket opened...
   Data from Client : SEND_CMDd a,l
   Request type : SEND_CMD
   ->> MVS command
       MVS command  : d a,l
>> Waiting for client...
   Client Socket opened...
   Data from Client : IPLINFO_
   Request type : IPLINFO_
   ->> IPLINFO
>> Waiting for client...
   Client Socket opened...
   Data from Client : STOP____
   Request type : STOP____
```

CLIENT IMPLEMENTATION

The client is implemented using several Java classes which implement the graphical interface.

Client.java.

```
import java.net.*;

public class Client
{
      public static void main (String args[])
      {
            server_request server_req =  new server_request();
            display_frame df =  new display_frame(server_req);
      }
}
```

display_frame.java.

```
import java.awt.*;

public class display_frame extends Frame
{
```

```
   final static int HS = 750;
   final static int VS = 100;

   public display_frame(server_request server_req)
   {
      addWindowListener( new manage_general_frame());

      status_window sw;

      setTitle("Java - MVS Manager");
      setSize(HS, VS);
      setBackground(Color.lightGray);
      setLayout( new BorderLayout( 2,2));

      setMenuBar( new frame_menu( this, server_req));

      add(sw = new status_window(),"South");
      add( new parm_window(server_req, sw), "North");

      show();
   }
}
```

frame_menu.java.

```
import java.awt.*;

public class frame_menu extends MenuBar
{
    public MenuItem mi_ipl_info, mi_stop_server, mi_about, mi_exit;
    public MenuItem mi_send_cmd;

    public frame_menu(Frame top_frame,
                      server_request server_req)
       {
          Menu m_action = new Menu("Action");

           mi_ipl_info = new MenuItem("Get last IPL Information");
           m_action.add(mi_ipl_info);
           mi_ipl_info.addActionListener( new

               manage_frame_menu(server_request.IPLINFO, server_req));

           mi_send_cmd = new MenuItem("Send a MVS command");
           mi_send_cmd.addActionListener( new
               manage_frame_menu(server_request.SEND_CMD, server_req));

           m_action.add(mi_send_cmd);

          m_action.addSeparator();

           mi_stop_server = new MenuItem("Stop Server");
```

```
          mi_stop_server.addActionListener( new

                        manage_frame_menu(server_request.STOP, server_req));
          m_action.add(mi_stop_server);

          m_action.addSeparator();

          mi_exit = new MenuItem("Exit");
          i_exit.addActionListener( new manage_frame_menu(server_request.EXIT,
                                                          server_req));

          m_action.add(mi_exit);

          this.add(m_action);

          Menu m_help = new Menu("Help");
          mi_about = new MenuItem("About...");
          m_help.add(mi_about);
          mi_about.addActionListener( new
manage_frame_menu(server_request.ABOUT,
                                                                server_req));

          this.add(m_help);
       }
}
```

manage_frame_menu.java.

```
import java.awt.*;
import java.awt.event.*;

public class manage_frame_menu  implements ActionListener
{
  private int id;
  private Frame top_frame;
  private server_request server_req;

  public manage_frame_menu( int id,
                        server_request server_req)
     {
          this.id = id;
          this.server_req = server_req;
     }

  public void actionPerformed(ActionEvent e)
     {
          switch(id)
          {
               case server_request.EXIT:
                    System.exit(0);
                    break;
               case server_request.IPLINFO:
                    server_req.process_request(id);
```

```
                    break;
               case server_request.SEND_CMD:
                    send_cmd_frame scf =  new send_cmd_frame(server_req);
                    break;
                     case server_request.STOP:
                    server_req.process_request(id);
                    break;
               case server_request.ABOUT:
                    server_req.frame(id,"Java - MVS Manager Version 1.Ø" );
                    break;

          }
     }

}
```

manage_general_frame.java.

```
import java.awt.*;
import java.awt.event.*;

class manage_general_frame  extends WindowAdapter
{
     public void windowClosing(WindowEvent e)   // close with system X button
     {
          Window w = e.getWindow();
          w.dispose();                    // close the window
     }
}
```

manage_parm_window.java.

```
import java.awt.*;
import java.awt.event.*;

public class manage_parm_window  implements ActionListener
{
  static final int UPDATE       = Ø;
  static final int DISCONNECT   = 1;

  private int id;
  private status_window sw;
  private server_request server_req;
  private TextField tf_hostname;
  private TextField tf_port;

  public manage_parm_window( int id,
                             status_window sw,
                             server_request server_req,
                             TextField tf_hostname,
                             TextField tf_port)
     {
          this.id = id;
```

```
            this.sw = sw;
            this.server_req = server_req;
            this.tf_hostname = tf_hostname;
            this.tf_port = tf_port;
      }

  public void actionPerformed(ActionEvent e)
      {
            switch(id)
            {
                  case UPDATE:
                        sw.print_hostname(tf_hostname.getText());
                        sw.print_port(tf_port.getText());
                        // System.out.println(tf_hostname.getText());
                        server_req.set_hostname(tf_hostname.getText());
                        server_req.set_port(Integer.parseInt(tf_port.getText()));

                        break;
            }
      }
}
```

parm_window.java.

```
import java.awt.*;

public class parm_window extends Panel
{

 public parm_window(server_request server_req,
                       status_window sw)
  {
    Font my_font =  new Font("Arial", Font.BOLD,20);
    setLayout( new FlowLayout());
    setBackground(Color.green);

    Label l_server_hostname = new Label("Server HostName: ");
    l_server_hostname.setFont(my_font);
    this.add(l_server_hostname);
    TextField tf_server_hostname =  new TextField(20);
    this.add(tf_server_hostname);

    Label l_server_port =  new Label("Port: ");
    l_server_port.setFont(my_font);
    this.add(l_server_port);
    TextField tf_server_port =  new TextField(5);
    this.add(tf_server_port);

    Button b_update =  new Button("Update Server Properties" );
    b_update.addActionListener( new
manage_parm_window(manage_parm_window.UPDATE,
                                                      sw,
                                                      server_req,
```

```
                                                    tf_server_hostname,
                                                    tf_server_port));
    this.add(b_update);
  }
}
```

send_cmd_frame.java.

```
import java.awt.*;
import java.awt.event.*;

class send_cmd_frame extends Frame implements ActionListener
{

  private TextField tf_send_cmd;
  private server_request server_req;

  public send_cmd_frame(server_request server_req)
     {
       this.server_req = server_req;
          addWindowListener( new manage_general_frame());
          setSize(600, 130);
               setBackground(Color.lightGray);
          setTitle("Send a MVS Command..." );
          setLayout(new FlowLayout());
          Label l1 = new Label("Command:");
          Font my_font = new Font("Arial", Font.BOLD,20);
          l1.setFont(my_font);
          add(l1);
          tf_send_cmd = new TextField(80);
          add(tf_send_cmd);
               Button b_send =  new Button("Send");
          b_send.addActionListener( this);
          add(b_send);
               show();
     }

  public void actionPerformed(ActionEvent e)
     {
       System.out.println(tf_send_cmd.getText());
       server_req.process_request(server_request.SEND_CMD,
                                  tf_send_cmd.getText());
     }
}
```

server_request.java.

```
import java.awt.*;
import java.net.*;
import java.io.*;

public class server_request
{
```

```
      static final int EXIT     = 0;
static final int IPLINFO  = 1;
static final int ABOUT    = 2;
static final int STOP     = 3;
static final int SEND_CMD = 4;

static final int ERROR    = 999;

static final String IPLINFO_str   = "IPLINFO_";
      static final String STOP_str      = "STOP____";
      static final String SEND_CMD_str  = "SEND_CMD";

      private String hostname;
      private int port;
      private String reply;
      private String error;

      public server_request()
       {
            this.hostname = "unkown";
            this.port = 0;
       }

      public String get_hostname()
       {
            return (hostname);
       }

      public int get_port()
       {
            return (port);
       }

      public void set_hostname(String new_hostname)
       {
            this.hostname = new_hostname;
            // System.out.println("New hostname = " + new_hostname);
       }

      public void set_port(int new_port)
       {
            this.port = new_port;
            // System.out.println("New port = " + new_port);
       }

      public String send_request(String request)
       {
            try
            {
        error = "NO";

              InetAddress address;
```

```
        Socket client_socket;

        address = InetAddress.getByName(hostname);

        client_socket = new Socket(address, port); // open socket


        BufferedWriter write_buffer =          // output stream
          new BufferedWriter(new
                    OutputStreamWriter(client_socket.getOutputStream(),"Cp037"));
        BufferedReader read_buffer =           // input stream
          new BufferedReader(new
                    InputStreamReader(client_socket.getInputStream(),"Cp037"));

           write_buffer.write(request);
           write_buffer.newLine();
        write_buffer.flush();

        reply = read_buffer.readLine();

        write_buffer.close();
        read_buffer.close();
        client_socket.close();  // close socket

      }
      catch (Exception e) {error = "YES";}

      return(reply);
}

public String process_request(int id,
                                String input_string)
{
      String request_string;
         String reply = new String("");;
      switch(id)
      {
        case SEND_CMD:
            request_string = SEND_CMD_str + input_string;
            // System.out.println(request_string);
            reply = this.send_request(request_string);
            reply = reply.substring(0,21);
break;
      }

      this.frame(id, reply);
      return(reply);
}

public String process_request(int id)
{
      String request_string;
 String reply = new String("");
```

```
      switch(id)
      {
       case IPLINFO:
            request_string = IPLINFO_str;
            reply = this.send_request(request_string);
         break;
  case STOP:
            request_string = STOP_str;
            reply = this.send_request(request_string);
            reply = reply.substring(0,21);
         break;
  case ABOUT:
            request_string = STOP_str;
            reply = this.send_request(request_string);
        break;
       }

      if (error.equals("YES") )
       {
         System.out.println("Error detected...");
         id = server_request.ERROR;
         reply = "";
       }

       this.frame(id, reply);

      return(reply);
}

public void frame(int id, String reply)
{
  Frame request_frame = new Frame();
  request_frame.addWindowListener(new manage_general_frame());

    request_frame.setBackground(Color.lightGray);
    request_frame.setTitle("Java - MVS Manager");

  Font my_font1 = new Font("Arial", Font.BOLD,20);
  Font my_font2 = new Font("Courier", Font.BOLD,12);

   Label l1, l2;

   switch(id)
      {
          case ERROR:
            request_frame.setSize(400, 200);
            request_frame.setLayout(new GridLayout(2,1));
            l1 =  new Label("Network error detected...",Label.CENTER);
            l1.setFont(my_font1);
            request_frame.add(l1);
            l2 = new Label("Verify Server properties...",Label.CENTER);
            l2.setFont(my_font2);
            request_frame.add(l2);
```

```
                break;
              case IPLINFO:
                request_frame.setSize(400, 200);
             request_frame.setTitle("Last IPL information:");
                request_frame.setLayout(new GridLayout(7,2));
                request_frame.add(new Button("SMFID:"));
                request_frame.add(new Label(reply.substring(0,4),Label.CENTER));
                request_frame.add(new Button("DATE:"));
                request_frame.add(new Label(reply.substring(4,10),Label.CENTER));
                request_frame.add(new Button("TIME:"));
                request_frame.add(new Label(reply.substring(10,18),Label.CENTER));
                request_frame.add(new Button("SYSRES - Volser:"));
                request_frame.add(new Label(reply.substring(18,24),Label.CENTER));
                request_frame.add(new Button("SYSRES - Address:"));
                request_frame.add(new Label(reply.substring(24,27),Label.CENTER));
                request_frame.add(new Button("LOADPARM:"));
                request_frame.add(new Label(reply.substring(27,35),Label.CENTER));
                request_frame.add(new Button("MVS Version:"));
                request_frame.add(new Label(reply.substring(35,43),Label.CENTER));
                break;
              default:
                request_frame.setSize(400, 200);
                l1 = new Label(reply,Label.CENTER);
                l1.setFont(my_font1);
                request_frame.add(l1);
                break;
            }
         request_frame.show();
      }
}
```

status_window.java.

```
import java.awt.*;

public class status_window extends Panel
{
 private Label hostname, port;

 public status_window()
  {
     Label l1, l2, l3;
     Font my_font = new Font("Arial", Font.BOLD,12);

     setBackground(Color.lightGray);
     setLayout(new FlowLayout(FlowLayout.LEFT));
     l1 = new Label("Current Server Properties =" );
     l1.setForeground(Color.blue);
     add(l1);

     l2 = new Label("HostName: ");
     l2.setForeground(Color.black);
     add(l2);
```

```
      add(hostname = new Label("?????????"));
      hostname.setForeground(Color.darkGray);
      hostname.setFont(my_font);

      l3 = new Label("Port: ");
      l3.setForeground(Color.black);
      add(l3);

      add(port = new Label("?????????"));
      port.setForeground(Color.darkGray);
      port.setFont(my_font);

  }
 public void print_hostname(String msg)
      {
           hostname.setText(msg);
      }
 public void print_port(String msg)
      {
           port.setText(msg);
      }

}
```

COMPILING JAVA CLASSES ON YOUR PC

On your PC, you can use the following command file to compile the different Java classes:

```
javac status_window.java
javac send_cmd_frame.java
javac manage_general_frame.java
javac manage_display_frame.java
javac manage_frame_menu.java
javac display_frame.java
javac frame_menu.java
javac manage_parm_window.java
javac parm_window.java
javac server_request.java
javac Client.java
```

STARTING THE CLIENT

To start the client, you should enter, in a DOS session, the following command:

```
java Client
```

Then you will get a client window as shown in Figures 1 and 2.

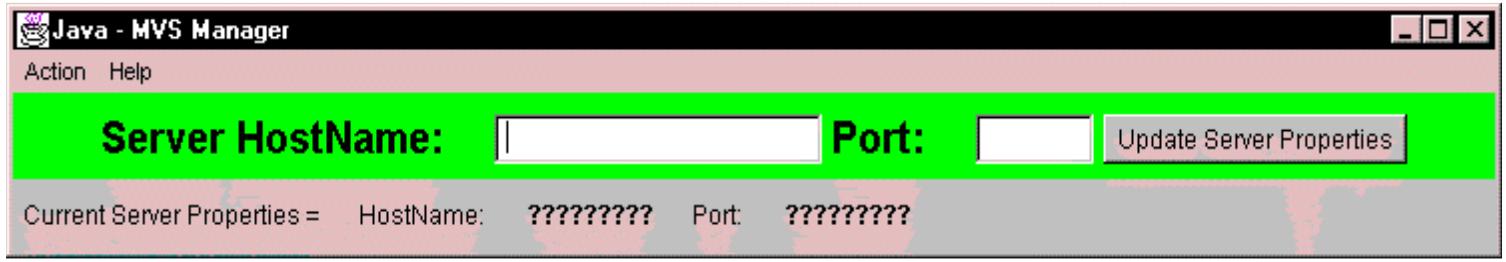

Figure 1: First specify and update the 'Server Properties': hostname and port. number

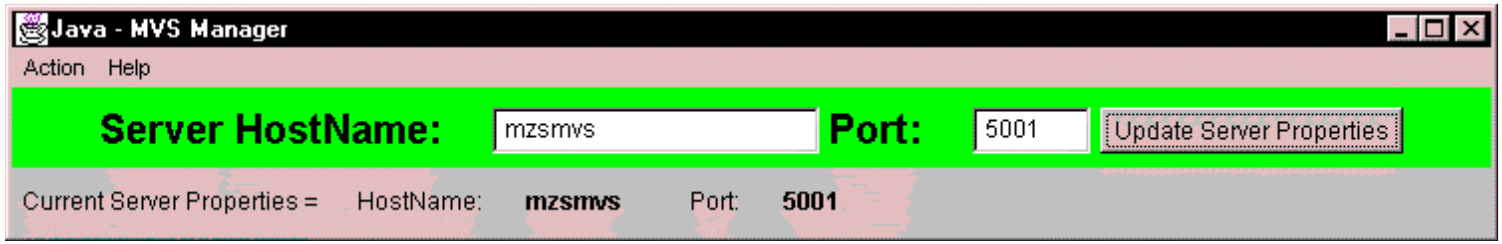

Figure 2: Select an action in the 'Action' menu

Figure 3: An example of an action could be – 'Get Last IPL Information'

Reference modification

THE PROBLEM

Have you ever needed to manipulate just a portion of a field? We often end up redefining the field five different ways for the different configurations and variants needed or we have to redefine it as a table of single bytes to be manipulated one byte at a time. As a result, we end up with horrors like the following:

```
  Ø5 DEPARTMENT.
     1Ø DEPT-DIVISION            PIC XXX.
     1Ø DEPT-GLASS.
        15  DEPT-GLASS-FACTORY   PIC XX.
        15  DEPT-GLASS-FUNCTION  PIC XX.
     1Ø DEPT-FIBER            REDEFINES DEPT-GLASS.
        15  DEPT-FACTORY         PIC X.
        15  DEPT-FUNCTION        PIC XXX.
     1Ø DEPT-RESEARCH         REDEFINES DEPT-GLASS.
        15  DEPT-FACTORY         PIC X.
        15  DEPT-FUNCTION        PIC XX.
        15  FILLER               PIC X.
     1Ø DEPT-TABLE            REDEFINES DEPT-GLASS.
        15  DEPT-TABLE-BYTE OCCURS 7 TIMES
                           INDEXED BY DPTX
                              PIC X.
```

In the PROCEDURE DIVISION:

```
IF DEPT-DIVISION = 'GLS'
  MOVE DEPT-GLASS-FUNCTION TO CURRENT-FUNCTION
ELSE
IF DEPT-DIVISION = 'FBR'
  MOVE DEPT-FIBER-FUNCTION TO CURRENT-FUNCTION
ELSE
IF DEPT-DIVISION = 'RCH'
  MOVE DEPT-RESEARCH-FUNCTION TO CURRENT-FUNCTION.
```

All this just to handle a few lines of code to split out a substring of ‘department’.

THE SOLUTION

Using reference modification, we can avoid all this fuss and bother. The field needs to be defined only once:

```
Ø5  DEPARTMENT          PIC X(7).
```

All the work is done in the code without all the extra data division coding:

```
   *    ** Move function from input record to work area **
IF DEPARTMENT (1:3) = 'GLS'
  MOVE DEPARTMENT (6:2) TO CURRENT-FUNCTION
ELSE
IF DEPARTMENT (1:3) = 'FBR'
  MOVE DEPARTMENT (5:3) TO CURRENT-FUNCTION
ELSE
IF DEPARTMENT (1:3) = 'RCH'
  MOVE DEPARTMENT (5:2) TO CURRENT-FUNCTION.
```

As you can see, except for the comment line, there is no more PROCEDURE DIVISION code than with the first method. The savings are in the DATA DIVISION. The general format for reference modification is:

```
data-name (n:m)
```

or:

```
data-name (n:)
```

where ‘n’ is the leftmost character position within the field and ‘m’ is the length (or byte count) of the data you want to deal with. If ‘m’ is not indicated, the operation will use all of the data from and including the ‘n’th position to the end of the field.

The data item (data-name) referenced must have a USAGE of ‘DISPLAY’. It may be qualified and/or subscripted.

‘n’ and ‘m’ must be positive, non-zero integers or arithmetic expressions. Obviously, they must be within the number of characters defined by data-name. They can be elementary data items themselves as long as they are positive, non-zero integers.

Arithmetic expressions that calculate to a non-integer are truncated. In other words, ‘8 / 3’ would yield ‘2’.

Allan Kalar
Systems Programmer (USA)

Shifting text and data in ISPF/PDF

THE PROBLEM

Sometimes your edit window does not show the full line you are working on, and you want to insert something into a line, but the window gets in the way. The line is long enough to accommodate the extra text, but the empty part is off the window to the right. As you try to insert the text, the existing text reaches the right margin of the window and will not go any further.

Maybe you want to delete something out of a line (perhaps spaces) and left shift everything behind it. As you hit the delete key, the line obediently shifts left and spaces are inserted between the last character on the window and the right edge. When you shift your window to the right to see the right end of the line(s), the rest (unseen portion) of the line is still where you left it – with spaces between it and the left portion of the line.

A SOLUTION

There is a way around these problems. Part of the solution involves the use of the BNDS. I will repeat some of the information for those who need to brush up.

Firstly, we will set the bounds of the area you want to work with by entering BNDS in the line command area. Any line will do.

```
000100  1000-EDIT-INPUT.
BNDS00      IF POINPUT-CUSTOMER-DEMAND-DATE < JOLL
000300        MOVE '           JOLLY ROGER DATE EX
```

Then we hit <enter> and we see:

```
000100  1000-EDIT-INPUT.
=BNDS> <
000200      IF POINPUT-CUSTOMER-DEMAND-DATE < JOLL
000300        MOVE '           JOLLY ROGER DATE EX
```

There is a '>' at the true right hand boundary of the new BNDS line. We cannot see it because it is off the screen to the right. We will re-set the c#by entering '<' and '>' (boundary markers) where we want them.

```
000100  1000-EDIT-INPUT.
=BNDS> <    <                              >
000200      IF POINPUT-CUSTOMER-DEMAND-DATE < JOLL
000300        MOVE '           JOLLY ROGER DATE EX
```

Notice that both the new boundary markers and the old markers are on the screen. PDF can tell which we have just entered and which are the old ones. After we press <enter>, the screen will look like this:

```
000100  1000-EDIT-INPUT.
=BNDS>      <                              >
000200      IF POINPUT-CUSTOMER-DEMAND-DATE < JOLL
000300        MOVE '           JOLLY ROGER DATE EX
```

Now we can shift the line using the line shift commands '(' and ')' in the line command area. '(' is the 'left shift' command and '00)' is the 'right shift'.

```
000100  1000-EDIT-INPUT.
=BNDS>      <                              >
(102 00     IF POINPUT-CUSTOMER-DEMAND-DATE < JOLL
000300        MOVE '           JOLLY ROGER DATE EX
```

Note the cursor is under the '2' in column 3. We want a left shift of 10 characters. Leaving the cursor under the '2' tells PDF where the number '10' ends. When we entered text in the 'data' area, PDF could tell which entries were new. In the Line Command area, it cannot. If we move the cursor somewhere else, we would have to enter a blank after the number to delineate the shift quantity as in the following line:

```
(10 00      IF POINPUT-CUSTOMER-DEMAND-DATE < JOLL
```

In either case, after we press <ENTER>, we get:

```
000100  1000-EDIT-INPUT.
=BNDS>      <                              >
000200      -CUSTOMER-DEMAND-DATE            < JOLL
000300        MOVE '           JOLLY ROGER DATE EX
```

Notice how everything shifted past the left boundary gets sent to the 'bit bucket'. Obviously we did not want to reset the boundaries. If we'd left the boundaries as we found them the result would have been similar to this:

```
000100  1000-EDIT-INPUT.
=BNDS> <
000200  NPUT-CUSTOMER-DEMAND-DATE < JOLLY-ROGER-LI
000300        MOVE '           JOLLY ROGER DATE EX
```

OK, so it is stupid. It is more likely that we will have deleted the spaces out of the message being moved. But, you get the idea. The important part is that we can do this in groups.

```
((0100  1000-EDIT-INPUT.
000200      IF POINPUT-CUSTOMER-DEMAND-DATE < JOLL
((1000        MOVE '           JOLLY ROGER DATE EX
```

We can enter the shift quantity behind either of the group commands. By the way, if we do not enter a shift quantity, the default is '2'. BNDS can be used to limit the effect of other commands, such as copy and overlay. Only the portions within the Bound marks get copied or overlaid. Do not forget to reset the boundaries after you have finished using them.

Allan Kalar
Systems Programmer (USA)

OS/390 Version 2 Release 10 directions

OS/390 VERSION 2 RELEASE 9

In *MVS Update*, April 2000, Issue 163 we looked at the latest release of OS/390 (Release 9). The functionality in Release 9 builds on the developments in previous releases: e-business enablement, Enterprise Application Intergration (EAI), server consolidation, technology extensions, and Unix System Services enhancements. Release 10 continues these themes.

OS/390 VERSION 2 RELEASE 10

Release 10, available on 29 September 2000, offers:

- Ease in porting C and C++ applications to S/390 through Extra Performance Linkage
- Application development flexibility through downward compatibility of Language Environment
- Better security in the network operating environment and the TCP/IP environment

- Increased flexibility in data storage and management
- Standard workstation access to SAM, PDS(E), and VSAM files
- Wizard technology, multimedia instructional animations, and Automatic Alter support to advance ease-of-use.

LINUX, JAVA, EJB, AND CORBA

Linux for S/390 is a separate entity to OS/390. Linux for S/390 can execute in one of three modes: S/390 native, S/390 LPAR, and VM guest (further information can be found at the following URL: http://www.ibm.com/s390/linux). Also, separate from the OS/390 product, WebSphere Application Server for OS/390 allows users to deploy Enterprise Java Beans and CORBA Business Objects on the S/390. IBM also offers Java 2 technology on OS/390 via the IBM Developer Kit for OS/390, Java 2 Technology Edition. (Further information on WebSphere Application Server for OS/390 can be found at the followinf URL: http://www.ibm.com/s390/ebusiness.)

PREREQUISITES FOR RELEASE 10

Release 10 will only run on servers that implement certain architectural enhancements. The following IBM servers have these enhancements:

- All models of the S/390 Parallel Enterprise Server except Release 1 models
- All models of the S/390 Multiprise
- All PC Server S/390 servers and RS/6000 with S/390 Server-on-Board models
- All S/390 Integrated Servers.

BEYOND RELEASE 10

Further information regarding Release 10 can be found on the IBM Web site. In the next edition we will review OS/390 directions beyond Release 10, including future usage-based pricing models, future hardware requirements, and IBM's future OS/390 strategy.

Action Software International has announced Version 6.01 of Change Action, the only active Change Management solution for the OS/390 environment. Change Action's tracking features monitor all changes in the system, especially those made to sequential or partitioned datasets.

In addition, Change Action's management features allow users to provide levels of control that protect data in relation to its importance. Change Action's interface to Tivoli Service Desk (formerly Info/ Management) turns TSD from a passive electronic paper system into an active change management system via Change Action's Tracking and Control features.

Change Action also includes other features and functions that can improve systems availability and can reduce both short- and long-term costs. New features in Version 6.01 include: the command tracking and control component that eliminates the need to search SYSLOG for commands that have been entered. PET (Program Execution Tracking) has been extended to support JES2 PROCs, TSO CLISTs and REXX EXECs. Tracking can continue even after the Started Task has been stopped.

Action Software International, 20 Valleywood Drive, Suite 107, Markham, Ontario, Canada, L3R 6G1
Tel: (905) 470 7113
Fax: (905) 470 6507

Action Software GmbH, Im Eichli 9, CH-6315 Oberaegeri, Switzerland.
Tel: +41 41 754 5088
Fax: +41 41 754 5089

http://www.actionsoftware.com

* * *

IBM has announced support for IMS Version 7 in its Extended Terminal Option Support (ETO Support) Version 1 Release 2, a front end to help implement, customize, and exploit the benefits of ETO.

The feature allows terminals, printers, and LTERMs to be created dynamically, reducing the need for IMSGENs and, therefore, helping to improve IMS availability.

ETO Support provides a front-end to ETO for implementation, tailoring, and control. It includes a batch interface for the bulk loading of descriptors and options and manages ETO through a set of dynamically refreshable tables.

It helps determine how structure names are created, assigns specific LUs to select logon descriptors, overrides ASOT/ALOT values for specified LUs, and controls the disposition of conversations and queued messages at signoff.

It can also control which unknown destinations create an SPQB/CNT, optionally perform command authorization, support batch loading/adding of nodes, LTERMs, and user IDs to the control database, and provide printer/LTERM cross reference support in on-line and batch model.

Contact your local IBM representative for further information.

http://www.ibm.com

* * *