In this issue

3  OS/390 strategy overview
9  Copying data between MVS and PCs
14 Controlling tape information
30 Copying to compressed tape using Stack
49 Executing a PL/I program from REXX
52 Keeping track of load module changes – part 2
64 Maintaining a profile in ISPF/PDF
72 MVS news
MVS Update

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

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.

© Xephon plc 2000. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

Printed in England.
OS/390 strategy overview

INTRODUCTION
The fourth quarter of this year will mark the beginning of a series of radical changes to both OS/390 and the System/390 platform. In *MVS Update* Issue 166 July 2000, we previewed some of the new features that will be available in the September 2000 release of OS/390 Version 2 Release 10. This month we consider some of the future directions for OS/390 both Release 10 and beyond, and we will review parallel developments in the System/390 platform.

IBM has been consistent in making releases of OS/390 available at approximate six month intervals. These General Availability (GA) times provide a consistent framework for users and purchasing managers. This regularity is made possible because IBM is developing OS/390 considerably in advance of the ‘next’ release. Therefore, the reliability of the IBM ‘previews’ of the forthcoming releases can be guaranteed. However, IBM provides preview information for forthcoming releases only six months in advance. This article reviews IBM’s short and long-term OS/390 strategy.

IBM’S OS/390 STRATEGY
IBM’s basic strategy for OS/390 has been clear from the consistency of themes in the recent releases (see *MVS Update* for overviews of recent releases). We have noted previously that e-business enablement, Enterprise Application Integration (EAI), server consolidation, Unix System Services, and ease of use are consistently enhanced with each release. As an example, consider the consistent elements between the two latest releases of OS/390 Version 2 (Releases 8 and 9).
The following is a summary of the principal elements of OS/390 Release 8:

- e-business enablement:
  - Security improvements in managing cryptographic keys, keyrings and certificates.
  - Crypto support for SNA communications and TN3270 users.
- Enterprise Application Integration (EAI):
  - Prerequisite for SAP R/3 Application Server on System/390.
- Server consolidation:
  - Significant print serving capability for new workloads, new clients, new protocols.
  - Enhanced print management.
- Unix System Services enhancements:
  - Support for more Unix programs (eg PERL scripts).
  - Enhancements for porting Unix programs.

The following is a summary of the principal elements of OS/390 Release 9:

- e-business enablement:
  - Standard support in WebSphere Application Server for Java Server Pages and servlets.
  - Enhanced security functions and exportability of security functions.
- Enterprise Application Integration (EAI):
  - Release 9 is a base for Customer Relationship Management (CRM) and ERP solutions such as SAP R/3 and DB.
- Server integration:
  - File and print serving for Windows workstations with Server Messaging Block (SMB).
• Unix System Services
  – DBX debugger, addition UNIX98 functions
  – easier maintenance for TCP/IP
  – data sharing for files systems (eg HFS)

These trends will continue with Release 10 and beyond, because they mirror the requirements of large enterprises that are:

• Evaluating or deploying e-business solutions.
• Reducing functions on the PC and moving towards browser-based access.
• Investing in high-bandwidth TCP/IP networks and routers.
• Physically moving and consolidating Unix, and NT servers to larger centres to reduce management overheads.
• Bringing departmental applications to large centres and cloning the remaining ones.
• Combining central System/390, Unix and NT servers in hybrid applications (eg Web, ERP, OLTP, collaborative computing, etc).
• Integrating Web, application and database servers in major centres to provide robust services.

Therefore, the trends in OS/390 are both predictable and consistent. However, significant changes will occur in the fourth quarter of 2000, and these are linked to parallel developments in the System/390 processor.

PROCESSOR DIRECTIONS
As with OS/390 IBM has been consistent in its release dates of processor technology. Between the release of the R1 in September 1994 and the G6 in May 1999, processors have been released at alternate intervals of nine and fourteen months successively. The
proposed release date for the next generation processor – the G7 – code-named the ‘Freeway’ early in the fourth quarter, indicates a much longer development cycle (around 18 months). This is because the G7 Freeway will support 64-bit addressing, and so the testing time has been considerably longer. The 64-bit G7 will mark as big a change in the mainframe world as the 3081 did twenty years ago.

64-BIT ADDRESSING
64-bit addressing provides considerable benefits for OS/390. These can be divided into improvements to real memory, integer arithmetic, and virtual memory.

Real memory:
• Improves performance by massively reducing data movement.
• Reduces costs because ES/CS movement shows non-linear growth with system size.
• Supports larger structures within the Coupling Facilities.

Integer arithmetic:
• Increases performance.
• Increases interoperability with other 64-bit platforms.

Virtual memory:
• Large virtual memory allows the exploitation of large real memory.
• Supports emerging applications such as Java and Enterprise JavaBeans (EJB).

The 64-bit addressing will be desirable for some current mainframe applications, as well as certain Unix/NT and Linux applications, and will be essential for future large MIPS systems (especially when deployment of 3000+ MIPS systems begin).
However, users should note that although the Freeway will be released in the fourth quarter 2000, OS/390 Version 2 Release 10 does not appear to contain full support for 64-bit operations.

LONG-TERM STRATEGIES

Users will be able to exploit 64-bit addressing in the short-term future. In the longer term there are other impacts tangentially associated with the operating system that will affect users.

Usage-based pricing

A longer-term application for the G7 processor will be the support of a new usage-based pricing model. Users have long complained that the current pricing models penalize those with unused capacity or those wishing to migrate large non-System/390 applications to the mainframe. These high software costs have also been detrimental to IBM because it has prevented the company from expanding its user base into the SME (Small to Medium Enterprise) market sector.

The pricing model (in conjunction with IBM’s Java, Linux, and Unix initiatives) provides further indications that IBM sees the future of the System/390 platform as the basis for applications rather than as a stand-alone operating system.

In the post OS/390 Release 10 timeframe, software pricing models will be based on the number of MIPS consumed. The key consideration for users is that metering will require both hardware and software support. Users wishing to benefit from these changes will require a G7 processor and the future release of OS/390 that supports usage-based pricing. Usage-based pricing or ‘Software Value Pricing’ will be of considerable benefit to users who will be able to pay for the capacity used, not the total system capacity. The financial benefits of usage-based pricing will probably mean that there will be a rapid move to the new processor, when the facility becomes available.
WebSphere
Since its release, WebSphere has gained considerable prominence. From a strategic perspective, IBM considers WebSphere to be the successor to CICS and IMS. This is because IBM predicts that both CICS and IMS will decline over time in favour of WebSphere, which has been Web and object database-enabled from the outset. This would suggest a widespread move from OLTP to e-TP (see MVS Update Issue 163, April 2000, page 7).

FUTURE OS/390 REQUIREMENTS
We have seen that there is going to be considerable new functionality in forthcoming releases of OS/390, but users who wish to exploit this functionality will require future processor upgrades. We have already seen that the release of OS/390 Release 10 will require a G2 processor or higher. A big leap in processor capacity will be required in the first half of 2001 (the Release 11 timeframe), which will require a G5 processor or above. By the first half of 2002 (the Release 13 timeframe) it is probable that OS/390 will require WLM operating in Goal Mode.

CONCLUSIONS
In the short term, OS/390 will continue to enhance its position as a platform supporting e-business enablement, Enterprise Application Integration (EAI), server consolidation, Unix System Services, and ease of use. Also in the short term, OS/390 will support 64-bit addressing. In the mid-term future, associated developments in usage-based pricing will bring down costs for those users running the latest processor and operating system release. In the longer-term future, it is likely that the role of WebSphere will increase in importance at the expense of CICS and IMS.
INTRODUCTION

MVS files can be copied from one MVS system to another via a PC. This is commonly done (for relatively small files) when there is no other connection between the source and target MVS systems. The Xephon Website has a good example of this, distributing source code to the subscribers.

The intermediate files are stored on a PC disk or diskette, and the file transfers are done by PC software. The PC 3270 terminal emulators usually have a built-in file transfer facility, and ISPF also has had its own transfer facility (option 3.7) since Version 4.2 in 1996. I will not discuss them here. You can also use various ZIP utilities to compress and decompress the data on the PC, but I will not describe them here either.

There are two potential problems copying data:

- Character conversions
- Only sequential files can be transferred.

CHARACTER CONVERSION

You can transfer data as binary or text.

Binary

If you simply copy a file from MVS to PC in BINARY form, you will not be able to read it there because the PC treats the characters as ASCII (and they were EDDCIC on MVS). If you copy the file to a different MVS system in binary form, you will be able to read it again, but some of the special characters may not look the same if the new MVS system has a different codepage and/or character-set from the first MVS system. But, if you are transferring data between two MVS systems in the same country the codepage and character-set are usually the same.
Remember that MVS load modules can be successfully transferred in binary form regardless of the codepage and character-set on the target system.

Text

If you copy the file as TEXT from MVS to PC then to another MVS, you can at least read it on the PC. But, some character conversions may occur on the PC, and when you transfer it to the new MVS system a second character conversion may result. Usually these conversions are reversible, but sometimes two different EBCDIC characters are converted to one ASCII character.

Thankfully, most of the common characters have the same codes for the alphabet (a-z and A-Z) and numbers (0-9). But, if you are copying data with some special characters (eg program source or ISPF panels), some of those characters may need to be manually fixed after the transfer. This is probably familiar to people who have downloaded code from the Xephon Web site.

A good practice is to copy a file containing all characters X'00' to X'FF' in order, when you are copying data from the source system. When that file is copied to the target system you can easily check which characters have changed (use EDIT or BROWSE and set HEX ON). You can then use a simple EXEC like the one shown below to fix all other text file transfers from the same source system.

```rexx
/*============================>> REXX <<=============================*/
/* Used for translating characters which were unloaded using the */
/* English (UK) {85Ω} / ASCII (285) code table, but were originally */
/* copied using a different code table. */
/* */
/* This can be adapted as required by putting the appropriate code */
/* * values in the two translate tables. */
/* * ie TRANSLATE(string,table_out,table_in) */
/* * To get these values you need a file from x'00' - x'FF' created */
/* * using the same code table as the original data. */
/*===================================================================*/
Trace 0
Address TSO
"ALLOC FI(INDD) DA(input.file) SHR REUS" /* <--- update DSname */
"ALLOC FI(OUTDD) DA(output.file) SHR REUS" /* <--- update DSname */
"EXECIO * DISKR INDD (STEM in. FINIS"
Say in.Ø 'lines read from INDD'
Do i = 1 to in.Ø
```
SEQUENTIAL FILES
The intermediate files on the PC must be in sequential form. If the original file is not sequential, you must convert it to/from sequential form on the MVS system.

Copying a PDS (in text mode)
The article Convert PDS to sequential dataset in MVS Update Issue 164 had a sample EXEC to convert a PDS to a sequential file (formatted for IEBUPDTE input with './ADD NAME=memb' before the text of each member), which could later be used to recreate the PDS. But, the new PDS would have no member statistics.

Startool from Serena also has commands COMBINE and SEPARATE for this.

The new PDS can have the original member statistics, because COMBINE can create './ ADD NAME=memb  string-of-member-statistics' for each member and SEPARATE will use those 'string-of-member-statistics'. IEBUPDTE can also use the output from the COMBINE command, but it ignores the statistics on the ADD cards and creates a new PDS without any member statistics.

I have often added an extra member (called $$HEX) to the start of the PDS with all characters X'00' – X'FF' before doing a text transfer. It will be the first member shown in the sequential file, and hence easily checked to see which characters have been converted in the copying.

Prepare the translation tables and dataset names in the EXEC, and run it to create a (corrected) sequential file. Then you can recreate the PDS.

A PC file transfer tool is used to copy the file, in TEXT mode. This method is good for simple text files or program code where the record length is 80 and the record format is fixed block.
Copying a PDS (in binary mode)

Copying a PDS (in binary mode) can be used for any PDS. It uses TSO XMIT to invoke IEBCOPY to create an unloaded form of the PDS. The steps are:

1. Preallocate a sequential file: unload_dataset with DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120).
2. Use ISPF option 6 to transmit the PDS to the sequential file command – XMIT sysid.userid DS (dataset) OUTDS (unload_dataset).
3. Use a PC file transfer tool to copy – unload_dataset to PC_file using binary transfer.
4. Preallocate a target sequential file – target_dataset with DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120) on the target MVS.
5. Use a PC file transfer tool to copy – PC_file to target_dataset using binary transfer.
6. Use TSO RECEIVE on the target MVS to build a new PDS command: RECEIVE INDS(target_dataset) then reply with the desired PDS dataset name.

That is the preferred method for many Internet sites to supply software. All member statistics are preserved. It is the best way to copy load modules.

PDS index listing

When you are copying PDS files, it is often useful to make an index listing of the library and copy that (in text mode) as well. You can then read the index listing to see what members are in your file.

One way to create such an index listing is to list the library via ISPF option 3.4 (dataset list) and enter ‘PX’ beside the dataset name. That writes an index listing to your LIST dataset. Enter command ‘LIST’, enter ‘3’ to keep the existing list dataset, then copy it to a PC file.
Copying VSAM

VSAM files can also be converted to sequential files, transferred in binary mode, then recreated on the target system. This can be done using IDCAMS REPRO as seen in the following:

`// EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
        REPRO IDS(vsam.dataset) ODS(sequential.file)
/*`

For DB2 linear VSAM datasets you should use the DSN1COPY program instead of REPRO.

CONCLUSION

You can transfer data between MVS and PCs when it is in sequential format. Some conversion methods are described above.

You can transfer the data as text or binary:

- If you transfer as text you can read it on the PC, but may have character conversion problems.
- If you transfer as binary you cannot read it on the PC, but the EDCDIC codes will be the same on the target MVS system.

If you choose the appropriate methods described above, you can copy many different types of data from MVS to a PC and back again to another MVS system.

Ron Brown  
Systems Programmer (Germany)  © Xephon 2000

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.
Controlling tape information

INTRODUCTION

Daily DFSMSShsm operations such as migration, back-up or dump, require considerable quantities of tape. These tapes have to be protected from rewriting until they contain active data. DFSMSShsm allows several methods for protecting these tapes. We chose IBM’s recommended methodology:

- The tape is protected by RACF in the profile HSMHSM of the TAPEVOL class.
- If Tape Manager is used, it is most practical to use the SCRATCH pool instead of defining a separate pool for DFSMSShsm. This is necessary if we want to use automatic loaders or automated tape library.
- The tapes are defined without an expiration limit and we scratch them with the ARCTVEXT exit explicitly.

However, this methodology can cause some problems. For example, we noticed some information discrepancies between DFSMSShsm active tapes in RACF and/or Tape Manager. These inconsistencies were often caused by mistakes made in RACF or Tape Manager administration. For example, sometimes migration, back-up and dump were interrupted with a cancel command, or the Tape Manager was stopped irregularly. We also experienced PTF problems, which caused further discrepancies.

DFSMShsm can occupy several hundred tapes, which means that it is very hard to check these discrepancies. We made jobs that check and repair information about DFSMSShsm tapes in RACF and Tape Manager. We use two different tape managers DFSMSrmm and AutoMedia (ZARA), so we prepared jobs for both. These are shown below:

- HSMKONTR – lists all ML2, BACKUP, and DUMP tapes which DFSMSShsm occupies.
• HSMEXT – extracts the serial numbers of tapes from the previous job.

• HSMRAC – compares the RACF HSMHSM profile of the TAPEVOL class with the information from DFSMSrmm and updates RACF to settle the information.

• HSMRMM – compares information about active tapes from DFSMSrmm with the information from DFSMShsm.

• HSMZARA – compares information about active tapes from AutoMedia(ZARA) with the information from DFSMShsm.

JOB HSMKONTR

//useridH JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID,MSGLEVEL=(2,Ø)
*******************************************************************************
//** Job lists all DFSMShsm tapes and submit HSMEXTR job
//** which processes the reports
//** Note:
//** Job is separated into two parts because DFSMShsm uses dynamic
//** allocation for generated reports
*******************************************************************************
***** CHANGE userid TO YOUR USERID <========================================
*******************************************************************************
//DELWORK EXEC PGM=IDCAMS,REGION=IM
//SYSPRINT DD SYSOUT=*  
//SYSSIN DD *  
 DELETE 'userid.HSM.#DMP.LIST'
 DELETE 'userid.HSM.#BKP.LIST'
 DELETE 'userid.HSM.#MIG.LIST'
 SET MAXCC = Ø
/*
*******************************************************************************
//*** Lists all HSM tapes
*******************************************************************************
//LISTHSMV EXEC PGM=IKJEFTO1,DYNAMNBR=6Ø
//SYSTERM DD SYSOUT=X
//SYSPRINT DD SYSOUT=X
//SYSTSPRT DD SYSOUT=X
//SYSTSN DD *
 HSENDCMD WAIT LIST DVOL ODS(userid.HSM.#DMP.LIST)
 HSENDCMD WAIT LIST ML2 ODS(userid.HSM.#MIG.LIST)
 HSENDCMD WAIT LIST BVOL ODS(userid.HSM.#BKP.LIST)
/*
//#KONT1 EXEC JOB,N=HSMEXTR,D='userid.DFHSM.CNTL'
JOB HSMEXTR

//useridH JOB MSGCLASS=X,MSGLEVEL=(2,1),NOTIFY=&SYSUID,CLASS=A
//******************************************************************************
//**** CONTINUATION OF HSMKONTR JOB
//******************************************************************************
//***** CHANGE userid TO YOUR USERID <<<=======================================
//******************************************************************************
//**** DELETION OF WORK DATASETS
//******************************************************************************
//DELWORK EXEC PGM=IDCAMS,REGION=ØM
//SYSPRINT DD SYSOUT=X
//SYSIN DD *
//   DELETE userid.HSM.#TAPE.LIST
//   SET MAXCC=Ø
//*/
//******************************************************************************
//**** Extracting the information from DFSMShsm tapes
//****** Serial numbers of tapes are placed from column 1 to column 6
//****** Columns 7-9 contains marker with the following values:
//****** 'HM ' for ML2 tapes
//****** 'HCM' for copy of ML2 tapes
//****** 'HB ' for BACKUP tapes
//****** 'HCB' for copy of BACKUP tapes
//****** 'HD ' for DUMP tapes
//******************************************************************************
//EXTRACTV EXEC PGM=ICETOOL,REGION=1M
//TOOLMSG DD SYSOUT=X
//DFSMSG DD SYSOUT=X
//HSMMIG DD DSN=userid.HSM.#MIG.LIST,DISP=SHR
//HSMBKP DD DSN=userid.HSM.#BKP.LIST,DISP=SHR
//HSMDMP DD DSN=userid.HSM.#DMP.LIST,DISP=SHR
//OUT DD DSN=userid.HSM.#TAPE.LIST,DISP=(MOD,CATLG,DELETE),
//      UNIT=SYSDA,DCB=(RECFM=FB,LRECL=9),
//      SPACE=(TRK,(5,5),RLSE)
//TOOLIN DD *
//   COPY FROM(HSMMIG) TO(OUT) USING(SMIG)
//   COPY FROM(HSMMIG) TO(OUT) USING(CMIG)
//   COPY FROM(HSMBKP) TO(OUT) USING(SBKP)
//   COPY FROM(HSMBKP) TO(OUT) USING(CBKP)
//   COPY FROM(HSMDMP) TO(OUT) USING(SDMP)
//*
//**** EXTRACTING MIG VOLUMES
//SMIGCNTL DD *
//   INCLUDE COND=(12,4,EQ,C'348Ø'),FORMAT=CH
//   OUTREC FIELDS=(2,6,C'HM ')
//**** EXTRACTING COPY OF MIG VOLUMES
//CMIGCNTL DD *
//   INCLUDE COND=(12,4,EQ,C'348Ø',AND,115,6,NE,C'*NONE*'),FORMAT=CH
//   OUTREC FIELDS=(115,6,C'HCM')
/*
//*** EXTRACTING BACKUP VOLUMES
//SBKPCNTL DD *
//   INCLUDE COND=(11,4,EQ,C'3590'),FORMAT=CH
//   OUTREC FIELDS=(2,6,C'HB ')
//*** EXTRACTING COPY OF BACKUP VOLUMES
//CBKPCNTL DD *
//   INCLUDE COND=(11,4,EQ,C'3590',AND,108.6,NE,C'*NONE*'),FORMAT=CH
//   OUTREC FIELDS=(108,6,C'HCB')
//*** EXTRACTING DUMP VOLUMES
//SDMPCNTL DD *
//   INCLUDE COND=(16,4,EQ,C'3590'),FORMAT=CH
//   OUTREC FIELDS=(2,6,C'HD ')
/*
//************************************************************
//***  SORT DFHSM TAPES
//************************************************************
//SORTC EXEC PGM=ICEMAN
//SYSPRINT DD SYSOUT=X
//SYSOUT DD SYSOUT=X
//SORTIN DD UNIT=SYSDA,DISP=SHR,DSN=userid.HSM.#TAPE.LIST
//SORTOUT DD UNIT=SYSDA,DISP=SHR,DSN=userid.HSM.#TAPE.LIST
//SYSIN DD *
// SORT FIELDS=(1,6,A),FORMAT=CH,WORK=1
END
/*
//************************************************************
//***  COMPARISON OF RACF AND HSM
//************************************************************
//HSMRACF EXEC JOB,N=HSMRACF,D='userid.DFHSM.CNTL'
/*
//************************************************************
//***  COMPARISON OF RMM AND HSM
//************************************************************
//HSMRMM EXEC JOB,N=HSMRMM,D='userid.DFHSM.CNTL'
/*
//************************************************************
//***  COMPARISON OF ZARA AND HSM
//************************************************************
//HSMZARA EXEC JOB,N=HSMZARA,D='userid.DFHSM.CNTL'

JOB HSMRACF
//useridR JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID,MSGLEVEL=(2,0)
//************************************************************
//*** Job compares active tapes in DFSMShsm with the contents of the
//**** HSMHSM profile. If it notices a discrepancy it will update the
//**** RACF database to add the tapes that HSM really occupies and
//**** delete the tapes which do not have HSM datasets.
//*** Note:
//*** Continuation of job HSMEXTR

****** CHANGE userid TO YOUR USERID <===================

//*** Deletion of work datasets

//DELWORK EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *

DELETE 'userid.HSM.#RACF.HSMHSM.LIST'
DELETE 'userid.HSM.#RACF.TAPE.LIST'
DELETE 'userid.HSM.#RACF.DIFR.LIST'
DELETE 'userid.HSM.#RACF.RACFUPD.LIST'
SET MAXCC = Ø

/*/ 

//*** Lists RACF profile HSMHSM in TAPEVOL class

//LISTRHSM EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø
//SYSTERM DD SYSOUT=X  
//SYSPRINT DD SYSOUT=X
//SYSTSPRT DD DSN=userid.HSM.#RACF.HSMHSM.LIST,DISP=(NEW,CATLG),
//          DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=616Ø),
//          UNIT=SYSDA,SPACE=(TRK,(3Ø,3Ø),RLSE)
//SYSTSIN DD *

RL TAPEVOL HSMHSM

/*/ 

//*** Extracts all HSM tapes from RACF report

//EXTRACTV EXEC PGM=IKJEFT1A,DYNAMNBR=5Ø,COND=(7,LE),
//          REGION=ØM,PARM=('HSMRACF')
//SYSPROC DD DSN=userid.USER.CLIST,DISP=SHR
//SYSTSPRT DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*  
//SYSTSIN DD DUMMY
//FIN DD DSN=userid.HSM.#RACF.HSMHSM.LIST,DISP=SHR
//FOUT DD DSN=userid.HSM.#RACF.TAPE.LIST,DISP=(NEW,CATLG),
//       DCB=(RECFM=FB,LRECL=9,BLKSIZE=),
//       SPACE=(TRK,(5,1),RLSE),UNIT=SYSDA

/*/ 

//*** SORT OF DFSMSHSM TAPES EXTRACTED FROM THE RACF REPORT

//SORTC EXEC PGM=ICEMAN
//SYSPRINT DD SYSOUT=X
//SYSOUT DD SYSOUT=X  
//SORTIN DD UNIT=SYSDA,DISP=SHR,DSN=userid.HSM.#RACF.TAPE.LIST
//SORTOUT DD UNIT=SYSDA,DISP=SHR,DSN=userid.HSM.#RACF.TAPE.LIST
//SYSIN     DD *
SORT FIELDS=(1,6,A),FORMAT=CH,WORK=1
END
/

******************************************************************************
//*** Finding the differences between DFSMshsm and RACF
******************************************************************************
//MKEDIF EXEC PGM=ICETOOL,REGION=1M
//TOOLMSG DD SYSSOUT=X
//DFSMGS DD SYSSOUT=X
//IN DD UNIT=SYSDA,DISP=SHR,DSN=userid.HSM.#RACF.TAPE.LIST
//OUT DD DSN=userid.HSM.#RACF.DIFR.LIST,DISP=(NEW,CATLG),
//      DCB=(RECFM=FB,LRECL=9,BLKSIZE=),
//      SPACE=(TRK,(5,1),RLSE),UNIT=SYSDA
//TOOLIN DD *
SELECT FROM(IN) ON(1,6,CH) TO(OUT) NODUPS
/

******************************************************************************
//*** Generating of statements for update of RACF profile HSMHSM
******************************************************************************
//ICETOOL EXEC PGM=ICETOOL,REGION=1M
//TOOLMSG DD SYSSOUT=X
//DFSMGS DD SYSSOUT=X
//HSMRACF DD DSN=userid.HSM.#RACF.DIFR.LIST,DISP=SHR
//OUT DD DSN=userid.HSM.#RACF.RACFUPD.LIST,DISP=(MOD,CATLG),
//  DCB=(RECFM=FB,LRECL=80),
//  SPACE=(TRK,(5,5),RLSE),
//TOOLIN DD *
COPY FROM(HSMRACF) TO(OUT) USING(RACF)
COPY FROM(HSMRACF) TO(OUT) USING(SHSM)
/

******************************************************************************
//*** Update of HSMHSM profile
******************************************************************************
//TSOBATCH EXEC PGM=IKJEFT01,DYNAMNBR=30
//SYSTERM DD SYSSOUT=X
//SYSPRINT DD SYSSOUT=X
//SYSTSPRT DD SYSSOUT=X
//SYSTSIN DD DSN=userid.HSM.#RACF.RACFUPD.LIST,DISP=SHR

JOB HSMRMM

//useridT JOB MSGCLASS=X,MSGLEVEL=(2,1),NOTIFY=&SYSUID,CLASS=A

//*****************************************************
//*** Job compares DFSMShsm information with the information
//*** from DFSMSrmm. When it notices differences it will generate
//*** statements for elimination of discrepancy if it is possible.
//***
//*** The possible results of comparison are:
//*** In RMM | Belongs to HSM | In HSM | Action
//*** ACT | Y | ACT | OK
//*** ACT | Y | NO | delete from RMM
//*** ACT | N | ACT | danger - tape overwritten
//*** ACT | N | NO | -
//*** SCR | ACT | activate in RMM
//*** SCR | NO | -
//*** Note:
//*** Continuation of HSMEXTR job
//*****************************************************
//***** CHANGE userid TO YOUR USERID <===============

//************************************************************
//*** Deletion of work datasets
//************************************************************
//DELWORK EXEC PGM=IDCAMS,REGION=ØM
//SYSPRINT DD SYSOUT=X
//SYSIN DD *
DELETEn userid.HSM.#RMM.SCR.LIST
DELETEn userid.HSM.#RMM.ACT.LIST
DELETEn userid.HSM.#RMM.SCRH.LIST
DELETEn userid.HSM.#RMM.ACTH.LIST
DELETEn userid.HSM.#RMM.ACTNH.LIST
DELETEn userid.HSM.#RMM.DIFA.LIST
DELETEn userid.HSM.#RMM.DIFS.LIST
DELETEn userid.HSM.#RMM.DIFW.LIST
DELETEn userid.HSM.#RMM.RMMUPD.LIST
DELETEn userid.HSM.#RMM.HSMUPD.LIST
DELETEn userid.HSM.#RMM.WORNING.LIST
SET MAXCC=Ø
/
//************************************************************
//*** Lists all DFSMSRMM tapes
//************************************************************
//TSOBATCH EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø
//SYSTERM DD SYSOUT=X
//SYSPRINT DD SYSOUT=X
//SYSTSPT DD DSN=userid.HSM.#RMM.ACT.LIST,DISP=(NEW,CATLG,DELETE),
//    SPACE=(TRK,(3Ø,3Ø),RLSE),DCB=(RECFM=FB,LRECL=133,BLKSIZE=Ø)
//SYSTSIN DD *
    RMM SEARCHDATASET OWNER(*) STATUS(PRIVATE) LIMIT(*)
/
//TSOBATCH EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø
//SYSTERM DD SYSOUT=X

© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
//SYSPRINT DD SYSOUT=X
//SYSTSPRT DD DSN=userid.HSM.#RMM.SCR.LIST,DISP=(NEW,CATLG,DELETE),
//         SPACE=(TRK,(30,30),RLSE),DCB=(RECFM=FB,LRECL=133,BLKSIZ=0)
//SYSTSIN DD *
//     RMM SEARCHVOLUME OWNER(*) STATUS(SCRATCH) LIMIT(*)
/*
******************************************************************************
//*** Extracting of RMM tapes with the following characteristics:
//*** In RMM    Belongs to HSM
//*** ACT       Y
//*** ACT       N
//*** SCR
******************************************************************************
//SELECTZH EXEC PGM=ICETOOL,REGION=1M
//TOOLMSG DD SYSOUT=X
//DFSMSG DD SYSOUT=X
//HSMZACT DD DSN=userid.HSM.#RMM.ACT.LIST,DISP=SHR
//HSMZSCR DD DSN=userid.HSM.#RMM.SCR.LIST,DISP=SHR
//ACTHSM DD DSN=userid.HSM.#RMM.ACTH.LIST,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,DCB=(RECFM=FB,LRECL=9,BLKSIZ=0),
//         SPACE=(TRK,(1,1),RLSE)
//ACTNOHSM DD DSN=userid.HSM.#RMM.ACTNH.LIST,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,DCB=(RECFM=FB,LRECL=9,BLKSIZ=0),
//         SPACE=(TRK,(1,1),RLSE)
//SCRHSM DD DSN=userid.HSM.#RMM.SCRH.LIST,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,DCB=(RECFM=FB,LRECL=9,BLKSIZ=0),
//         SPACE=(TRK,(1,1),RLSE)
//TOOLIN DD *
COPY FROM(HSMZACT) TO(ACTHSM) USING(ZACT)
COPY FROM(HSMZACT) TO(ACTNOHSM) USING(ZANH)
COPY FROM(HSMZSCR) TO(SCRHSM) USING(ZSCR)
/*
//ZACTCNTL DD *
INCLUDE COND=(1,12,EQ,C'HSM.HMIGTAPE',OR,
           1,12,EQ,C'HSM.BACKTAPE',OR,
           1,8,EQ,C'HSM.COPY',OR,
           1,7,EQ,C'HSM.DMP'),FORMAT=CH
OUTREC FIELDS=(46,6,C'AH ')
/*
//ZANHCNTL DD *
INCLUDE COND=(1,6,NE,C'READY ',AND,
          1,1,NE,C' ',AND,
          1,1,NE,C'-',AND,
          62,1,NE,C'C',AND,
          1,8,NE,C'EDG3012I',AND,
          1,4,NE,C'END ',AND,
          1,12,NE,C'HSM.HMIGTAPE',AND,
          1,12,NE,C'HSM.BACKTAPE',AND,
          1,8,NE,C'HSM.COPY',AND,
          1,7,NE,C'HSM.DMP'),FORMAT=CH
OUTREC FIELDS=(46,6,C'A ')
/*
**** Active tapes which do not belong to HSM are the candidates for SCRATCH

**** Generation of statements for resolving the following situation:

<table>
<thead>
<tr>
<th>In RMM</th>
<th>Belongs to HSM</th>
<th>In HSMU</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT</td>
<td>Y</td>
<td>NO</td>
<td>delete from RMM</td>
</tr>
</tbody>
</table>

**** Scratch tapes which is active in HSM are candidates for activation.

**** Generation of statements for resolving the following situation:

<table>
<thead>
<tr>
<th>In RMM</th>
<th>Belongs to HSM</th>
<th>In HSM</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCR</td>
<td>ACT</td>
<td></td>
<td>activate in RMM</td>
</tr>
</tbody>
</table>
OUTREC FIELDS=(C' RMM CV ',1,6,
        C' RELEASEACTION(SCRATCH) STATUS(MASTER)' ,28X)
/*
******************************************************************************
***** Overwritten tape
***** Generating of statements for resolving the following situation:
***** In RMM Belongs to HSM In HSM Action
***** ACT N ACT DANGER - tape is overwritten
******************************************************************************
//TESTWORN EXEC PGM=ICETOOL,REGION=1M
//TOOLMSG DD SYSOUT=X
//DFSMSG DD SYSOUT=X
//IN DD UNIT=SYSDA,DISP=SHR,DSN=userid.HSM.#RMM.ACTNH.LIST
// DD UNIT=SYSDA,DISP=SHR,DSN=userid.HSM.#TAPE.LIST
//DIFW DD DSN=userid.HSM.#RMM.DIFW.LIST,DISP=(NEW,CATLG),
// DCB=(RECFM=FB,LRECL=9,BLKSIZE=),
// SPACE=(TRK,(5,1),RLSE),UNIT=SYSDA
//WARNING DD DSN=userid.HSM.##RMM.WARNING.LIST,DISP=(NEW,CATLG),
// DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=),
// SPACE=(TRK,(5,1),RLSE),UNIT=SYSDA
//HSMUPD DD DSN=userid.HSM.##RMM.HSMUPD.LIST,DISP=(NEW,CATLG),
// DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=),
// SPACE=(TRK,(5,1),RLSE),UNIT=SYSDA
//TOOLIN DD *
SELECT FROM(IN) ON(1,6,CH) TO(DIFW) ALLDUPS
COPY FROM(DIFW) TO(WARNING) USING(ZWOR)
COPY FROM(DIFW) TO(HSMUPD) USING(ZWOD)
COPY FROM(DIFW) TO(HSMUPD) USING(ZWOB)
/*
//ZWORCNTL DD *
INCLUDE COND=(7,1,EQ,C'H'),FORMAT=CH
OUTREC FIELDS=(C' HSEND LIST TTOC('',1,6,
        C') ODS(userid.HSM.#RMM.TTOC.LIST)' ,29X)
/*
//ZWODCNTL DD *
INCLUDE COND=(7,2,EQ,C'HD'),FORMAT=CH
OUTREC FIELDS=(C' HSEND DELVOL ' ,1,6,C' DUMP(LASTCOPY,PURGE)' ,58X)
/*
//ZWOBCNTL DD *
INCLUDE COND=(7,2,EQ,C'HB'),FORMAT=CH
OUTREC FIELDS=(C' HSEND DELVOL ' ,1,6,C' BACKUP(PURGE)' ,65X)
/*
******************************************************************************
***** Updating of RMM database
******************************************************************************
//TSOBATCH EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø
//SYSTERM DD SYSOUT=X
//SYSPRINT DD SYSOUT=X
//SYSTSRT DD SYSOUT=X
//SYSTSIN DD DSN=userid.HSM.##RMM.RMMUPD.LIST,DISP=SHR
/*
******************************************************************************

//*** Updating of HSM database
//*********************************************************************
//TSOBATCH EXEC PGM=IKJEFT01,DYNAMNBR=30
//SYSTERM DD SYSOUT=X
//SYSPRINT DD SYSOUT=X
//SYSTSRT DD SYSOUT=X
//SYSTSIN DD DSN=userid.HSM.##RMM.HSMUPD.LIST,DISP=SHR

JOB HSMZARA

//useridZ JOB MSGCLASS=X,MSGLEVEL=(2,1),NOTIFY=&SYSUID,CLASS=A
//*********************************************************************
// Job compares DFSMShsm information with the information
// from ZARA. When it notices differences it will generate
// statements for th elimination of discrepancy, if it is possible.
//*********************************************************************
// The possible results of comparison are:
// In ZARA Belongs to HSM In HSM Action
// ACT Y ACT OK
// ACT Y NO delete from ZARA
// ACT N ACT danger - tape overwritten
// ACT N NO -
// SCR ACT activate in ZARA
// SCR NO -
// Note:
// Continuation of HSMEXTR job
//*********************************************************************
// CHANGE userid TO YOUR USERID <---------------
// Deletion of work datasets
//*********************************************************************
//DELWORK EXEC PGM=IDCAMS,REGION=0M
//SYSPRINT DD SYSOUT=X
//SYSIN DD *

DELETE userid.HSM.#ZARA.SCR.LIST
DELETE userid.HSM.#ZARA.ACT.LIST
DELETE userid.HSM.#ZARA.SCRH.LIST
DELETE userid.HSM.#ZARA.ACTH.LIST
DELETE userid.HSM.#ZARA.ACTNH.LIST
DELETE userid.HSM.#ZARA.DIFA.LIST
DELETE userid.HSM.#ZARA.DIFS.LIST
DELETE userid.HSM.#ZARA.DIFW.LIST
DELETE userid.HSM.#ZARA.ZARAUPD.LIST
DELETE userid.HSM.#ZARA.HSMUPD.LIST
DELETE userid.HSM.#ZARA.WORNING.LIST
SET MAXCC=0
/*
*********************************************************************
// LISTS ALL ZARA TAPES
*********************************************************************
//LISTZACT EXEC ZARAUTL
//SYSUDUMP DD *
//ZARAUTL.SYSPRINT DD DSN=userid.HSM.#ZARA.ACT.LIST,DISP=(NEW,CATLG),
//   UNIT=SYSDA,DCB=(RECFM=FB,LRECL=133,BLKSIZE=Ø),
//   SPACE=(TRK,(5,5),RLSE)
//SYSIN DD *
//  LIST ACTIVE $$ /
//LISTZSCR EXEC ZARAUTL
//SYSUDUMP DD *
//ZARAUTL.SYSPRINT DD DSN=userid.HSM.#ZARA.SCR.LIST,DISP=(NEW,CATLG),
//   UNIT=SYSDA,DCB=(RECFM=FB,LRECL=133,BLKSIZE=Ø),
//   SPACE=(TRK,(5,5),RLSE)
//SYSIN DD *
//  LIST SCRATCH $$ /

//*********************************************************************
//*** Extracting tapes with the following characteristics from ZARA:
//***   In ZARA   Belongs to HSM
//***    ACT Y
//***    SCR
//*********************************************************************
//SELECTZH EXEC PGM=ICETOOL,REGION=1M
//TOOLMSG DD SYSOUT=X
//DFSMSG DD SYSOUT=X
//HSMZACT DD DSN=userid.HSM.#ZARA.ACT.LIST,DISP=SHR
//HSMZSCR DD DSN=userid.HSM.#ZARA.SCR.LIST,DISP=SHR
//ACTHSM DD DSN=userid.HSM.#ZARA.ACTH.LIST,DISP=(NEW,CATLG),
//   UNIT=SYSDA,DCB=(RECFM=FB,LRECL=9,BLKSIZE=Ø),
//   SPACE=(TRK,(1,1),RLSE)
//ACTNOHSM DD DSN=userid.HSM.#ZARA.ACTNH.LIST,DISP=(NEW,CATLG),
//   UNIT=SYSDA,DCB=(RECFM=FB,LRECL=9,BLKSIZE=Ø),
//   SPACE=(TRK,(1,1),RLSE)
//SCRHSM DD DSN=userid.HSM.#ZARA.SCRH.LIST,DISP=(NEW,CATLG),
//   UNIT=SYSDA,DCB=(RECFM=FB,LRECL=9,BLKSIZE=Ø),
//   SPACE=(TRK,(1,1),RLSE)
//TOOLIN DD *
COPY FROM(HSMZACT) TO(ACTHSM) USING(ZACT)
COPY FROM(HSMZACT) TO(ACTNOHSM) USING(ZANH)
COPY FROM(HSMZSCR) TO(SCRHSM) USING(ZSCR)
*/
//ZACTCNTL DD *
INCLUDE COND=((9,12,EQ,C'HSM.HMIGTAPE',OR,
   9,12,EQ,C'HSM.BACKTAPE',OR,
   9,8,EQ,C'HSM COPY',OR,
   9,7,EQ,C'HSM.DMP'),AND,
   66,4,EQ,C'USER'),FORMAT=CH
OUTREC FIELDS=(2,6,C'AH ')
*/
//ZANHCNTL DD *
INCLUDE COND=(1,1,NE,C'1',AND,
2,1,NE,C' ',AND,
8,1,NE,C':',AND,
9,1,NE,C' ',AND,
9,12,NE,C'HSM.HMIGTAPE',AND,
9,12,NE,C'HSM.BACKTAPE',AND,
9,8,NE,C'HSM.COPY',AND,
9,7,NE,C'HSM.DMP'),FORMAT=CH
OUTREC FIELDS=(2,6,C'A  ')
/
/ZSCRCNTL DD *
  INCLUDE COND=(118,9,EQ,C'SCRATCHED'),FORMAT=CH
  OUTREC FIELDS=(4,6,C'SH ')
/
//*********************************************************************
//***  Active tapes which don't belong to HSM are candidates for scratch
//***  Generating of statements for resolving the following situation:
//***   In ZARA   Belong to HSM   In HSM              Action
//***    ACT           Y              NO            delete from ZARA
//*********************************************************************
//GENZSCR EXEC PGM=ICETOOL,REGION=1M
//TOOLMSG DD SYSOUT=X
//DFSMSG DD SYSOUT=X
//IN DD UNIT=SYSDA,DISP=SHR,DSN=userid.HSM.#ZARA.ACTH.LIST
// UNIT=SYSDA,DISP=SHR,DSN=userid.HSM.#TAPE.LIST
//ACTTOSCR DD DSN=userid.HSM.#ZARA.DIFS.LIST,DISP=(NEW,CATLG),
// DCB=(RECFM=FB,LRECL=9,BLKSIZE=),
// SPACE=(TRK,(5,1),RLSE),UNIT=SYSDA
//ZUPD DD DSN=userid.HSM.##ZARA.ZARAUPD.LIST,DISP=(NEW,CATLG),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=),
// SPACE=(TRK,(5,1),RLSE),UNIT=SYSDA
//TOOLIN DD *
 SELECT FROM(IN) ON(1,6,CH) TO(ACTTOSCR) NODUPS
 COPY FROM(ACTTOSCR) TO(ZUPD) USING(ZSCR)
/
//*********************************************************************
//***  Scratch tapes which belong to HSM are candidates for activation
//***  Generation of statements for resolving the following situation:
//***   In ZARA       Belongs to HSM   In HSMU         Action
//***    SCR                        ACT       activate in ZARA
//*********************************************************************
//GENZACT EXEC PGM=ICETOOL,REGION=1M
//TOOLMSG DD SYSOUT=X
//DFSMSG DD SYSOUT=X
//IN DD UNIT=SYSDA,DISP=SHR,DSN=userid.HSM.#ZARA.SCRH.LIST
// UNIT=SYSDA,DISP=SHR,DSN=userid.HSM.#TAPE.LIST
//SCRTOACT DD DSN=userid.HSM.#ZARA.DIFA.LIST,DISP=(NEW,CATLG),
// DCB=(RECFM=FB,LRECL=9,BLKSIZE=),
// SPACE=(TRK,(5,1),RLSE),UNIT=SYSDA
//ZUPD DD DSN=userid.HSM.##ZARA.ZARAUPD.LIST,DISP=MOD
//TOOLIN DD *
SELECT FROM(IN) ON(1,6,CH) TO(SCRTOACT) ALLDUPS
COPY FROM(SCRTOACT) TO(ZUPD) USING(ZACT)
COPY FROM(SCRTOACT) TO(ZUPD) USING(ZAC1)
/*
//ZACTCNTL DD *
INCLUDE COND=(7,2,EQ,C'SH'),FORMAT=CH
OUTREC FIELDS=(C'UPDVOL VOLSER=',1,6,C'VSTAT=A $$',47X)
/*
//ZAC1CNTL DD *
INCLUDE COND=(7,2,EQ,C'SH'),FORMAT=CH
OUTREC FIELDS=(C'UPDVOL VOLSER=',1,6,
C'EXPDT=USØØ2 FSEQ=1 FSTAT=A $$',28X)

//*********************************************************************
//*** Overwritten tape
//*** Generation of statements for resolving the following situation:
//*** In ZARA Belongs to HSM In HSM Action
//*** ACT N ACT DANGER - tape overwritten
//*********************************************************************
//TESTWORN EXEC PGM=ICETOOL,REGION=1M
//TOOLMSG DD SYSOUT=X
//DFSMSG DD SYSOUT=X
//IN DD UNIT=SYSDA,DISP=SHR,DSN=userid.HSM.#ZARA.ACTNH.LIST
// DD UNIT=SYSDA,DISP=SHR,DSN=userid.HSM.#TAPE.LIST
//DIFW DD DSN=userid.HSM.#ZARA.DIFW.LIST,DISP=(NEW,CATLG),
// DCB=(RECFM=FB,LRECL=9,BLKSIZE=),
// SPACE=(TRK,(5,1),RLSE),UNIT=SYSDA
//WARNING DD DSN=userid.HSM.##ZARA.WARNING.LIST,DISP=(NEW,CATLG),
// DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=),
// SPACE=(TRK,(5,1),RLSE),UNIT=SYSDA
//HSMUPD DD DSN=userid.HSM.##ZARA.HSMUPD.LIST,DISP=(NEW,CATLG),
// DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=),
// SPACE=(TRK,(5,1),RLSE),UNIT=SYSDA
//TOOLIN DD *
SELECT FROM(IN) ON(1,6,CH) TO(DIFW) ALLDUPS
COPY FROM(DIFW) TO(WARNING) USING(ZWOR)
COPY FROM(DIFW) TO(HSMUPD) USING(ZWOD)
COPY FROM(DIFW) TO(HSMUPD) USING(ZWOB)
/*
//ZWORCNTL DD *
INCLUDE COND=(7,1,EQ,C'H'),FORMAT=CH
OUTREC FIELDS=(C'HSEND LIST TTOC('',1,6,
C' ) ODS(userid.HSM.#ZARA.TTOC.LIST)',29X)
/*
//ZWODCNTL DD *
INCLUDE COND=(7,2,EQ,C'HD'),FORMAT=CH
OUTREC FIELDS=(C'HSEND DELVOL ' ,1,6,C' DUMP(LASTCOPY,PURGE)' ,58X)
/*
//ZWOBBCNTL DD *
INCLUDE COND=(7,2,EQ,C'HB'),FORMAT=CH
OUTREC FIELDS=(C' HSEND DELVOL ',1,6,C' BACKUP(PURGE)','65X)
/*
//***************************************************************
//** Update of ZARA database
//***************************************************************
//ZARAUUD  EXEC ZARAUUL
//SYSUDUMP DD *
//SYSIN    DD DSN=userid.HSM.##ZARA.ZARAUUD.LIST,DISP=(OLD)
/*
//***************************************************************
//** Update of HSM database (optional)
//***************************************************************
//HSMUUPD  EXEC PGM=IKJEFT01,DYNAMNBR=30
//SYSTERM   DD SYSOUT=X
//SYSPRINT  DD SYSOUT=X
//SYSTSPRT  DD SYSOUT=X
//SYSTSIN   DD DSN=userid.HSM.##ZARA.HSMUUPD.LIST,DISP=SHR
//

REXX HSMRACF

/** REXX *******************************************************
/** Select tape volid from RACF list
/** TRACE ?R */
/* EOF='NO'
   reci.Ø=Ø
   reco.Ø=Ø
"EXECIO Ø DISKR FIN (OPEN)"
"EXECIO Ø DISKW FOUT (OPEN)"
I=Ø
CALL GET_FIN
/* skip other information in RACF list */
DO WHILE(EOF='NO')
   if substr(reci.1,1,2Ø) = "OTHER VOLUMES IN SET"
      then leave;
      CALL GET_FIN
END
/* skip ———— in RACF list */
CALL GET_FIN
/* get first record with volumes */
CALL GET_FIN
/* select volumes from 4. columns */
/* until get blank rows */
DO WHILE(EOF='NO')
   j=Ø
   if substr(reci.1,1,6) -= ' '
then do
  j=j+1
  reco.j = substr(reci.1,1,6)!!"R"
end
else leave
if substr(reci.1,9,6) ¬= ' '
then do
  j=j+1
  reco.j = substr(reci.1,9,6)!!"R"
end
if substr(reci.1,17,6) ¬= ' '
then do
  j=j+1
  reco.j = substr(reci.1,17,6)!!"R"
end
if substr(reci.1,25,6) ¬= ' '
then do
  j=j+1
  reco.j = substr(reci.1,25,6)!!"R"
end
reco.Ø=j
i=i+j
"EXECIO "j" DISKW FOUT (STEM reco."
CALL GET_FIN
END
"EXECIO Ø DISKR FIN (FINIS)"
"EXECIO Ø DISKW FOUT (FINIS)"
SAY " selected " i "tapes"
EXIT
GET_FIN: PROCEDURE EXPOSE RECI. EOF
"EXECIO 1 DISKR FIN (STEM RECI.)"
IF RC>=2
  THEN EOF='YES'
RETURN

---

Emina Spesic and Dragon Nikolic
Systems Programmers © Xephon 2000

---

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 e-mailed to them.
INTRODUCTION
It would be highly beneficial for many enterprises to condense unchangeable files, such as historical data format. This would:

• Reduced off-site costs.
• Reduced future tape purchases.
• Maximize Storage Tek’s Redwood/9840 and IBM Magstar tape utilization.
• Increase robotic set-up, etc.

DESCRIPTION
The program gets an input file (DDNAME SYS030), including a tape file list, allocates each file on the list dynamically (DDNAME SYS010) specifying volume it also allocates an output tape file (DDNAME SYS020) with the same name as input; copies the files using the Stack technique (multi-files), recatalogues the files with file sequence, and logs all information (DDNAME SYS050) of the copied files.

Restrictions
It cannot copy DFDSS dumped/copied files, nor multi-volume files.

Characteristics
The program can copy single/multi-file and single volumes. The input allocation happens on each new input volume or when missing a file sequence of the series (for example if you are copying the first, missing the second, and copying the third file on the tape). It works with copied concatenation files.

It works utilizing a chained scheduling technique, then it uses REGION=0K EXEC parameters.
If the input file is not catalogued or the volume in the MVS catalog is not the same on the SYS030 file, the file recatalogue does not occur.

Even though DCB attributes are not specified, this program accepts the label DCB attributes and copies them to the output file. It works with SWA above 16MB.

The program will abend if a datacheck occurs on the output file and after all datachecks in input files. Anyway, the log (SYS050) will show all files on which datacheck occurs.

The other attributes are copied to output files, including creation date/hour, expiration date, from SYS030. SYS030 lay-out is shown in Figure 1:

<table>
<thead>
<tr>
<th>Position</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>001-006</td>
<td>X(06)</td>
<td>volume serial number (VOLSER)</td>
</tr>
<tr>
<td>007-007</td>
<td>X(01)</td>
<td>reserved</td>
</tr>
<tr>
<td>008-051</td>
<td>X(44)</td>
<td>absolute dataset name (DSNAME)</td>
</tr>
<tr>
<td>052-053</td>
<td>X(02)</td>
<td>data set organization (DSORG) – optional</td>
</tr>
<tr>
<td>054-056</td>
<td>X(03)</td>
<td>record format (RECFM) – optional</td>
</tr>
<tr>
<td>057-061</td>
<td>9(05)</td>
<td>block size (BLKSIZE) – optional</td>
</tr>
<tr>
<td>062-066</td>
<td>9(05)</td>
<td>logical record length (LRECL) – optional</td>
</tr>
<tr>
<td>067-072</td>
<td>X(06)</td>
<td>reserved</td>
</tr>
<tr>
<td>073-079</td>
<td>9(07)</td>
<td>creation date (AAAADDD) – optional</td>
</tr>
<tr>
<td>080-080</td>
<td>X(01)</td>
<td>reserved</td>
</tr>
<tr>
<td>081-087</td>
<td>9(07)</td>
<td>expiration date (AAAADDD) – optional</td>
</tr>
<tr>
<td>088-092</td>
<td>9(05)</td>
<td>input file sequence number (FILSEQ)</td>
</tr>
<tr>
<td>093-098</td>
<td>X(06)</td>
<td>creation hour (HHMMSS) – optional</td>
</tr>
<tr>
<td>099-100</td>
<td>X(02)</td>
<td>reserved</td>
</tr>
</tbody>
</table>

*Figure 1: SYS030 lay-out*

NA08

PRINT NOGEN
TITLE 'NA08 - COPIES TAPE TO TAPE, USING STACK TECHNIQUE'
NA08 PXENTRY RENT=NO,BASES=(13,12) * YOUR LINKAGE CONVENTION
THE PROGRAM GETS AN INPUT FILE (DDNAME SYSØ3Ø) INCLUDING A TAPE FILE
LIST, ALLOCATES EACH FILE ON THE LIST DYNAMICALLY (DDNAME SYSØ1Ø)
SPECIFYING VOLUME IT ALSO ALLOCATES OUTPUT TAPE FILE (DDNAME SYSØ2Ø)
WITH THE SAME NAME AS INPUT, COPIES THE FILES, WITH STACK TECHNIQUE
(MULTI-FILES), RECATALOGS THE FILES, WITH FILE SEQUENCE AND LOGS ALL*
INFORMATION (DDNAME SYSØ5Ø) OF THE COPIED FILES.

LA    R13,Ø(R13)                   * COMPATIBILITY 31-BIT MODE
LA    R12,Ø(R12)                   *
OPEN  (SYSØ3Ø,,SYSØ5Ø,(OUTPUT))    *
LOOP3Ø   GET   SYSØ3Ø,WORKIN                * GET FILE LIST
          TIME  DEC                          * GET DATE/HOUR
          ST   R1,DATATIME                  *
          ST   RØ,HORATIME                  *
          MVC   RETCODE,=F'Ø'                *
          BAL   R1Ø,MONTALIN                 * ALLOCATE INPUT TAPE
          CLC   RETCODE,=F'Ø'                * SUCCESSFUL?
          BNE   LOOP3Ø                       * NO, PROCESS NEXT FILE
          BAL   R1Ø,MONTALOU                 * ALLOCATE OUTPUT TAPE
          CLC   RETCODE,=F'Ø'                * SUCCESSFUL?
          BNE   LOOP3Ø                       * NO, PROCESS NEXT FILE
          BAL   R1Ø,GRAVAFIT                 * COPY FILE
          CLC   RETCODE,=F'Ø'                * DATACHEK IN OUTPUT?
          BZ    LOOP3Ø                       * NO, PROCESS NEXT FILE
          FIM    CLOSE (SYSØ3Ø,,SYSØ5Ø)         *
          L     R1,COUNT1                    *
          LTR   R1,R1                        *
          BZ    TERMNOR                      *
          WTO ' #NAØ8.9Ø1C PROBLEMS WITH INPUT FILES',ROUTCDE=11
          ABEND 4ØØØ,DUMP                    *
          TERMNOR  WTO ' #NAØ8.999I SUCCESSFUL END',ROUTCDE=11
          PXEXIT RENT=NO,RC=Ø                * YOUR EXIT CONVENTION
          EJECT

ALLOCATE DYNAMICALLY INPUT FILE

MONTALIN ST   R1Ø,SAVMTIN            *
          USING INFØJFCB,R6                * CONVERT RECORD FORMAT
          XC    RECFMIN,RECFMIN              *
          CLI   RECIN,C'F'                  * RECFM = F?
          BNE   MNTVERV                     *
          OI    RECFMIN,B'10000000'         *
          B     MNTVERB                     *
          MNTVERV  CLI   RECIN,C'V'             * RECFM = V?
          BNE   MNTVERU                     *
          OI    RECFMIN,B'Ø1000000'         *
          B     MNTVERB                     *
          MNTVERU  OI    RECFMIN,B'11000000'         * ELSE RECFM = U?
          B     MNTVERB                     *
          MNTVERB  CLI   RECIN+1,C'B'               * BLOCKED?
BNE   MNTVERC
OI   RECFMIN,B'00010000'

MNTVERC CLI  RECFMIN,B'0000001000'
BNE   MNTVERM
OI   RECFMIN,B'0000001000'

MNTVERM CLI  RECFMIN,B'0000001000'
BNE   MNTPRTO
OI   RECFMIN,B'0000001000'

MNTPRTO MVC  DSNALIN,DSNIN
MVC  VOLALIN1,VOLIN1
CLC  VOLIN2,=CL6"'
BE   MNTMVALC
MVC  VOLPARM,=X'0002'
MVC  VOLALIN2,VOLIN2

MNTMVALC PACK  DOUBLE,LALOCIN
CVB  R1.DOUBLE
LH  R2,FILSEQ
LA  R2,1(R2)
CR  R1,R2
STH  R1,FILSEQ
BNE   MNTALOCØ
CLC  VOLIN1,VOLANT
BE   SEGALOC
MNTALOCØ CLC  COUNT,=F'0'
BE   MNTALOC1
CLOSE  DCB1Ø
FREEPOOL DCB1Ø
LA  R1,S99PRUN
DYNALLOC
LTR  R15,R15
BNZ   CANCEL

MNTALOC1 L  R1,COUNT
LA  R1,1(R1)
ST  R1,COUNT
MVC  VOLANT,VOLIN1
LA  R1,S99PRMIN
DYNALLOC
LTR  R15,R15
BZ    MNTLINEX
LH  R15,S99ARIN+4
LTR  R15,R15
BNZ   NAOALIN

MNTLINEX RDJFCB SYSØ1Ø
LA  R6,JFCBARIØ
MVC  JFCBDSNM,DSNALIN
MVC  DSNUNIN,DSNIN
MVC  JFCBFLSQ,FILSEQ
OI   JFCBLTYP,JFCDSEQN
MVC  JFCBNVOL,VOLPARM+1
MVC JFCBVOLS,VOLALIN1  
CLC VOLPARM,=H'1'  
BE MNTEXIT1  
MVC JFCBVOLS+6,VOLALIN2  
MNTEXIT1 L R10,SAVTIN  
BR R10  
SEGALOC CLOSE (DCB10,LEAVE)  
* CLOSE LAST FILE,  
FREEPOOL DCB10  
* KEEPING THE VOLUME SET  
B MNTLINEX  
NAOALIN MVC MESSLG,=CL62 'ERROR ALLOCRS= '  
MVC DSNLG,DSNALIN  
CVD R15,DOUBLE  
* FILE DO NOT COPIED  
UNPK MESSLG+14(4),DOUBLE  
* WRITE LOG  
OI MESSLG+17,X'F0'  
LA R3,DATATIME  
BAL R10,CONVIDATA  
MVC DATALG,DATA  
UNPK DOUBLE,HORATIME  
MVC HORALG,DOUBLE+1  
MVC DATACR,DATACIN  
MVC HORACR,HORACIN  
MVC MESSLG+19(6),CATVOL1  
MVC ERROFIT,=C'*'  
* ERROR INDICATOR  
PUT SYSØ5Ø,LOGOUT  
MVC RETCODE,=F'4'  
* NEXT FILE  
B MNTEXIT1  
EJECT  
* ALLOCATE DYNAMICALLY OUTPUT FILE  
*  
MONTALOU NOP MNTLOUSG  
* ALLOCATE ONCE OUTPUT DYNAMIC.  
USING IHADCB,R7  
* TO THE NEXT FILES, ONLY  
USING IOBSTD,R1  
* CHANGE JFCB AREA  
USING DECB,R2  
ST R10,SAVTOUT  
MVC DSNALOU,DSNIN  
* MOVE INPUT DSN TO OUTPUT  
LA R7,DCB10  
LA R6,JFCBARIØ  
MVC JFCRECFM,=B'00000000'  
*  
MVC JFCBLKSI,=H'Ø'  
* LRECL/BLKSIZE/RECFM  
MVC JFCLRECL,=H'Ø'  
* LIKE NO JCL - JFCB AREA  
MVC DCB10,SYMØ10  
MVC DCBLRECL,=H'Ø'  
* LRECL/BLKSIZE/RECFM  
MVC DCBBLKSI,=H'Ø'  
* LIKE NO JCL - DCB AREA  
MVC DCBRECFM,=B'00000000'  
* TO GET FROM LABEL  
OPEN (DCB10,(INPUT)),TYPE=J  
*  
RDJFCB SYSØ10  
LA R6,JFCBARIØ  
MVC RECFMIN,JFCRECFM  
* SAVE LRECL/BLKSIZE/RECFM  
MVC BLKALOU,JFCBLKSI  
* GET FROM LABEL INPUT
MVC LREALOU,JFCLRECL *
MVC RECALOU,RECFMIN *
BAL R10,TRANSFM * TRANSFORM RECFM IN TEXT
LH R1,JFCBLKSI *
LTR R1,R1 * NO BLKSIZE?
BZ DESPDSN * NEXT INPUT FILE
LH R1,JFCLRECL *
LTR R1,R1 * NO LRECL?
BZ DESPDSN * NEXT INPUT FILE
CLC RETENC,=CL7'00000000' * EXPDT = Ø?
BE ALOCAOUT * YES, BYPASS
MVC RETPD,RETENC * ELSE MOVE TO OUTPUT
MVI ALCNOP,X'ØØ' * SET ON EXPDT FLAG
ALOCAOUT LA R1,S99PRMOU *
DYNALLOC * ALLOCATE OUTPUT FILE
LTR R15,R15 *
BZ MNTLOPEN *
LA R15,S99AROU+4 *
LTR R15,R15 *
BNZ NAOALOU *
MNTLOPEN LA R7,DCB2Ø *
MVC DCB2Ø,SYSØ2Ø *
MVC DCBLRECL,LREALOU * REPLACE DCB ATTRIBUTES
MVC DCBBLKSI,BLKALOU * FROM INPUT FILE
MVC DCBBUFL,BLKALOU *
MVC DCBRECFM,RECFMIN *
OPEN (DCB2Ø,(OUTPUT)) *
RDJFCB SYSØ2Ø * FIND OUT 1. VOLUME MOUNT
LA R6,JFCBAR2Ø *
MVC CATVOL1,JFCBVOLS *
LA R15,Ø *
MVI MONTALOU+1,X'FØ' * TURN DOWN 2ND ALLOCATION
MNTLOUEX L R10,SAVMTOU *
BR R10 *
MNTLOUSG RDJFCB SYSØ2Ø * 2ND FILE AND NEXT
LA R6,JFCBAR2Ø *
LA R1,FLSEQA *
LTR R1,R1 *
BZ NAOQUEBR *
LA R1,1(R1) * NEXT FILSEQ
STH R1,JFCBFLSQ *
MVC FLSEQA,=H'Ø' *
B QUEBRFLS *
NAOQUEBR LH R1,JFCBFLSQ *
LA R1,1(R1) *
STH R1,JFCBFLSQ * CHANGE FILE SEQ JCL
QUEBRFLS 0I JFCBLTYP,JFCDSEQN *
MVC DSNALOU,DSNIN * MOVE DSN TO OUTPUT
LA R7,DCB1Ø *
MVC DCB1Ø,SYSØ1Ø *

MVC JFCBDSNM,DSNALOU * AND JCL TOO
LA R6,JFCBARI0 *
MVC JFCRECFM,=B'00000000' * LRECL/BLKSIZE/RECFM
MVC JFCBLKSI,=H'0' * LIKE NO JCL - JFCB AREA
MVC JFCLRECL,=H'0' *
MVC DCBLRECL,=H'0' * LRECL/BLKSIZE/RECFM
MVC DCBBLKSI,=H'0' * LIKE NO JCL - DCB AREA
MVC DCBRECFM,=B'00000000' * RECFM = U
OPEN (DCB10,(INPUT)),TYPE=J *
RDJFCB SYS010 *
LA R6,JFCBARI0 *
MVC RECFMIN,JFCRECFM * KEEP LRECL/BLKSIZE/RECFM
MVC BLKALOU,JFCBLKSI *
MVC LREALOU,JFCLRECL *
BAL R10,TRANSFM * TRANSFORM RECFM IN TEXT
LH R1,JFCBLKSI *
LTR R1,R1 *
BL R1,TRANSFM
LB R1,JFCBLKSI *
LTR R1,R1 *
LRETENC,=CL7'00000000' * EXPDT = Ø?
BE MNTALOUT *
PACK DOUBLE,LRETENC(4) *
CVB R1,DOUBLE *
S R1,=F'1900' * EXPDT RELATIVE TO 1900
PACK DOUBLE,LRETENC+4(3) *
CVB R0,DOUBLE *
SLL R1,16 *
OR R0,R0 *
STCM R1,7,JFCBXPDT *
MNTALOUT LA R7,DCB20 *
MVC DCB20,SYS020 *
MVC DCBLRECL,LREALOU * REPLACE DCB ATTRIBUTES
MVC DCBBLKSI,BLKLALOU *
MVC DCBBUFL,BLKLALOU *
MVC DCBRECFM,RECFMIN *
MVC JFCBLKSI,BLKLALOU *
MVC JFCLRECL,LREALOU *
MVC JFCRECFM,RECFMIN *
OPEN (DCB20,(OUTPUT)),TYPE=J *
LA R15,Ø *
B MNTLOUEX *
NAOALOU MVC MESSLG,=CL62'ERROR ALLOCRS=' *
MVC DSNLG,DSNALOU *
CVD R15,DOUBLE *
UNPK MESSLG+14(4),DOUBLE *
OI MESSLG+17,X'F0' *
LA R3,DATATIME *
BAL   R1Ø,CONVDATA           *
MVC   DATALG,DATA            *
UNPK  DOUBLE,HORATIME        *
MVC   HORALG,DOUBLE+1        *
MVC   DATACR,DATACIN         *
MVC   HORACR,HORACIN         *
MVC   MESSLG+19(6),CATVOL1   *
MVC   ERROFIT,=C'*'          *
PUT   SYSØ5Ø,LOGOUT         *
MVC   RETCODE,=F'4'          *
B     MNTLOUX               *

DESPDSN MVC   MESSLG,=CL62'INCOMPAT. FILE                  ' *
MVC   DSMLG,DSNIN           *
LA    R3,DATATIME           * WRITE LOG
BAL   R1Ø,CONVDATA          * INCOMPATIBLE
MVC   DATALG,DATA           * FILE
UNPK  DOUBLE,HORATIME       *
MVC   HORALG,DOUBLE+1       *
MVC   DATACR,DATACIN        *
MVC   HORACR,HORACIN        *
MVC   MESSLG+19(6),CATVOL1  *
MVC   ERROFIT,=C'*'         *
PUT   SYSØ5Ø,LOGOUT        *
MVC   RETCODE,=F'5'         * DESPREZA ARQUIVO
B     MNTLOUXE             *

CANCEL  WTO   '#NAØ8.101C ERROR DESALLOC INPUT FILE', X*
       ROUTCDE=11             *
       ABEND 4000,DUMP        *
       EJECT                 *

* ——————————————————————————————————————————————————————————————————————*
*  COPY TAPE TO TAPE, UTILIZING STACK                                 *
* ——————————————————————————————————————————————————————————————————————*

GRAVAFIT ST   R1Ø,SAVGRVF          *
LERIØ   LA    RØ,WK               * MVCL SPACES TO WK
       LA    R1,7                 *
       SLL   R1,12               *
       LA    R1,4Ø95(R1)         *
       LR    R2,RØ               *
       LA    R3,=A(40000000)     *
       MVCL  RØ,R2               *
       GET   DCB1Ø,WK             * READ INPUT FILE
       PUT   DCB2Ø,WK             * WRITE OUTPUT FILE
       B     LERIØ               *

ERSYSØ1Ø EQU   *                     * I/O ERROR SYSØ1Ø
       B     GLOGIOIN           *

ERSYSØ2Ø WTO   '#NAØ8.902E I/O ERROR SYSØ2Ø',ROUTCDE=11 *
       CLOSE DCB2Ø              *
       FREEPOOL DCB2Ø           *
       MVC   RETCODE,=F'4'      *
       B     SAIGFITA            *

FIM1          RDJFCB SYS020    * INPUT EOF
LA             R6,JFCBARR0   *
CLC            VOLSEQ,VOLSEQA  * CHANGED VOLSEQ?
BNE            QUEBRAVS      * YES, CHECKA DSN JFCB CATALOG

ATUADSN        MVC CATDSN,JFCBDSNM  *
L              R1,VOLSEQ     *
MVC            CATVOL2,=CL6' '  *
MVC            CATFSQ1,JFCBFLSQ  *
C              R1,=F'5'     *
BH              CATALOGA     *
BCTR           R1,0          *
SR             R0,R0        * DESLOC. JFCBVOLS
M              R0,=F'6'     *
LA             R1,JFCBVOLS(R1)  *
MVC            CATVOL1,0(R1)  *
B              CATALOGA     *
QUEBRAVS       MVC CATDSN,JFCBDSNM  *
L              R1,VOLSEQA    *
BCTR           R1,0          *
C              R1,=F'3'     * MORE THAN FIVE VOLUMES?
BH              TRATAMAI     *
SR             R0,R0        * DESLOC. JFCBVOLS
M              R0,=F'6'     *
LA             R1,JFCBVOLS(R1)  *
MVC            CATVOL1,0(R1)  *
MVC            CATFSQ1,JFCBFLSQ  *
MVC            CATVOL2,6(R1)  *
LH             R1,JFCBFLSQ   *
STH            R1,CATFSQ2   *
STH            R1,FLSEQA    *
STH            R1,JFCBFLSQ   *
CATALOGA       LA             R1,PARMCATL    * RECATALOG OUTPUT FILE
BAL             R10,CATL     *
MVC            VOLSEQA,VOLSEQ  *
MVC            MESSLG,=CL62'ARQ.GRAV.FS= V=    *
MVC            MESSLG+32(30),=C'LRECL=XXXXX,BLKSIZE=XXXXX,    *
MVC            MESSLG+19(6),CATVOL1  *
MVC            MESSLG+26(6),CATVOL2  *
LH             R1,JFCBLRECL   *
CVD             R1,DOUBLE    *
UNPK           MESSLG+38(5),DOUBLE+5(3) *
OI              MESSLG+42,X'F0'  *
LH             R1,JFCBLKSI   *
CVD             R1,DOUBLE    *
MVC            MESSLG+58(3),RECFM  *
UNPK           MESSLG+52(5),DOUBLE+5(3) *
OI              MESSLG+56,X'F0'  *
LH             R1,JFCBFLSQ   *
CVD             R1,DOUBLE    *
UNPK           MESSLG+12(5),DOUBLE+5(3) *

© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
OI MESSLG+16,X'F0' *
LA R3,DATATIME *
BAL R10,CONVDATA *
MVC DATALOG,DATA *
UNPK DOUBLE,HORATIME *
MVC HORALG,DOUBLE+1 *
MVC DSNLG,JFCBDSNM *
MVC DATACR,DATACIN *
MVC HORACR,HORACIN *
MVC ERROFIT,=C' ' *
MVC VOLUANT,VOLIN1 * VOLUME LAST
PUT SYSO50,LOGOUT * WRITE LOG INFORMATION
CLC CATVOL2,=CL6' ' * CHANGED VOLUME ?
BE SAIGRAVF *
MVC CATVOL1,CATVOL2 *
SAIGRAVF CLOSE (DCB20,LEAVE) * KEEP TAPE SET
FREEPOOL DCB20 *
SAIGFITA L R10,SAVGRVF *
BR R10 *
TRATAMAI C R1,=F'4' * MORE THAN 5 VOLUMES
BH MAISQ6 *
LA R1,JFCBVOLS+24 * DESLOCAMENTO JFCBVOLS
MVC CATVOL1,Ø(R1) *
MVC CATFSQ1,JFCBFSQ *
AMODE311 @AMODE 31 * 31-BIT MODE
LA R2,SWA_EPA * SWA EPA
DROP R2 *
USING SWAEPAR2 *
XC SWAPEA,SWAEPAR2 *
MVC SWVA(3),JFCBEXAD *
SWAREQ FCODE=RL,* READ/LOCATE REQUEST X
  EPA=SWEPAPTR,* ENTRY PARAMETER LIST X
  MF=(E,SWAPARMS),*
  UNAUTH=YES *
L R2,SWBLKPTR *
LA R1,4(R2) * DESLOC. VOLS JFCBX
MVC CATVOL2,Ø(R1) *
AMODE241 @AMODE 24 * 24-BIT MODE
DROP R2 *
USING DECB,R2 *
LH R1,JFCBFLSQ *
STH R1,CATFSQ2 *
STH R1,FLSEQA *
STH R1,JFCBFLSQ *
B CATALOGA *
MAISQ6 C R1,=F'198' *
BH MAISQ2ØØ * MORE THAN 2ØØ VOLUMES
S R1,=F'4' *
LR R9,R1 *
AMODE312 @AMODE 31 * 31-BIT MODE
LA R2,SWA_EPA                      * SWA EPA
DROP R2
USING SWAEPA,R2
XC SWAEPA,SWAEPA                       * CLEAR AREA
MVC SWVA(3),JFCBXAD                   * MOVE TOKEN JFCB
SWAREQ FCODE=RL,
   EPA=SWEPAPTR,
   MF=(E,SWAPARMS),
   UNAUTH=YES
L R2,SWBLKPTR                         * ADDRESS JFCB
SR R8,R8
D R8,=F'15'
LTR R9,R9
BZ MOVVOLX2
PROXJFX BCT R9,POSIJFX
   LTR R8,R8                      * FIRST JFCBX
   BZ MOVVOLX1
   ICM R3,7,Ø(R2)
   LA R2,SWA_EPA                   * SWA EPA
   XC SWAEPA,SWAEPA                 * CLEAR AREA
   STCM R3,7,SWA                   * MOVE TOKEN JFCB
   SWAREQ FCODE=RL,
      EPA=SWEPAPTR,
      MF=(E,SWAPARMS),
      UNAUTH=YES
   L R2,SWBLKPTR                     * ADDRESS JFCB
   B PROXJFX
POSIJFX ICM R3,7,Ø(R2)
   LA R2,SWA_EPA                    * SWA EPA
   XC SWAEPA,SWAEPA                 * CLEAR AREA
   STCM R3,7,SWA                    * MOVE TOKEN JFCB
   SWAREQ FCODE=RL,
      EPA=SWEPAPTR,
      MF=(E,SWAPARMS),
      UNAUTH=YES
   L R2,SWBLKPTR                  * ENDERECA JFCB
   B PROXJFX
MOVVOLX1 MVC CATVOL1,88(R2)           * MOVE LAST VOL JFCB
   ICM R3,7,Ø(R2)
   LA R2,SWA_EPA                   * SWA EPA
   XC SWAEPA,SWAEPA                * CLEAR AREA
   STCM R3,7,SWA                   * MOVE TOKEN JFCB
   SWAREQ FCODE=RL,
      EPA=SWEPAPTR,
      MF=(E,SWAPARMS),
      UNAUTH=YES
   L R2,SWBLKPTR                     * ADDRESS JFCB
   MVC CATVOL2,4(R2)
   B PROXJF CX
MOVVOLX2 LR R9,R8

© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
BCTR R9,0
SR R8,R8
M R8,=F'6'
LA R1,4(R9,R2)
MVC CATVOL1,Ø(R1)
MVC CATFSQ1,JFCBFLSQ
MVC CATVOL2,6(R1)

PROXJFCX LH R1,JFCBFLSQ
STH R1,CATFSQ2
STH R1,FLSEQA
STH R1,JFCBFLSQ

AMODE24 @AMODE 24
* 24-BIT MODE
DROP R2
USING DECB,R2
B CATALOGA

MAISQ200 WTO '#NAØ8.1Ø2C MORE THAN 200 VOLUME OUTPUT TAPE', ROUTCDE=11
B FIM

GLOGIOIN MVC MESSLG,=CL62'I/O INPUT ERROR
MVC DSNLG,DSNALIN
LA R3,DATATIME
BAL R1Ø,CONVDATA
MVC DATALG,DATATIME
MVC DATAcr,DATAcin
MVC MESSLG+19(6),CATVOL1
MVC ERROFIT,=C'*'
MVC VOLUANT,VOLIN1
PUT SYSØ5Ø,LOGOUT
L R1,COUNT1
LA R1,i(R1)
ST R1,COUNT1
B SAIGRAVF
EJECT

* CONVERT TODAY DATE (ØCAADDD)-COMP-3 TO EBCDIC (AAAAMMDD) *

CONVDATA ST R10,SAVCNV
MVC TABMES(26),TABMES1
UNPK DOUBLE,Ø(4,R3)
OI DOUBLE+7,X'3Ø'
TRT DOUBLE,TABNUM
LA R15,4
BNZ CONVDEXT
SR R1,R1
IC R1,ø(R3)
MH R1,=H'100'
SR RØ,RØ

IC  R0,1(R3)  *  
ZAP  DOUBLE,=P'Ø'  *  
SLL  R0,4  *  
ST  R0,DOUBLE+4  *  
OI  DOUBLE+7,X'ØF'  *  
CVB  R0,DOUBLE  *  
A  R0,=F'1900'  *  
AR  R1,R0  *  ADD YEAR  
SR  R0,R0  *  
ST  R1,ANO  *  
D  R0,=F'100'  *  DIVISIBLE 100?  
C  R0,=F'0'  *  
BE  CENTURY  *  YES, TREAT CENTURY  
L  R1,ANO  *  
SR  R0,R0  *  
D  R0,=F'4'  *  BICEXTO?  
C  R0,=F'0'  *  
BE  BICEXTO  *  YES, TREAT YEAR BICEXTO  
B  DESCMES  *  
CENTURY  L  R1,ANO  *  CENTURY YEAR BICEXTO  
SR  R0,R0  *  HAVE TO DIVISIBLE 400  
D  R0,=F'400'  *  
C  R0,=F'0'  *  
BE  BICEXTO  *  YES  
B  DESCMES  *  
BICEXTO  LA  R1,TABMES+24  *  TREAT BICEXTO  
LA  R0,11  *  
LOOPCV1  AP  Ø(2,R1),=P'1'  *  
BCTR  R1,0  *  
BCTR  R1,0  *  
BCT  R0,LOOPCV1  *  
DESCMES  LA  R1,TABMES+24  *  FIND OUT MONTH  
LA  R0,13  *  SEARCH LAST DAY  
LOOPCV2  CP  2(2,R3),Ø(2,R1)  *  
BH  ACHOUCV  *  
BCTR  R1,0  *  
BCTR  R1,0  *  
BCT  R0,LOOPCV2  *  
LA  R15,4  *  
B  CONVDEXT  *  
ACHOUVC  ZAP  DOUBLE,2(2,R3)  *  FIND OUT DAY  
SP  DOUBLE,Ø(2,R1)  *  
UNPK  DIA,DOUBLE  *  
OI  DIA+1,X'F0'  *  
C  R0,=F'12'  *  
LA  R15,4  *  
BH  CONVDEXT  *  
CVD  R0,DOUBLE  *  
UNPK  MES,DOUBLE  *  CONVERT MONTH (REG Ø)  
OI  MES+1,X'F0'  *
L R1,ANO  *
CVD R1,DOUBLE  * CONVERT YEAR
UNPK ANO,DOUBLE  *
OI ANO+3,X'FØ'  *
LA R15,Ø  *
CONVDEXT L R1Ø,SAVCNVD
BR R1Ø
EJECT

* RECATALOG DSNAME IF IT'S ALREADY CATALOGUED WITH SAME INPUT VOLUME *

CATL  STM R14,R12,SAVCATL  *
L R1,Ø(R1)  *
LA R2,0  *
ST R2,VOLCOUNT  * ZERO TO VOLCOUNT
LA R5,44(R1)  *
LA R3,VOL+2  *
LOOPVL  CLC Ø(6,R5),=CL6' '  * LAST VOLUME
BE FIMVOL  *
L R2,VOLCOUNT  *
LA R2,1(R2)  *
ST R2,VOLCOUNT  *
MVC 4(6,R3),Ø(R5)  * MOVE VOL-SER
MVC Ø(4,R3),=X'78048Ø81'  * DEVICE TYPE
MVC 1Ø(2,R3),6(R5)  * FSEQ
LA R5,8(R5)  *
LA R3,12(R3)  *
B LOOPVL  *
FIMVOL  L R2,VOLCOUNT  *
C R2,=F'1'  *
BL ERROVL  *
STH R2,VOL  *
LR R5,R1  *
DELM1  MVC DSNCAT(44),Ø(R5)  * PREPARE CAMLIST AREA
LOCATE CAMLSTS  *
LTR R5,R15  * IT IS ALREADY CATALOGUED?
BNZ RETURN  * NO, RETURN
CLC VOL1+6(6),VOLIN1  * INPUT VOLUME IS THE SAME?
BNE RETURN  * NO, RETURN
CATALOG CAMLSTD  * UNCATLG DSNAME
LTR R5,R15  * ERROR?
BNZ MENSAG1  * YES, RETURN
CATALOG CAMLSTC  * RECATALOG DSNAME
LTR R5,R15  * ERROR?
BNZ MENSAG2  * YES, SEND A MESSAGE AND ABEND
LA R15,Ø  *
RETURN  LM R14,R12,SAVCATL  *
BR R1Ø  *
MENSAG1  STM RØ,R1,SAVERØ1
WTO 'CATAL.9Ø1I ERROR UNCATLG',ROUTCDE=11
ABEND 4000,DUMP
MENSAG2 STM R0,R1,SAVER0R1
WTO 'CATAL.9011 ERROR CATLG',ROUTCDE=11
ABEND 4000,DUMP
ERROVL WTO 'CATAL.9011 EMPTY VOL-SER',ROUTCDE=11
ABEND 4000,DUMP
EJECT

* —————————————————————————————————————————————————————————————————————— *
*        TRANSFORM RECFM IN INTERNAL TEXT                             *
* —————————————————————————————————————————————————————————————————————— *
TRANSFM ST R10,SAVTRAFM *
TM RECFMIN,B'10000000' *
BNO CNTVERV *
MVI RECIN,C'F' *
B CNTVERB *
CNTVERV TM RECFMIN,B'01000000' *
BNO CNTVERU *
MVI RECIN,C'V' *
B CNTVERB *
CNTVERU MVI RECIN,C'U' *
THEN RECFM = U
CNTVERB TM RECFMIN,B'00010000' *
BNO CNTVERC *
MVI RECIN+1,C'B' *
BLOCKED?
CNTVERC TM RECFMIN,B'00001000' *
BNO CNTTRX *
MVI RECIN+2,C'A' *
ASA CONTROL ?
B CNTTRX *
CNTVERM TM RECFMIN,B'00000100' *
BNO CNTTRX *
MVI RECIN+2,C'M' *
MACHINE CONTROL ?
CNTTRX L R10,SAVTRAFM *
BR R10 *
EJECT

* —————————————————————————————————————————————————————————————————————— *
*        EXIT TAPE MOUNT CARTRIDGE/TAPE                               *
* —————————————————————————————————————————————————————————————————————— *
SCRTAPE STM R14,R12,SAVSTAP *
DROP R13,R12 *
DROP R2 *
USING SCRTAPE,R15 *
USING OENTID,R2 *
LR R2,R1 *
TM OENTFLG,OENTOE0V *
CHECK OPEN
BNO SCREXT *
YES, EXIT
L R1,VOLSEQ *
LA R1,1(R1) *
ADD 1 TO VOLSER
ST R1,VOLSEQ *
INALIZE VOLUME CHANGED
SCREXT LM R14,R12,SAVSTAP *
LA R15,0 *
RETURN
BR R14 *
DROP R15
EJECT

SAVERØR1 DS 2F * WORKING STORAGE
SAVMTIN DS F * SAVE AREAS
SAVMTOU DS F *
SAVCNVN DS F *
SAVGRVF DS F *
SAVCATL DS 15F *
SAVSTAP DS 15F *
SAVTRAFM DS F *

WORKIN DS ØCL100 * INPUT SYSØ3Ø - TAPE LIST
VOLIN1 DS CL6 *
  DS CL1 *
DSNIN DS CL44 *
DSOIN DS CL2 *
RE Cin DS CL3 *
BLKIN DS CL5 *
LREIN DS CL5 *
VOLIN2 DS CL6 *
DATA Cin DS CL7 *
  DS CL1 *
LRETENC DS CL7 *
LALOCIN DS CL5 *
HORACIN DS CL6 *
  DS CL2 *

RECFMIN DC XL1'Ø' *
VOLANT DC CL6' ' * LAST VOLUME
TABNUM DC 240X'FF' * TRANSLATE AND TEST
  DC 10X'00' * NUMERIC FIELDS
  DC 6X'FF' *

LOGOUT DS ØCL14Ø * LOG SYSØ5Ø
DATALG DS CL8 *
HORALG DS CL6 *
DSNLG DS CL44 *
MESSLG DS CL62 *
DATA CR DS CL7 *
HORACR DS CL6 *
ERROFIT DS CL1 *
VOLUANT DC CL6' ' *

*  CONVERSION AREAS TIME AND DATE  *

DATATIME DS F *
HORATIME DS F *
DATA DS ØCL8 *
ANO DS CL4 *
MES DS CL2 *
DIA DS CL2 *

TABMES DC PL2'0',PL2'31',PL2'59',PL2'90',PL2'120',PL2'151'
  DC PL2'181',PL2'212',PL2'243',PL2'273',PL2'304'
DC PL2'334',PL2'365',PL2'0',PL2'0'
TABMES1 DC PL2'Ø',PL2'31',PL2'59',PL2'90',PL2'120',PL2'151'
DC PL2'181',PL2'212',PL2'243',PL2'273',PL2'304'
DC PL2'334',PL2'365',PL2'0',PL2'0' * RECURSIVITY
DS 0F *
S99PRMIN DC X'80',AL3(S99ARIN) * DYNAMICALLY ALLOCATION SYS010
S99ARIN DC AL1(20) *
DC X'01' *
DC XL6'0' *
ALOCIADR DC A(ALLOCIN) *
DC XL8'0' *
ALLOCIN DC A(ALINDSP),A(ALINDDN),A(ALINNDS),A(ALININVOL)
DC A(ALINUNIT),A(ALINSEQ),AL1(128),AL3(ALINDSN)
ALINUNIT DC X'0015',X'0001',X'0004'
ALINUNIT DC CL4'ROBO' *
DS 0F *
ALINVOL DC X'0010' *
VOLPARAM DC X'0001',X'0006' *
VOLALIN1 DC CL6',X'0006' *
VOLALIN2 DC CL6' *
DS 0F *
ALINDSP DC X'0004',X'0001',X'0001'
DSPIN DC X'01' *
DS 0F *
ALINNDS DC X'0005',X'0001',X'0001'
NDSPIN DC X'08' *
DS 0F *
ALINDDN DC X'0001',X'0001',X'0008'
NDALIN DC CL8'SYS010' *
DS 0F *
ALINDSN DC X'0002',X'0001',X'002C'
DSNALIN DC CL44' ' *
DS 0F *
ALINSEQ DC X'001F',X'0001',X'0002'
FILSEQ DC X'0001' *
DS 0F * DYNAMICALLY ALLOCATION SYS020
S99PRMOU DC X'80',AL3(S99AROU) *
S99AROU DC AL1(20) *
DC X'01' *
DC XL6'0' *
ALOCOU DC A(ALCOU) *
DC XL8'0' *
ALCOU DC A(ALCOUDSP),A(ALCOUDDN),A(ALCOUDSN),A(ALCOUINVOL)
DC A(ALCOUCAT),A(ALCOUUNC),A(ALCOUREC),A(ALCOUER)
DC A(ALCOUBLK),A(ALCOUDSO),A(ALCOUSEQ)
ALOCNOP DC X'80',AL3(ALCOUCNT) *
DC X'80',AL3(ALCOUREP) *
DS 0F *
ALCOUDDN DC X'0001',X'0001',X'0008'
DDNALOU DC CL8'SYS020' *
* DCB AREAS - SYS010/SYS020 UTILIZING CHANNED SCHEDULING TECHNIQUE
SYS010 DCB DSORG=PS,MACRF=(GM),DDNAME=SYS010,EODAD=FIM1,
OPTCD=C, EXLST=RDJFCB10, SYNADE=ERSYS010, BUFNO=64
SYS020 DCB DSORG=PS, MACRF=(PM), DDNAME=SYS020, X
OPTCD=C, EXLST=RDJFCB20, SYNADE=ERSYS020, BUFNO=64
SYS030 DCB DSORG=PS, MACRF=GM, DDNAME=SYS030, EODAD=FIM, X
RECFM=FB, LRECL=100
SYS050 DCB DSORG=PS, MACRF=PM, DDNAME=SYS050, X
RECFM=FB, LRECL=140

   DS  OF

DCB10 DS CL100 * COPY SYS010 DCB
DCB20 DS CL100 * COPY SUS020 DCB
DS OF

CAMLSTC CAMLST CAT, DSNCAT,, LOCAREA * CATALOG MACRO
CAMLSTS CAMLST NAME, DSNCAT,, LOCAREA1 * SEARCH CATALOG MACRO
CAMLSTD CAMLST UCATDX, DSNCAT * UNCATALOG MACRO
DSNCAT DC CL44' '
LOCAREA DS 0D
VOL DS CL265
LOCAREA1 DS 0D
VOL1 DS CL265
DS OF

DOUBLE DS D * WORKING AREA
POINT DC F'0' *
COUNT1 DC F'0' *
COUNT DC F'0' *
VOLCOUNT DC F'0' *
PARMCATL DC AL1(128), AL3(PARMCAT) *
PARMCAT DS 0CL66 *
CATDSN DC CL44' '
CATVOL1 DC CL6' '
CATFSQ1 DC H'0'
CATVOL2 DC CL6' '
CATFSQ2 DC H'0', CL6' '
VOLSEQ DC F'1' *
VOLSEQA DC F'1' *
FLSEQA DC H'0' *
RETCODE DS F * FULL WORD
SWEPAPTR DC A(SWA_EPA) *
SWA_EPA DS XL28 * SWA AREA MACRO
SWAPARMS SWAREQ UNAUTH=YES, MF=L *
RDJFCB20 DS 0F *
DC X'17', AL3(SCRTAPE) * VOLUME MOUNT EXIT ROUTINE
DC X'87', AL3(JFCBAR20) * JFCB SYS020 ROUTINE
RDJFCB10 DS 0F *
DC X'87', AL3(JFCBAR10) * JFCB SYS010 ROUTINE
JFCBAR10 DC 176X'0' * WORK JCL AREA SYS010
JFCBAR20 DC 176X'0' * WORK JCL AREA SYS020

LTORG
WK DS CL32768 * BUFFER RECORD
DSECT
IEFJFCBN
Executing a PL/I program from REXX

INTRODUCTION
The following program allows users to execute a PL/I program, needing DD sysprint, from REXXX. It is called by a shell script, using the environment variable _bpx_batch_spawn.

The program requires OS/390 Version 2 Release 8 or higher, in addition to Unix, PL/I, and REXX.
SHRXPLSP JCL

//TSHVRC JOB (,'SHRXPLSP',TIME=1440,NOTIFY=&SYSUID, // REGION=0,CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),COND=(4,LT) //BPXBATCH SH -> SHELL SCRIPT -> REXX -> PL/1 PROGRAM -> SYSPRINT //IF YOU NEED TO EXECUTE PL/I PROGRAM,NEEDING DD SYSPRINT, //FROM REXX, CALLED BY SHELL SCRIPT, CALLED BY BPXBATCH //USE _BPX_BATCH_SPAWN //ENVIRONMENT VARIABLE _BPX_BATCH_SPAWN=YES (FROM V2R8 ONWARDS?) //TO EDIT THIS MEMBER:||CAPS OFF ||UNNUM ||NUMBER OFF //THE FOLLOWING COPIES SHELL SCRIPT,REXX EXEC TO HFS //COMPILES/LINKS SAMPLE PL/1 NEEDING SYSPRINT //EXECUTES BPXBATCH //shdir should be executable //COPY SHELL SCRIPT TO HFS //SHCOPY EXEC PGM=IKJEFT01 //SYSIN DD *
#!/bin/sh
echo 'in '$Ø
echo 'steplib=''$STEPLIB
echo 'printenv begin'
printenv
echo 'printenv end'
echo 'set begin'
set
echo 'set end'
#setting steplib in STDENV not enough!!
export STEPLIB='TSHVR2.LOAD.TEST'
echo 'steplib=''$STEPLIB
echo 'calling shrxplsp.cmd'
/home/tshvr/shrxplsp.cmd
/*
//SHDIR DD PATH='/home/tshvr/shrxplsp.sh',
//     PATHOPTS=(OCREAT,OTRUNC,OWRONLY),PATHMODE=SIRWXU
//SYSTSPRT DD SYSOUT=X
//SYSTSIN DD *
//OCOPY INDD(SYSIN) OUTDD(SHDIR)
*/
/*
/*COPY REXX TO HFS
//CMDCOPY EXEC PGM=IKJEFT01 //SYSIN DD *,DLM=$$
/* REXX */
rc=Ø
/*trace i*/
do j=1 to __environment.Ø
   say '__environment.'j'='__environment.j
end
ADDRESS LINKMVS 'SHRXPLSP'
say 'ADDRESS LINKMVS rc='rc
return rc
**

```pli
//SHDIR    DD  PATH='/home/tshvr/shrxplsp.cmd',
     PATHOPTS=(OCREAT,OTRUNC,OWRONLY),PATHMODE=SIRWXU
//SYSTSPRT DD  SYSOUT=X
//SYSTIN  DD *
     OCOPY INDD(SYSIN) OUTDD(SHDIR)
/*

//COMPILE PL/1
//CMPL    EXEC PGM=IEL1AA
//STEPLIB  DD  DISP=SHR,DSN=IEL8.SIELCOMP
//SYSTIN  DD *
SHRXPLSP:PROCEDURE(PARMS)
     OPTIONS(MAIN,NOEXECOPS,REENTRANT);
     DCL PARMS CHAR(*) VARYING;
     DCL SYSPRINT FILE STREAM OUTPUT;
     PUT SKIP LIST('IN PL/I SHRXPLSP');
END SHRXPLSP;
*

//SYSLIN  DD DSN=&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,
     SPACE=(800,(500,500))
//SYSPRINT DD SYSOUT=* 
//SYSUT1  DD SPACE=(800,(500,500)...ROUND),UNIT=SYSDA
/*LINK PL/1
//LKED    EXEC PGM=IEWL,
     // PARM='NOXREF,AMODE=31,RMODE=ANY,RENT,REUS'
//SYSLIB  DD DISP=SHR,DSN=CEE8.CEELEKED
//SYSPRINT DD SYSOUT=* 
//SYSUT1  DD SPACE=(1024,(50,50)),UNIT=SYSDA
//SYSLMOD  DD DISP=SHR,DSN=TSHVR2.LOAD.TEST(SHRXPLSP)
/*EXECUTE
//BPXBATCH EXEC PGM=BPXBATCH,
     // PARM='SH /home/tshvr/shrxplsp.sh'
//SYSPRINT DD SYSOUT=X
//STDIN  DD PATH='/dev/null'
//STDOUT DD PATH='/home/tshvr/shrxplsp.out',
     // PATHOPTS=(OCREAT,OTRUNC,OWRONLY),PATHMODE=SIRWXU
//STDERR DD PATH='/home/tshvr/shrxplsp.err',
     // PATHOPTS=(OCREAT,OTRUNC,OWRONLY),PATHMODE=SIRWXU
//STDENV DD *
     _BPX_BATCH_SPAWN=YES
     _BPX_SHAREAS=MUST
     STEPLIB='TSHVR2.LOAD.TEST'
/*
```

Herman Vierendeels
Systems Programmer (Belgium) © Xephon 2000

Keeping track of load module changes – part 2

This month we complete our look at a program that allows users to find out how many times a program has been changed since it was first link-edited. This is particularly useful for change management purposes.

**JES/2 PROCEDURE LMODPROC**

```plaintext
//CHANGE      PROC OUT=,IN=
//**********************************************************
//* Procedure  : Lmodproc
//* Function   : Extracts the LMODDATES of a load library
//* of which name is given by the user.
//* Parameters :
//*
//*    IN  : Load Library name as input.
//*   OUT  : LMOD dataset of the load library of which name is given
//*          by the parameter 'IN'.
//**********************************************************
//*STEP1      EXEC PGM=AMBLIST
//**********************************************************
//* Execute the Amblist service aid and get all the load module
//* information.
//**********************************************************
//*SYSPRINT     DD   DSN=&&IDR1,DISP=(,PASS),UNIT=SYSDA,
//*   SPACE=(CYL,(1,1)),DCB=(LRECL=121,BLKSIZE=27951,RECFM=FBA)
//*SYSLIB       DD   DISP=SHR,DSN=&IN
//*LOADLIB      DD   DISP=SHR,DSN=&IN
//*SYSIN        DD   DISP=SHR,DSN=SDID.MVS.LIB.DATA(CTRL1),FREE=CLOSE
//***********************************************************
//* Extract the necessary portions of the Amblist output by using
//* ISPF Batch Search-for utility.
//**********************************************************
//*STEP2        EXEC PGM=ISRSUPC,PARM='DELTAL,SRCHCMP,ANYC'
//*SYSPRINT     DD   SYSOUT=*  
//*NEWDD        DD   DSN=&&IDR1,DISP=(OLD,DELETE)
//*OUTDD        DD   DSN=&&IDR2,DISP=(,PASS),UNIT=SYSDA,
//*   SPACE=(CYL,(1,1)),DCB=(LRECL=133,BLKSIZE=3325,RECFM=FBA)
//*SYSIN        DD   DISP=SHR,DSN=SDID.MVS.LIB.DATA(CTRL2),FREE=CLOSE
```
/*
******************************
/*  Build the LMOD dataset by using the PL/I program (LMODPLI). Here,
/*  the dates in the format of both Julian and Normal is written to
/*  it. Input given by the user is passed to the PL/I program.
/*
******************************
/*
//STEP3        EXEC PGM=LMODPLI,PARM='&IN'
//SYSLIB      DD   DISP=SHR,DSN=SDIAGAS.USER.LOAD
//SYSPRINT     DD   SYSOUT=*  
//GO.IN        DD   DSN=&IDR2,DISP=(OLD,DELETE)
//GO.OUT       DD   DISP=SHR,DSN=&OUT
/*
******************************
/*  Sort the LMOD dataset by Julian date.
******************************
/*
//STEP4        EXEC PGM=SORT
//SYSOUT       DD   SYSOUT=*  
//SORTIN       DD   DISP=SHR,DSN=&OUT
//SORTOUT      DD   DISP=SHR,DSN=&OUT
//SORTWKØ1     DD   UNIT=SYSDA,SPACE=(CYL,2Ø)
//SORTWKØ2     DD   UNIT=SYSDA,SPACE=(CYL,2Ø)
//SYSIN        DD   DISP=SHR,DSN=SDID.MVS.LIB.DATA(CTRL3),FREE=CLOSE
/*

CLIST SDICLMOD

PROC Ø

/*********************/
/*  KEEPING TRACK OF LOAD MODULE CHANGES */
/*  ISPF part of the utility */
/*  LMOD, HISTORY and TRANSACTION datasets are browsed under ISPF */
/*  using 'ISPEXEC BROWSE' command. */
/*********************/

CONTROL NOFLUSH NOLIST NOCONLIST NOSYMLIST NOMSG
SET &SYSMSG = OFF

ISPEXEC VGET (ZCMD) PROFILE    /* Get the option chosen */
   /* on panel SDINLIB2.     */

SELECT (&ZCMD)
   WHEN(1) DO
ISPEXEC DISPLAY PANEL(SDINLIB2)
ISPEXEC VGET (ZCMD) PROFILE /* Get the option chosen */
/* on panel SDINLIB2. */
IF &ZCMD = 1 THEN ISPEXEC BROWSE DATASET('SDID.MVS.LIB.CHANGE1')
IF &ZCMD = 2 THEN ISPEXEC BROWSE DATASET('SDID.MVS.LIB.CHANGE2')
EXIT
END

WHEN(2) DO
  ISPEXEC DISPLAY PANEL(SDINLIB2)
  ISPEXEC VGET (ZCMD) PROFILE /* Get the option chosen */
  /* on panel SDINLIB2. */
  IF &ZCMD = 1 THEN ISPEXEC BROWSE DATASET('SDID.MVS.LIB.HISTS1')
  IF &ZCMD = 2 THEN ISPEXEC BROWSE DATASET('SDID.MVS.LIB.HISTS2')
  EXIT
END

WHEN(3) ISPEXEC BROWSE DATASET('SDID.MVS.LIB.SUM')
OTHERWISE
END /* Select end

END /* End of Clist

CLIST SDICLIB2
PROC Ø

/*******************************************************************************/
/*  KEEPING TRACK OF LOAD MODULE CHANGES */
/*  Extract the last link-edited load modules. */
/******************************************************************************/
CONTROL NOCONLIST NOSYMLIST

SET &SYSMSG = OFF
SET &BUGUN = &SYSJDATE
SET &BUGUN = &STR(&BUGUN)
SET &GUN1 = &SUBSTR(4:6,&BUGUN)
SET &YIL = &SUBSTR(1:2,&BUGUN)
SET &GUN2 = &GUN1 - 1
IF &LENGHT(&GUN2)=1 THEN SET &GUN2=STR(ØØ)&GUN2
IF &LENGHT(&GUN2)=2 THEN SET &GUN2=STR(Ø)&GUN2
SET &BUGUN  = &STR(&YIL)&STR(&GUN1)        /* FORMAT CONVERSION*/
SET &HAFTA  = &STR(&YIL)&STR(&GUN2)        /* 92.134  --> 92134*/

ALLOC F1(CHANGE) DA('SDID.MVS.LIB.CHANGEX') SHR
ALLOC F1(SUM)    DA('SDID.MVS.LIB.SUM') SHR

OPENFILE CHNAGE INPUT
OPENFILE SUM    OUTPUT
GETFILE  CHNAGE
SET &KUTTAR     = &SUBSTR(11.15,7CHANGE)

DO WHILE &KUTTAR >= &HAFTA
    SET &SUM = &CHNAGE
    PUTFILE SUM
        /* Extract last changed */
        /* load modules */
    GETFILE CHANGE
    SET &KUTTAR  = &SUBSTR(11:15,&CHANGE)
END

CLOSEFILE CHNAGE
CLOSEFILE SUM

FREE F1(CHANGE)
FREE F1(SUM)

END /* End of Clist

CLIST SDICLMOD
PROC Ø

/*********************************************/
/*                                      */
/*  Clist : Sdiclmod                    */
/*  Function : This CLIST gets the name of load library from the TSO */
/*            user, prepares a JCL and submits it, then presents */
/*            the result dataset to the user.                      */
/*                                      */
/*********************************************/

CONTROL MAIN NOCONLIST NOMSG NOFLUSH
SET &SYSMSG = OFF

/*********************************************/
/* Get input load dataset from the TSO user. */
/*********************************************/
WRITE Please enter the load library of which LMODDATES you want to see
READ &KITAPLIK

/*********************************************/
/* Verify whether it is a cataloged dataset. */
/*********************************************/
IF &SYSdsn('&KITAPLIK') = OK THEN +
   DO
      WRITE There is no such library in the system.
      EXIT
   END
/*******************************************************************************
/* If it is catalogued, then check out that it is a load library. */
/* The LISTDSI statement can retrieve information about a dataset. */
/* If record format is 'U' then it is definitely a load library. */
*******************************************************************************/
LISTDSI '&KITAPLIK'
IF &SYSRECFM = U THEN +
   DO
      WRITE Sorry, the dataset you entered is not a load library.
      EXIT
   END
/*******************************************************************************
/* Allocate the result dataset in which LMOD DATES of library members */
/* will be written. */
*******************************************************************************
SET &SONUC = &SYSuid..LMOD.CHG
IF &SYSdsn('&SONUC') = OK THEN +
   DO
      FREE DA('&SONUC')
      ALLOC FI(SON) DA('&SONUC') NEW SPACE(1,1) TRACKS VOL(SYSDAO1) +
           BLKSIZE(27966) LRECL(79) DSORG(PS) RECFM(F,B)
   END
   FREE DA('&SONUC')
ALLOC DA('&SONUC') SHR REUSE
WRITE Please wait just a few seconds. Job is being submitted. It may take several minutes for big load libraries.

/*******************************************************************************
/* The procedure LMODPROC is invoked. At the end of run of this */
/* procedure, the result file is ready to be browsed by the TSO */
/* user. &SONUC is the result LMOD dataset. */
*******************************************************************************
SUBMIT * END(ZZ)
//&SYSUID.U JOB (&SYSUID),CLASS=A,NOTIFY=SDIAGAS,
// MSGCLASS=X,MSGLEVEL=(1,1),TIME=1440
//PERNAS EXEC LMODPROC,OUT='&SONUC',
// LOAD='&KITAPLIK'
ZZ
/*******************************************************************************
/* Wait here until the above job is finished. Then browse the LMOD */
/* dataset. */
*******************************************************************************
MARTA:ISPEXEC CONTROL ERRORS RETURN

IF &ZFBROWS = &Z THEN ISPEXEC CONTROL NONDISPL END
ISPEXEC BROWSE DATASET('&SONUC')
SET &RC = &LASTCC
ISPEXEC CONTROL ERRORS CANCEL
IF &RC = 12 THEN GOTO MARTA

/*********************************************************************/
/* The LMOD dataset is presented to the user. The user can keep     */
/* browsing it until PFØ3 key is pressed. Once he/she gets out of it */
/* it is deleted.                                                   */
/*********************************************************************/
ISPEXEC BROWSE DATASET('&SONUC')
FREE DA('&SONUC')
DELETE '&SONUC'
END /* End of Clist

PANEL SDINLIB1

)ATTR DEFAULT(%+_)
* TYPE(TEXT) INTENS(HIGH) COLOR(PINK) CAPS(OFF)
? TYPE(TEXT) INTENS(LOW) COLOR(BLUE) CAPS(OFF)
> TYPE(TEXT) INTENS(HIGH) COLOR(RED) CAPS(OFF)
)BODY
%-----------------* KEEPING TRACK OF %-----------------------
%                 * LOAD MODULE CHANGES  %
%SELECTION ===> _ZCMD
+
%1 -+LMOD DATES of load modules
%2 --Change counts and dates of load modules
%3 -+Load modules link edited yesterday
%4 --Find LMOD DATES of any load library
+
+
% X+ Exit
+
)INIT
&ZPRIM  = NO
&ZCMD   = &Z
)PROC
VPUT (ZCMD) PROFILE
&ZSEL=TRANS( TRUNC (%ZCMD,'.')
  1,'CMD(SDICLIB1)'
  2,'CMD(SDICLIB1)'
  3,'CMD(SDICLIB1)'
  4,'CMD(SDICLMOD)'
  '',''
 X,'EXIT')
&ZTRAIL = .TRAIL
)END
PANEL SDINLIB1

)ATTR DEFAULT(%+_)
* TYPE(TEXT) INTENS(HIGH) COLOR(PINK) CAPS(OFF)
? TYPE(TEXT) INTENS(LOW) COLOR(BLUE) CAPS(OFF)
> TYPE(TEXT) INTENS(HIGH) COLOR(RED) CAPS(OFF)
)BODY
%-----------------* KEEPING TRACK OF %-----------------------
%                 * LOAD MODULE CHANGES %
%SELECTION ===> _ZCMD 
+ %1+- SDIAGAS.USER.LOAD 
 %2+- SDIAGAS.AGMPV.LOADLIB 
 %3+- ... 
 %4+- ... 
+ 
 + % X+ Exit 
+ 
 )INIT
 .CURSOR = ZCMD
 &ZPRIM = NO
 &ZCMD = &Z
 )PROC
 VPUT (ZCMD) PROFILE 
 &EBPFK = .PFKEY 
)END

JOB NULLIFY

//SDIAGAS1 JOB (SDIAGAS),MSGCLASS=X,MSGLEVEL=(1,1),
// NOTIFY=SDIAGAS
//*
******************************************************************************
//** Nullifies the utility sequential datasets
******************************************************************************
//*
//ADIM1       PROC NULLF=
//STEP        EXEC PGM=IEBGEBER
//SYSPRINT DD  SYSOUT=* 
//SYSUT1 DD DISP=SHR,DSN=NULLFILE,DCB=SDID.MVS.LIB.HISTS1
//SYSUT2 DD DISP=SHR,DSN=&NULLF
//SYSIN DD DUMMY
// PEND
//*
//ADIM2 PROC NULLF=
//STEP EXEC PGM=IEBGEBER
//SYSPRINT DD SYSOUT=* 
//SYSUT1 DD DISP=SHR,DSN=NULLFILE,DCB=SDID.MVS.LIB.CHNAGE1
//SYSUT2 DD DISP=SHR,DSN=&NULLF
//SYSIN DD DUMMY
//            PEND
//*/
//CALL1        EXEC ADIM1,NULLF='SDID.MVS.LIB.HISTS1'
//CALL2        EXEC ADIM1,NULLF='SDID.MVS.LIB.HISTS2'
//*/
//CAGIR1       EXEC ADIM2,NULLF='SDID.MVS.LIB.CHANGE1'
//CAGIR2       EXEC ADIM2,NULLF='SDID.MVS.LIB.CHANGE2'
//CALLSUM      EXEC ADIM2,NULLF='SDID.MVS.LIB.SUM'
//*/
//CAGIRX       EXEC ADIM2,NULLF='SDID.MVS.LIB.CHANGEX'
//*/
//CAGIR1       EXEC ADIM2,NULLF='SDID.MVS.LIB.CHANGE1'
//CAGIR2       EXEC ADIM2,NULLF='SDID.MVS.LIB.CHANGE2'
//*/

AMBLIST SERVICE AID CONTROL STATEMENT (CTRL1)
LISTIDR OUTPUT=ALL.TITLE=('LOAD MODULES CREATION DATES',27)

ISPF SEARCH UTILITY CONTROL STATEMENT (CTRL2)
SRCHFOR 'MEMBER NAME'
SRCHFOR 'OF YEAR'

DFSORT CONTROL STATEMENT (CTRL3)
SORT FIELDS=(11,5,D),FORMAT=BI
RECORD TYPE=F,LENGTH=17
END

DFSORT CONTROL STATEMENT (CTRL4)
SORT FIELDS=(1,9,A),FORMAT=CH
RECORD TYPE=F,LENGTH=79
END

DFSORT CONTROL STATEMENT (CTRL5)
SORT FIELDS=(13,5,D),FORMAT=CH
RECORD TYPE=VB,LENGTH=6027
END

SDID.MVS.LIB.CHANGE1
This holds the LMOD dates of the load library sorted by date.
DENE2  98296  23.OCTOBER   .1998 SDIAGAS.USER.LOAD
LMODPLI4 98295  22.OCTOBER   .1998 SDIAGAS.USER.LOAD
SDID.MVS.LIB.CHNAGE2

This holds the LMOD dates of the load library sorted by date.

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Year</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATOS</td>
<td>98296</td>
<td>17.OCTOBER</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
<tr>
<td>ATOS</td>
<td>98168</td>
<td>17.JUNE</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
<tr>
<td>ATOS</td>
<td>98148</td>
<td>28.MAY</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
<tr>
<td>ATOS</td>
<td>98146</td>
<td>26.MAY</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
<tr>
<td>ATOS</td>
<td>98141</td>
<td>21.MAY</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
<tr>
<td>ATOS</td>
<td>98146</td>
<td>26.MAY</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
</tbody>
</table>

SDID.MVS.LIB.CHANGE1

This holds the LMOD dates of one load library sorted by member name.

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Year</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATALAY</td>
<td>98290</td>
<td>17.OCTOBER</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
<tr>
<td>DENE2</td>
<td>98296</td>
<td>23.OCTOBER</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
<tr>
<td>DIRECT</td>
<td>98141</td>
<td>21.MAY</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
<tr>
<td>GRAPH</td>
<td>90108</td>
<td>18.APRIL</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
<tr>
<td>GUL</td>
<td>98138</td>
<td>18.MAY</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
<tr>
<td>LMODPLI</td>
<td>98264</td>
<td>21.SEPTEMBER</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
<tr>
<td>LMODPLI2</td>
<td>98283</td>
<td>10.OCTOBER</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
<tr>
<td>LMODPLI3</td>
<td>98275</td>
<td>02.OCTOBER</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
<tr>
<td>LMODPLI4</td>
<td>98295</td>
<td>22.OCTOBER</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
<tr>
<td>MARTA</td>
<td>98152</td>
<td>01.JUNE</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
<tr>
<td>MARTAPV</td>
<td>98149</td>
<td>29.MAY</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
<tr>
<td>MERGE</td>
<td>98170</td>
<td>19.JUNE</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
<tr>
<td>VILLER</td>
<td>98146</td>
<td>26.MAY</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
<tr>
<td>PDS</td>
<td>98141</td>
<td>22.MAY</td>
<td>.1998 SDIAGAS.USER.LOAD</td>
</tr>
</tbody>
</table>
### SDID.MVS.LIB2.CHNAGE2

This holds the LMOD dates of one load library sorted by member name.

<table>
<thead>
<tr>
<th>Load Library</th>
<th>Date</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATOS1</td>
<td>98148</td>
<td>1998</td>
</tr>
<tr>
<td>ATOS2</td>
<td>98168</td>
<td>1998</td>
</tr>
<tr>
<td>ATOS3</td>
<td>98296</td>
<td>1998</td>
</tr>
<tr>
<td>ATOS4</td>
<td>98141</td>
<td>1998</td>
</tr>
<tr>
<td>ATOS5</td>
<td>98146</td>
<td>1998</td>
</tr>
</tbody>
</table>

### SDID.MVS.LIB.CHANGEX

Merge dataset consisting of all LMOD datasets sorted by date.

<table>
<thead>
<tr>
<th>Load Library</th>
<th>Date</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATOS3</td>
<td>98296</td>
<td>1998</td>
</tr>
<tr>
<td>DENE2</td>
<td>98296</td>
<td>1998</td>
</tr>
<tr>
<td>LMODPLI4</td>
<td>98295</td>
<td>1998</td>
</tr>
<tr>
<td>ATALAY</td>
<td>98290</td>
<td>1998</td>
</tr>
<tr>
<td>LMODPLI2</td>
<td>98283</td>
<td>1998</td>
</tr>
<tr>
<td>LMODPLI3</td>
<td>98275</td>
<td>1998</td>
</tr>
<tr>
<td>LMODPLI</td>
<td>98264</td>
<td>1998</td>
</tr>
<tr>
<td>MERGE</td>
<td>98170</td>
<td>1998</td>
</tr>
<tr>
<td>ATOS2</td>
<td>98168</td>
<td>1998</td>
</tr>
<tr>
<td>MARTA</td>
<td>98152</td>
<td>1998</td>
</tr>
<tr>
<td>MARTAPV</td>
<td>98149</td>
<td>1998</td>
</tr>
<tr>
<td>ATOS1</td>
<td>98148</td>
<td>1998</td>
</tr>
<tr>
<td>ATOS5</td>
<td>98146</td>
<td>1998</td>
</tr>
<tr>
<td>VILLER</td>
<td>98146</td>
<td>1998</td>
</tr>
<tr>
<td>DIRECT</td>
<td>98141</td>
<td>1998</td>
</tr>
<tr>
<td>PDS</td>
<td>98141</td>
<td>1998</td>
</tr>
<tr>
<td>ATOS4</td>
<td>98141</td>
<td>1998</td>
</tr>
<tr>
<td>GUL</td>
<td>98138</td>
<td>1998</td>
</tr>
<tr>
<td>TAPECOPY</td>
<td>98100</td>
<td>1998</td>
</tr>
</tbody>
</table>

### SDID.MVS.LIB.HISTV1

These are the history VSAM datasets that hold the historic change dates of the load modules for one load library.
KEY OF RECORD - ATALAY
ATALAY  0004 98148.. 98283 98285 98290
KEY OF RECORD - DENE2
DENE2  0003 98168.. 98295 98295 98296
KEY OF RECORD - DIRECT
DIRECT  0001 98141
KEY OF RECORD - GRAPH
GRAPH  0001 90108
KEY OF RECORD - GUL
GUL  0001 98138
KEY OF RECORD - LMODPLI
LMODPLI  0001 98264
KEY OF RECORD - LMODPLI2
LMODPLI2  0010 98264.. 98265 98266 98269 98272 98273 98275 98280 98282 98283
KEY OF RECORD - LMODPLI3
LMODPLI3  0007 98264.. 98265 98266 98269 98272 98273 98275
KEY OF RECORD - LMODPLI4
LMODPLI4  0002 98264.. 98265
KEY OF RECORD - MARTA
MARTA  0001 98152
KEY OF RECORD - MARTAPV
MARTAPV  0001 98149
KEY OF RECORD - MERGE
MERGE  0001 98170
KEY OF RECORD - PDS
PDS  0001 98141
KEY OF RECORD - REDLOCK
REDLOCK  0001 88004
KEY OF RECORD - TAPECOPY
TAPECOPY  0001 91100
KEY OF RECORD - TAPEMAP
TAPEMAP  0001 86291
KEY OF RECORD - TAPEMAP2
TAPEMAP2  0001 84202
KEY OF RECORD - TAPESCAN
TAPESCAN  0001 82329
KEY OF RECORD - VILLAR
VILLAR  0001 98146
IDC0005I NUBER OF RECORDS PROCESSED WAS 19

SDID.MVS.LIB.HISTV2
These are the history VSAM datasets that hold the historic change
dates of the load modules for one load library.

KEY OF RECORD - ATOS1
ATOS1  0001 98148
KEY OF RECORD - ATOS2
ATOS2  0001 98168
KEY OF RECORD - ATOS3
ATOS3 ØØØ3 98170.. 98295 98296
KEY OF RECORD - ATOS4
ATOS4 ØØØ1 98141
KEY OF RECORD - ATOS5
ATOS5 ØØØ1 98146
IDC0005I NUBER OF RECORDS PROCESSED WAS 5

SDID.MVS.LIB.HISTS1
This is the history sequential dataset. It is the sequential copy of the HISTV VSAM datasets. They are used to present the user contents of VSAM datasets under ISPF.

$MEMBER $COUTN$DATES
$====== $==== $========
LMODPLI2 ØØ1Ø 98264.. 98265 98266 98269 98272 98273 98275 98280 98282 98283
LMODPLI3 ØØØ7 98264.. 98265 98266 98269 98272 98273 98275 98280 98282 98283
ATALAY ØØØ4 98148.. 98283 98285 98290
DENE2 ØØØ3 98168.. 98293 98285 98290
LMODPLI4 ØØØ2 98264.. 98265
DIRECT ØØØ1 98141
GRAPH ØØØ1 90108
GUL ØØØ1 98138
LMODPLI ØØØ1 98264
MARTA ØØØ1 98152
MARTAPV ØØØ1 98149
MERGE ØØØ1 98170
PDS ØØØ1 98141
REDLOCK ØØØ1 88004
TAPECOPY ØØØ1 91100
TAPEMAP ØØØ1 86291
TAPEMAP2 ØØØ1 84202

SDID.MVS.LIB.HISTS2
This is the history sequential dataset. It is the sequential copy of the HISTV VSAM datasets. These are used to present the user contents of VSAM datasets under ISPF.

$MEMBER $COUTN$DATES
$====== $==== $========
ATOS3 ØØØ3 98170.. 98295 98296
ATOS1 ØØØ1 98148
ATOS2 ØØØ1 98148
ATOS4 ØØØ1 98148
ATOS5 ØØØ1 98148
INTRODUCTION

PDF’s PROFILE has a lot to do with how the screen behaves when you work with a file.

Each file type has a separate PROFILE. The file type is usually determined from the last segment of the filename. For instance, if the filename is ‘A.B.C.JCL’, then the file type is ‘JCL’. This can be overridden on the ISPF selection screen where you specified the file you wished to work with.

SDID.MVS.LIB.SUM

This is a transaction dataset. It contains the recently updated load modules. By using this dataset, HISTV VSAM datasets are updated daily.

Atalay Gul
Systems Programmer
Central Bank of Turkey (Turkey)

© Xephon 2000
While in an application, such as EDIT, enter ‘PROF’ or ‘PROFILE’ on the command line. Something similar to the following lines will appear under the command line:

```
=PROF> ....COBOL (FIXED - 80)....RECOVERY ON....NUMBER ON COB..............
=PROF> ....CAPS ON....HEX OFF....NULLS ON STD....TABS ON ;....SETUNDO REC......
=PROF> ....AUTOSAVE ON....AUTONUM OFF....AUTOLIST OFF....STATS ON.............
=PROF> ....PROFILE LOCK....IMACRO NONE....PACK OFF....NOTE ON..................
=PROF> ....INITIALSEARCH ON....INITIALSEARCH OFF....INITIALSEARCH STAT......
```

What does it mean? We will consider the elements individually and elaborate on the more interesting ones:

- **COBOL (FIXED - 80)** – the last part of our dataset name was COBOL, so this profile defaulted to that. ‘FIXED - 80’ is obvious.

- **RECOVERY ON** – if your session is interrupted by a network problem or mainframe malfunction, the next time you log on and try to get into this function (EDIT), a special screen will come up
asking if you want to resume editing your member. If you choose ‘yes’, the session will resume where you left off. It will not include any changes you made since the last time you pressed <Enter> or some other interrupt, such as a PF key, but it will have all your other changes (remember, the mainframe does not know what you are doing on the screen until you press <Enter> or a PF key). You can change the setting of RECOVERY by entering ‘RECOVERY OFF’ or ‘RECOVERY ON’ on the command line.

- NUMBER ON COB – COBOL numbering is on (notice the numbers to the left of the code below our profile). Possible settings (on the command line) are NUMBER ON COB, NUMBER ON STD (numbers in cols 73-80), NUMBER OFF, or UNNUM (remove numbers and set NUMBER to OFF).

- CAPS ON/OFF – if caps are ‘ON’, entered text will be changed to caps when you press <Enter> or a PF key. Existing lower-case text would not be changed. If caps are ‘OFF’, text will be recorded as entered (upper/lower case).

- HEX ON/OFF – see the discussion on ‘Hex’ at the close of the article.

- NULLS ON STD – trailing blanks on a line, except for the first one, will be nulls (hex ‘00’) rather than spaces (hex ‘40’). Spaces take up space and get in the way of inserting characters; nulls don’t. Items entered to the right of a null-filled line will left shift to the end of the existing line when <Enter> is pressed, space-filled lines will stay where they are put. If you press ‘End’ to erase to end of line, you always create nulls until <Enter> is pressed, then this setting tells the computer what to do with the deleted area: leave as nulls or convert to spaces. Using the ‘Delete’ key produces nulls to the right of the line as it left-shifts. If the field is entirely empty, it is written as all spaces.

ON ALL – specifies that all trailing blanks and all-blank fields are written as nulls.

OFF – specifies that trailing blanks in each data field are written as spaces.
• TABS ON – tabbing is on and the logical tab character is ‘;’ (a semi-colon). You can enclose the character in quotes, although this is not necessary unless a quote or a comma (,) is used as the tab character.

TABS OFF – turns tabs mode off, which means that logical tabs cannot be used. Attribute bytes are deleted from all hardware tab positions.

TABS STD – activates all hardware tab positions (asterisks) that contain a blank or null character. The editor inserts attribute bytes, which cannot be typed over, at these positions. STD is the default operand.

TABS ALL – causes an attribute byte to be inserted at all hardware tab positions. Characters occupying these positions are blanked out and the attribute bytes cannot be typed over.

• SETUNDO REC – enables the ‘UNDO’ command by saving changes in the recovery file (‘REC’ or ‘RECOVER’) or memory (‘STG’, ‘STORE’, ‘STOR’, or ‘STO’). Command line settings: ‘SETUNDO REC’, ‘SETUNDO OFF’, ‘SETUNDO STO’, etc. If RECOVERY is ON, SETUNDO OFF is the same as SETUNDO REC. If RECOVERY is OFF, it will be turned on by this command:

```
=PROF> ....AUTOSAVE ON....AUTONUM OFF....AUTOLIST OFF....STATS ON..............
```

• AUTOSAVE ON – automatically saves your file and changes when you exit the session, for example, entering ‘END’ on the command line, pressing PF3, etc. Entering ‘CAN’ on the command line will leave the session without saving your changes. If AUTOSAVE is OFF, you will have to enter ‘SAVE’ on the command line before you exit the session.

• AUTONUM OFF – when you insert new lines they will be numbered between the existing lines until the computer runs out of numbers, then as many lines as necessary after the new work will be renumbered to accommodate the inserts. You will have to enter ‘RENUM’ on the command line to refresh the numbers. When this is ON inserted lines will cause all following line
numbers to be re-sequenced using the default scheme (number by 100s in the case of COBOL numbering).

- **AUTOLIST ON/OFF** – this sends a source listing into the ISPF list dataset when you end the edit session (assuming you made changes and saved them). The disposition of the ISPF list dataset depends upon your settings. It will be printed, saved, or deleted when you log off from ISPF.

- **STATS ON** – update statistics will be generated for this file. This is the information you see when you list the contents of a PDS, such as ‘Created’ date, ‘Changed’ date, ‘Size’, etc:

  ```
  =PROF> ....PROFILE LOCK....IMACRO NONE....PACK OFF....NOTE ON................
  ```

- **PROFILE LOCK** – when you issue this command, the profile attributes are locked. Any changes made after that will be forgotten when the session ends. Changes during subsequent sessions will also be forgotten when the session is over. If the profile is UNLOCKed, changes made to the profile’s attributes will remain and be available the next time that particular profile is used.

- **IMACRO NONE** – the IMACRO primary command saves the name of an initial macro in the current edit profile. The editor runs an initial macro after it reads but before it displays data. The macro might initialize empty data sets, define program macros, or initialize PF keys. A complete discussion of initial macros is beyond the scope of this article.

- **PACK OFF** – the PACK primary command sets pack mode, which controls whether the data is to be stored in packed format.

- **NOTE ON** – the NOTES primary command sets note mode, which controls whether notes are displayed when a dialog development model is inserted into the data. This is used in conjunction with the MODEL command and is beyond the scope of this article.

  ```
  =COLS> —1——+——2——+——3——+——4——+——5——+——6——+——7——+—-
  ```
• COLS – just what it looks like. Enter COLS in the line command area to get this line anywhere in the screen. It will stay there until cleared.

Another command that is very nice is HILITE. If you have a colour terminal (3270 emulation, etc) it will change the colour of key words in your code. It will not work with PROCOMM or any other terminal that emulates a monochrome terminal.

The HILITE primary edit command is used to change colour highlighting settings. HI and HILIGHT are valid synonyms:

• HILITE RESET – reset defaults (AUTO, ON, Find and Cursor on).
• HILITE ON – set program colouring on (without logic highlighting).
• HILITE OFF – set program colouring OFF.
• HILITE AUTO – let ISPF determine the language.
• HILITE <lang> – force the language.
• HILITE LOGIC – turn on IF and DO logic matching.
• HILITE IFLOGIC – turn on IF logic matching only.
• HILITE DOLOGIC – turn on DO logic matching only.
• HILITE NOLOGIC – turn off all logic matching.
• HILITE FIND – toggle highlighting FIND strings.
• HILITE CURSOR – toggle highlighting of the phrase with the cursor.
• HILITE PAREN – toggle matching of parentheses.
• HILITE SEARCH – finds the first unmatched END, ELSE, }, or ) between the first line in the file, and the first line being displayed. For END, ELSE or } highlighting, you must have the LOGIC enabled. The search for mismatched only occurs for lines above the last displayed line, so you may need to scroll to the bottom of the file.
• HILITE DISABLE – disables all highlighting and removes the action bar. (Note: the DISABLE setting is not retained between edit sessions.)

• HILITE – with no operands it presents a dialog that allows you to change various colouring options.

In many cases, the ISPF editor can determine the language of the file you are editing. If you want to override the automatic language determination, specify the language you want on the HILITE command. Valid language names are:

AUTO    ASM  C       COBOL   DTL      IDL     JCL     PANEL
PASCAL  PLI  OTHER   REXX    BOOK     SKEL    DEFAULT

Example:

COMMAND ===> hi cobol

This will turn on logical highlighting for COBOL program code.

OTHER is a pseudo-language similar to PL/I but with only very basic keywords (DO, END, SELECT, WHEN, IF, THEN, ELSE, etc). OTHER can be used on many languages such as CLIST. OTHER also does not support any compiler directives. DEFAULT is used when AUTO is specified, but no language can be determined.

You can use the edit PROFILE command to see the colouring status. If a language was explicitly selected, the language will be highlighted in RED. Otherwise it will be WHITE.

CLEANING UP THE SCREEN

To get rid of all profiles, tab lines, or column lines, enter ‘RESET’ on the command line. Entering ‘D’ in an individual line command area will clear that line only.

The HELP command has a lot of information on Profile commands although they are sometimes a bit difficult to navigate through. Remember, in HELP you can make use of up (PF7), down(PF8), left (PF10), and right (PF11), as well as <Enter> to navigate through screens. If a screen has +More, <Enter> will get the next screen. <Enter> will often navigate you through everything in a topic.
HEX

The dreaded message comes up:

-CAUTION- Data contains invalid (non-display) characters. Use command
  ===> FIND P'.' to position cursor to these

So you enter ‘f p'.’ in the command line and the browser (or whatever
you’re using) positions you to the offending line in the list. Now, how
to find out what’s really there?

Enter ‘HEX’ on the command line and the listing will be converted to
three lines and a blank line for each original line that was there (with
a lot fewer lines per page). The lines will be: the original line in regular
characters, followed by two lines of hex, each hex equivalent directly
beneath the original character, for example:

ABCD EFG 123
CCCC4CCC4FF0000000 etc.
12340567012300000000

When you are finished, enter ‘HEX OFF’ and things will return to
normal.

You can use Hex any time you need to see the hex equivalent of
something. If you are in EDIT mode, you can edit the hex equivalent
lines to produce characters not on your keyboard or to modify packed
decimal or binary fields.

Allan Kalar
Systems Programmer (USA)  © Xephon 2000
IBM has announced the High Level Assembler for MVS, VM and VSE Release 4. Among the new features are the capability whereby Assembler options can be specified in an external file and the PROCESS OVERRIDE statement allows the setting of fixed options for a source module.

There are two new options: CODEPAGE, which supports the creation of Unicode character constants from EBCDIC data, and the NOTHREAD option, which allows users to specify that the location counter should be reset to zero for each control section, which helps with program debugging and address computation.

The XATTR statement lets users assign attributes to external symbols, to assist with using DLLs, and there are new DC constant types, including: R – PSECT address, for use with programs using constructed reentrancy, CU – Unicode character constant, FD – doubleword aligned 8-byte fixed-point constant, AD – Doubleword aligned 8-byte address constant; and AMODE and RMODE statements are enhanced.

The new release comes with various usability enhancements: literal operands are always entered in the literal pool, providing more uniform behaviour of attribute references to literal operands; message wording is said to be improved and more information is provided about any operands involved.

Contact your local IBM representative for further information.

http://www.ibm.com

Tivoli has announced Version 7.0 of its workload scheduler (TWS), which provides a single point of control for open workload management and uses open interfaces to enable communications with OS/400, OS/390, Unix, and Windows environments. The Extended Agents enable management of workloads on non-TWS platforms and ERP applications.

There is a new Java-based GUI console that can simultaneously manage both TWS and OPC networks and there’s support for ten languages. Management of systems across multiple time zones is improved by attributing the appropriate time zone to each workstation. It now logs changes made to the TWS database or plan to a log file for auditing and security purposes.

The new Job Scheduling Console is for configuring, viewing, and modifying all aspects of workload planning at a site. It also allows scheduling of jobs and job streams to create a plan for job execution, using the different TWS job dependency types. Finally, it can monitor and modify the execution of all jobs and job streams, by adding, modifying, or deleting jobs or job streams in the plan.

For further information contact: Tivoli Systems, 9442 Capital of Texas Highway, North Austin, TX 78759, USA. Tel: 512 436 8000 Fax: 512 794 0623

Tivoli Systems, Sefton Park, Bells Hills, Buckinghamshire, SL2 4HD, UK. Tel: 01753 896 896 Fax: 01753 896 899 http://www.tivoli.com

http://www.tivoli.com