In this issue

3  Printing JES output from a PC
4  Converting sequential data into HTML
8  The importance of COBOL for OS/390 Version 2 Release 2
13 Synchronizing housekeeping tasks in a Sysplex
27 Managing dynamic dump datasets
48 REXPDSM REXX function
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

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

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.
Printing JES output from a PC

If like me you work at a site remote from the central service and, in particular, from central JES printer support, you may find the following REXXX useful. This can be used either ‘as is’ or as a basis for your own developments. In its supplied form it is an edit macro which takes JES output that has been saved into a dataset and converts the data into rich text format. It does this by reading the standard ASA characters and machine code printer control codes in the print and using them to format the print. The resulting data can then be downloaded to a PC as an .rtf file and printed from, for example, Microsoft Word on a local PC-connected printer in such a manner that it will appear as if the print was done on a mainframe printer.

Should you wish to modify this macro for your site, I did find a copy of the latest RTF specification on the following Web site that may be of use:

http://www.cena.dgac.fr/~sagnier/info/formats/rtf/rtfspe15.htm

SOURCE

/* REXX */
/* */
/* This edit macro is designed to allow mainframe print at up to 67 */
/* lines to a page to be printed from products such as MS Word on a */
/* PC in such a manner that it ought to appear like mainframe print.*/
/* */
address isredit
'MACRO'
font='Courier New'
/* */
/* First set up the RTF header information */
/* */
QUEUE '{\rtf1\ansi\ansicpg1252\deffØ'
QUEUE 'paperw1684Ø\paperh119Ø7\margl144Ø\margr144Ø'
QUEUE '\margt454\margb454\landscape\viewscale100'
QUEUE '{\fonttbl'
QUEUE '{\fØ\froman\fcharsetØ\fprq2'||font';}}{s16\sl\mult1'
'CAPS OFF'
'ISREDIT (WIDE) = DATA_WIDTH'
"C P'.' ' ' 2" wide "ALL" /* remove unprintables */
"C X'7D' X'07' 2" wide "ALL" /* convert quotes to avoid errors */
"C X'7F' X'08' 2" wide "ALL" /* convert quotes to avoid errors */
Converting sequential data into HTML

These days more and more information needs to be made available via a Web browser. However, if you have been working with mainframes for years, the odds are you have a lot of information in PDSs etc, which it would be nice to publish but which you do not want to maintain on a PC. The following REXXX might therefore be of use. It is a simple edit macro which will convert the text in a member to HTML format, that can then be copied to a file for download to a PC or Web server.
In order to use the macro, copy it into a library in your SYSPROC or SYSEXEC concatenation with a suitable name. In my case it is called HTML2. You will also need to copy the associated help panel into your panel library (in which case you can remove the LIBDEF statements); alternatively, you can use any PDS and simply change the ‘your.library’ statement in the LIBDEF for this to work.

In its most basic form, HTML2 will assume that the first line of the data being converted counts as a title, and that your are happy with the colours chosen for text and the background. Should you want to change these, simply follow the instructions in the help panel. Note that to display the help panel, specify a ‘?’ as the parameter to HTML2 and it will be shown.

HTML2 REXX

/* REXX */
/* */
/* This edit macro is designed to convert PDS members into HTML for */
/* displaying in a browser. */
/* */
ADDRESS ISREDIT
'MACRO (string)'
UPPER string
IF string='?' THEN DO /* how to use this has been requested */
  ADDRESS ISPEXEC 'LIBDEF ISPPLIB DATASET ID(your.library)'
  ADDRESS ISPEXEC 'DISPLAY PANEL(HTML2H)'
  ADDRESS ISPEXEC 'LIBDEF ISPPLIB'
  EXIT
END
'(dsname)=DATASET'
'(member)=MEMBER'
IF INDEX(string,'BACKGROUND=')¬=Ø THEN DO
  PARSE VAR string 'BACKGROUND=' bgcolour .
END
ELSE bgcolour='lightcyan'
IF INDEX(string,'NOTITLE')¬=Ø THEN title='N'
ELSE title='Y'
IF INDEX(string,'TEXT=')¬=Ø THEN DO
  PARSE VAR string 'TEXT=' txcolour .
END
ELSE txcolour='black'
/* */
/* First set up the HTML header information */
/* */
QUEUE '<HTML><HEAD>'
QUEUE '<TITLE>'dsnname('member')</TITLE></HEAD>'
QUEUE ' <BODY BGCOLOR='bgcolour' TEXT='txcolour'>'
/* */
"C P'.' ' ' ALL" /* remove unprintables */
"C X'7D' X'07' ALL" /* convert quotes to avoid errors */
"C X'7F' X'08' ALL" /* convert quotes to avoid errors */
"C '&' X'09' ALL" /* convert ampersands to avoid errors */
'(start) = LINENUM .ZF'
'(endit) = LINENUM .ZL'
DO point=start UNTIL point>=endit
  '(line) = LINE point
  IF title='N' & point=1 THEN DO
    QUEUE '<PRE>'
    QUEUE line
    END
  ELSE IF title='Y' & point=1 THEN DO
    QUEUE '<H1>'
    QUEUE line
    QUEUE '</H1><PRE>'
    END
  ELSE QUEUE line
END
QUEUE '</PRE></BODY></HTML>'
/* */
'DEL ALL .ZF .ZL' /* clear the member */
/* */
DO queued()
  '(endit) = LINENUM .ZL'
PARSE PULL line
'LINE_AFTER' endit '=' DATALINE ' ''line'''
END
"C X'07' X'7D' ALL" /* reset quotes */
"C X'08' X'7F' ALL" /* reset quotes */
"C X'09' '&' ALL" /* reset ampersands */
"LOCATE 1"
EXIT 1

HTML2H HELP PANEL

)ATTR
  ' TYPE(PT) /* panel title line */
  ) TYPE(ET) /* emphasized text attribute */
  ~ AREA(SCRL) /* scrollable area attribute */
)BODY
  '——— Help panel for text to HTML ————
  + Command ==>_ZCMD +
  +
  +
The HTML2 edit macro is designed to convert PDS members (or sequential) data into HTML documents. To do this it operates according to the defaults that are preset in the macro, though these can be overridden by the use of four control words.

From the point of view of defaults, the HTML will be generated assuming the following:

The background will be light cyan
The text colour will be black
The first line of the data is assumed to be a header.

If this is inappropriate the following keywords can be used in the macro:

BACKGROUND=keyword where the keyword is any one of the acceptable HTML colour names.
TEXT=keyword where keyword is any one of the acceptable HTML colour names.
NOTITLE This is simply a keyword to switch off the use of the first line as a title line.

PROC
&ZTOP=HTML2H
&ZUP=HTML2H
&ZCONT=HTML2H
)END

Systems Programmer (UK) © Xephon 2000
The importance of COBOL for OS/390 Version 2
Release 2

INTRODUCTION

As each new version of COBOL for OS/390 is released, more and more of the inherent shortcomings of COBOL are ironed out. The latest release is no exception. This one will leave critics of COBOL going back to the drawing board to find more good reasons for their negative feelings.

The latest release of COBOL for OS/390 contains enhancements that are aimed at improving performance, accuracy, and flexibility. It also contains new features that clearly illustrate IBM's commitment towards creating a totally integrated environment for users of OS/390. These features include enabling COBOL applications to be developed and executed in the OS/390 UNIX System Services environment, and the inclusion of a compiler interface to the DB2 coprocessor. There are memory savings, cost and time savings, and a bunch of enhancements that are geared at making the lives of systems developers just that little bit easier.

PERFORMANCE ENHANCEMENTS

Improvements in performance are achieved by making some changes to the way debugging hooks are handled internally and also by alterations to the way binary data is truncated.

Traditionally, debugging hooks were compiled into the program when it was established that the program needed debugging. These extra statements enabling debugging can greatly affect performance if the program is to run in production with that source. The new version uses overlay hooks for debugging which mean that the relevant sections for debugging are only paged into memory when debugging is being done and these have no impact when the program is executing normally. This enhancement also means that developers can leave the version of the program enabled for debugging in production for longer, which is particularly helpful during the parallel run and cut over phases of the development lifecycle.
The new TRUNC(BIN) enhancement improves performance during operations on binary data that needs binary truncations rather than standard COBOL truncation at base-10. This benefit applies primarily to half-word and full-word binary data and will be more obvious in COBOL applications running on CMOS processors.

**IMPROVED ACCURACY AND FLEXIBILITY**

Enhancements that improve accuracy and flexibility for companies with a high reliance on numeric data are the introduction of the new 31-digit length restriction for decimal data, and the new COMP-5 data type. The changes below show a definite move towards narrowing the gap between COBOL and languages traditionally used for more arithmetic intensive calculations. With COBOL’s English-like ease of programming, and the new, powerful options for processing numeric data, we will see a move towards using COBOL in a wider range of business solutions. The previous introduction of OS/390’s Language Environment means that, even where COBOL cannot be used, the OS/390 platform can still be used to support other programming languages, thus reducing the necessity for more than one platform to create computing solutions.

The increase in precision of decimal data from 18 digits to 31 digits, combined with increases in the storage space utilized during calculations in arithmetic statements, is a welcome change and will increase accuracy of calculations. This enhancement will also allow numeric values like account numbers and reference numbers that sometimes have arithmetic meaning and sometimes referential meaning, to be longer and therefore more flexible. This will be a relief for companies who are doing performance intensive fancy coding to get around the previous restrictions.

The new COMP-5 data type allows items to be represented in storage as binary data. This means that the capacity of the native binary representation (2, 4, and 8 bytes) can be fully utilized instead of being limited by the number of 9s defined for the item. Apart from the increased flexibility, this enhancement will have a positive impact on performance and memory space will be better utilized.
OS/390 UNIX SYSTEM SERVICES

The following list of enhancements that allows development and execution of applications in the OS/390 UNIX System Services environment will be a great asset for a number of businesses that are already using both. The enhancements also mean that businesses looking to migrate to the OS/390 platform are going to be looking at OS/390 Unix System Services, because of the obvious advantage of the integrated platform. Programmers with COBOL skills can now be used to create solutions for both platforms, which allows companies to use their resources more flexibly.

HFS support will be provided at development time for COBOL source files, COPY books, object modules, listings, IDL files, ADATA files and executable modules.

COBOL programs will be compile and linked within the OS/390 Unix shell by using the command COB2.

The new version is enhanced to process with COPY or BASIS statements.

The following OS/390 Unix execution applications will be able to run OS/390 Unix applications: OS/390 shell, OS/390 ISPF shell, TSO/E, and OS/390 batch.

COBOL programs can call the standard Unix/POSIX functions.

HFS files and MVS data sets COBOL will be accessible to programs running under OS/390 Unix.

One will be able to optionally route the DISPLAY output to HFS, to stdout, or to stderr, with an enhancement to the DISPLAY statement.

Obtaining data from the HFS or STDIN can be achieved using the format-1 ACCEPT statement.

A new debugging enhancement allows use of the Debug Tool to debug COBOL applications in the OS/390 Unix environment through the remote interface.
THE COMPILER INTERFACE WITH THE DB2 COPROCESSOR

As with the Unix change, this enhancement means that businesses already using both OS/390 and DB2 will gain the convenience of the integrated environment. Interactive debugging of applications using SQL statements with Debug Tool will be radically improved, as they can now be debugged at the original source level. In the past, the application would have been debugged at the level of the expanded source produced by the DB2 pre-compiler. Another big advantage of this is that it will be much more difficult for programs to be erroneously moved back into production using the wrong version of DB2. This change will reduce production problems and stabilize database systems, while reducing the necessity for an extra step in the QA process when migrating test systems into production. Mass recompilations as a result of re-organizations of the database for maintenance will also be quicker now that the extra step isn’t necessary. It must also be said that anyone currently making a decision to migrate towards OS/390 will also be inclined to use DB2 as their database solution.

In short, the new COBOL compiler will interface directly with the DB2 coprocessor as it encounters SQL statements. The coprocessor then analyses the statements and returns any native language statements that must be generated. This means that COBOL programs containing SQL statements no longer require pre-compilation with the DB2 pre-compiler, a welcome change for programmers.

OTHER COST AND TIME SAVINGS

This release of COBOL comes standard with the Millennium Language Extensions, so there is no need to order and install the Millennium Language Extensions separately. The product is also year 2000 ready and Euro ready, which adds to the list of advantages.

Memory savings

Yet another debugging enhancement allows for optimization and reduction of large amounts of memory required for load modules in production. The COBOL symbolic debugging information can now be optionally generated in a separate file from the object module.
Responding to needs of developers to make the language more usable and powerful.

The following list of enhancements show us that someone out there is listening to us after all. They are all ‘nice to have’ changes which go towards making COBOL a more usable, powerful language.

The previous dependence on the pre-linking of COBOL applications is a thing of the past, as applications can now be linked using the DFSMS binder alone. The old pre-linker will now only be necessary under exceptional circumstances.

An enhancement to I/O allows files to be dynamically allocated by using an environment variable.

A new compiler option, DIAGTRUNC, allow us to diagnose moves that resulted in numeric truncation.

Specifying BLKSZE=0 when defining the listing dataset generates a system-determined block size.

The maximum block size for QSAM tape files is now raised to 2 GB. CICS applications become a little more flexible and friendly by using the enhanced DISPLAY statement to display to the system logical output device and the enhanced ACCEPT statement for ascertaining date and time.

SUMMARY OF THE BENEFITS

As a user of various COBOL products over the last ten years, analysing the improvements in the new release of COBOL for OS/390 has left me feeling more positive than ever that COBOL is alive and well and will probably be here a long, long time. These enhancements reduce the gap between COBOL and other languages that have been more popular over the years and start to change the reputation of COBOL for being nothing but a ‘number cruncher’ to being a powerful solution in it’s own right. There is now no reason why it can’t be used to create solutions for a far greater variety of businesses than before. With COBOL resources being relatively easy to come by a lot more businesses may jump on the COBOL bus now that the language has more flexibility and less restrictions.
The OS/390 platform continues to provide greater and greater levels of integration allowing businesses to spend less time trying to get their different products to live together under one roof, and more time solving business problems. As the environment becomes more integrated and easy to manage, businesses will be able to free up resources for other tasks and reduce their reliance on highly paid experts. Another advantage is that an already very complicated and lengthy Quality Assurance process can now be tailored, as some of the steps can be removed or reduced.

Of course, it’s COBOL for OS/390 that has all these wonderful enhancements. This means that as a package deal, the combination of COBOL for OS/390, Unix System Services, and a DB2 database is becoming a very attractive proposition, and one that I will certainly be looking forward to using in the years to come.

Leanda Altman  
Systems Analyst (Australia)  

Synchronizing housekeeping tasks in a Sysplex

THE PROBLEM

There are certain housekeeping tasks in a Sysplex that have the following characteristics:

- The task has to run on a daily basis.
- The task can run from any one of the Sysplex members.
- The task should run once a day.

Samples of jobs that have these requirements are jobs that switch the SYSLOG or LOGREC processing, if the latter is done using the LOGGER facility. The problem is this: if we run such a job at 07:00 am every morning on all of the Sysplex members, it will cause the same task to be done many times. If for instance we switch the SYSLOG and write it to a GDG we will get a new GDG entry each
time the dump job runs. So, if we have five members in the Sysplex, we will get five GDGs in one day – which is not what we want. The obvious solution would be to run the dump job on only one of the members. But what happens if that member happens to be down at 07:00 am? It means that the entire Sysplex misses out on getting a log switch and dump.

A SOLUTION
A far better approach would be if we could have the job start up on each of the Sysplex members, but have only one of them actually do the task. This is what this utility is all about. If it is used, the dump job can start on all systems in the Sysplex at 07:00 am. The ‘first one in’ (say from System A) will actually do the log dump, whilst the others will patiently sit and wait for it to complete, then one by one verify that the job has been done successfully and, if this is the case, terminate without repeating the task. In the (hopefully unlikely) event that the Sysplex member on which the dump process actually occurs should ‘die’ or the job should fail for whatever reason, the next job (say from System B) would step in, detect that the first job has not completed successfully, and run the task. The others behind it will then come in, verify one by one that all has been completed, and terminate without doing anything.

So how does all of this work? The idea is to do an EXCLUSIVE ENQ on a Sysplex-wide name. So, if ten copies of the job (coming from the ten Sysplex members) start all at once, only one will get the ENQ, the others will wait. The one that has the ENQ then opens a control file and leaves a footprint in there to say that the task is in progress. The footprint is a two-byte field in a control file and can be in one of three stages and contain one of three possible values:

• **==** – the task is in progress
• **OK** – the task has been completed successfully
• **AB** – the task has abended.

As soon as the task that got in first releases the ENQ, the others will get in one by one and look out for the OK. A test is then done: if the word OK is found and today’s date is found in the control file, it means
the task has already run today. No matter how many times the task is re-run from any of the Sysplex members, ‘OK’ with today’s date will ensure it is not run again. For information purposes the control file (an 80-byte file/PDS member) also carries the name and number of the job that last updated it and the name of the system that the job ran on.

The utility is used to encapsulate the program we wish to call. For example, if we wanted to run the following JCL:

```
//STEP1        EXEC      PGM=MYPGM,PARM='MYPARM'
//DD1      D   DSN=….
```

We will now have:

```
//STEP1        EXEC   PGM=SYNCHRON,ROUTINE='MYPGM'
//DD1      DD  DSN=…
//CNTLFILE DD  DSN= footprint.file,DISP=SHR
//PARMFILE DD  * MYPARM
//
```

This means that the ENQ is done, MYPGM is called with a parameter of MYPARM and that the control file is then updated with the status of MYPGM (or any utility etc).

Using the ROUTINE= parameter is optional. If no routine name is given, the footprint dataset is simply marked with OK, time stamped, and SYNCHRON then terminates. It will still do the EXCLUSIVE ENQ though to ensure only one task updates the footprint file. This would be usable if you would rather have the utility that does the real work (for instance dump the SYSLOG) run as a next step in the procedure. The drawback of doing it this way is of course that you then have no further control if the dump utility should fail. By encapsulating the dump utility as in the example, our program will know whether it has been successful or not and update the first two bytes of the control file accordingly.

There is another optional parameter, MAXRC. This defines what being ‘OK’ actually means for the called program (MYPGM in the above example). For some programs RC=4 is acceptable, etc. If that is the case, we can have MAXRC=4.
A good application of this as mentioned would be the LOGREC facility, that nowadays can be run making use of the Sysplex LOGGER feature. The dump process should typically be run once a day, from any one but only one of the Sysplex members. Here is some sample JCL to make this work:

```plaintext
//DUMP EXEC PGM=SYNCHRON,REGION=5M,PARM='ROUTINE=IFCEREPI'
//STEPLIB DD DSN=PROD.LOADLIB,DISP=SHR
//SYSPRINT DD SYSOUT=* 
//CNTLFILE DD DSN=PROD.LOGFILE(TMESTAMP),DISP=SHR
//PARMFILE DD *
HIST,ACC=Y,SYSUM
//ACCDEV DD SPACE=(CYL,1),UNIT=3390
//ACCIN DD DSN=SYSPLEX.LOGREC.RECORDS, SUBSYS=(LOGR,IFBSEXIT), 
// DSB=(RECFM=VB,BLKSIZE=4000)
//SERLOG DD DUMMY
//DIRECTWK DD UNIT=SYSDA,SPACE=(CYL,15,,CONTIG)
//TOURIST DD SYSOUT=*,DCB=BLKSIZE=133
//EREPPT DD SYSOUT=*,DCB=BLKSIZE=133
//SYSIN DD DUMMY
```

SYNCHRON

**********************************************************************
** This program checks a control file to see if it contains "OK"
** followed by today's date. If it does, the program terminates with
** RC=4, else it does the following:
**
** - ENQ on RNAME=DATESTMP and QNAME= Name of CNTLINFO dataset
** - Checks the control file again for "OK" and today's date. 
**   If the control file does contain it, terminate.
**   If not, see if a routine name was passed with the parameter
**   ROUTINE=xxxxxxxx. If it was, call the routine.
** - Update the control file with date, JOBNAME, system name and
**   also the return code of the (optional) called routine.
**
** There is an optional parameter MAXRC that can be used to
** specify the acceptable RC of the called routine. If MAXRC is
** not specified, the default value for it is Ø.
**
** If there was no called routine, OK is put into the first 2
** characters of the control record.
** If the called routine has an RC = Ø or RC <= the optional
** MAXRC parameter, OK is put into the first 2 characters of
```
* the control record.
* If the called routine has an RC > the optional MAXRC
* parameter or the routine has abended, the characters AB
* are put in the first 2 bytes of the control record.
* 
* This program's return codes are:
* RC=Ø      Successful completion, control file updated
* RC=4     The control file has already been updated so the
*          task did not execute
* RC=16    The called routine abended or exceeded its maximum
*          allowable return code.
* 
* PARAMETERS: Optional MAXRC= and optional ROUTINE= parameters
* OUTPUT:     Updates control record
* AMODE:     31
* RMODE:     24
* Caller's mode: Any (This program does a BAKR / PR)
* Called routns: As per ROUTINE= parameter
* DD-cards:  CNTLFILE required, file to time-stamp and update
* Special regs: R4 = Pointer to input parameters
*                R11,R12= Base registers
*                R13= Pointer to general save area and work areas
**********************************************************************
SYNCHRON CSECT
SYNCHRON AMODE 31
SYNCHRON RMODE 24

BAKR R14,Ø               .Save caller's status
BALR R12,Ø               .Our base register
USING *,12               .Addressability to this program
*********************************************************************
*        Main driver routine
*********************************************************************
LR    R4,R1               .Preserve parm pointer
STORAGE  LA    R3,STORSIZE         .Size of storage to get and clear
STORAGE OBTAIN,LENGTH=(3),LOC=BELOW,SP=Ø
LR    R2,R1               .Point to getmained area
LA    R3,STORSIZE          .Length of storage to clear
XR    R9,R9               .Fill with binary zeroes
MVCL  R2,R8               .Propagate binary zeroes
USING STORAREA,R1         .Addressability to getmained area
ST    R13,SAVEAREA+4      .Backchain
DROP   R1                  
LR    R13,R1               .Address of getmained area
USING STORAREA,R13        .Addressability to getmained area
ST    R4,PARMADDR         .Starting address of passed parms
BAS   R14,GETPARMS        .Read & check the input parameters
C     R15,=F'4'           .Acceptable parms?
BH    RETURN              .No, get out
BAS   R14,CHKSTAMP        .Read & check the date stamp
L     R15,RETCODE         .Pick up the return code
LTR  R15,R15 .Already completed for today/ error?
BNZ  RETURN .Yes, get out

* More than 1 task can at this point intend to update the control file.
* The ENQ is used to serialize from this point on, to make sure that
* only one task calls the parm-supplied module and also to make sure
* that there are no concurrent updates to the PDS.
*
BAS  R14,GETENQ .Go ENQ on resource(time stamp file)
BAS  R14,CHKSTAMP .Read & check the date stamp (again)
L  R15,RETCODE .Pick up the return code
LTR  R15,R15 .Already completed for today/ error?
BNZ  RETURN .Yes, get out
BAS  R14,CALLROUT .Go call routine
BAS  R14,DEQ .DEQ resource

RETURN LR  R2,R13 .Pointer to storage area
L  R4,RETCODE .Pick up the return code
LA  R3,STORSIZE .Size of storage to free
STORAGE RELEASE,LENGTH=(3),ADDR=(2),SP=Ø
LR  R15,R4 .Reload return code
PR .Back to our caller

**********************************************************************
* This routine analyses the input parms
**********************************************************************

GETPARMS BAKR  R14,Ø .Preserve caller's status
L  R4,PARMADDR .Load parm pointer as at start
L  R4,Ø(R4) .Point to actual data
XR  R7,R7
ICM  R7,3,Ø(R4) .Length of passed parm
ST  R7,PARMLENG .Length of passed data
LTR  R7,R7 .Look at the length
BNP GETPARMX .No parms specified, get out

TODATA LA  R1,2(R4) .Where actual parm starts
AR  R1,R7 .Add the length
BCTR  R1,Ø .Correct length
ST  R1,LASTCHAR .Address of last byte of data
LA  R4,2(R4) .Skip over length field
ST  R4,FRSTCHAR .Address of first byte of data
MAXRCODE L  R7,PARMLENG .Length of passed parm
SH  R7,=H'6' .Length of 'MAXRC='
LTR  R7,R7 .Parm long enough?
BP  LENGK1 .Yes

LENGK1 L  R4,FRSTCHAR .Point to start of data
MAXRCLP CLC Ø(6,R4),=C'MAXRC=' .See if MAXRC parameter was spec.
BE  FNDFMAXRC .Yes, it was
LA  R4,1(R4) .No, bump up pointer
BCT  R7,MAXRCLP .Scan entire text
WTO 'SYNCHRON(I): MAXRC not specified',ROUTCDE=11
B  GETROUTN .Not specified, get out

FNDFMAXRC EQU *
LA  R8,6(R4) .Start of actual MAXRC value
XR   R10,R10          .Length of qualifier
SRCMAXRC EQU   *   
C    R8,LASTCHAR      .Have we reached the end?
BH   MAXRCEND         .Yes, MAXRC value ends here
CLI  Ø(R8),C','        .Do we have a comma
BE   MAXRCEND         .End of MAXRC value
LA   R10,1(R10)       .Add 1 to length
LA   R8,1(R8)         .Bump up search pointer
B    SRCMAXRC         .Redo the loop
MAXRCEND CH  R10,=H'1'  .Length must be at least 1 byte
BL   MAXRCWTO         .Invalid MAXRC value specified
CH  R10,=H'2'        .Length must not exceed 2
BH   MAXRCWTO         .Invalid MAXRC value specified
BCTR  R10,Ø           .Reduce by 1 to update move instr
STC  R10,MOVMAXRC+1    .Update the move instruction
MOVMAXRC MVC  MAXRC,6(R4)  .Move the MAXRC value in
CLI  MAXRC+1,X'ØØ'     .Did user only specify 1 byte?
BNE  PACKIT            .No
MVC  MAXRC+1,MAXRC     .Move 1 byte to the right
NI  MAXRC,X'ØØ'        .Clear the first byte
PACKIT PACK  DOUBLE,MAXRC .Pack the maximum return code
CVB  R1,DOUBLE         . and convert it to binary, then
STH  R1,MAXRC          . store it back in binary format
XR  R15,R15            .Clear the return code
ST  R15,RETCODE        .Plug it
B    GETROUTN          .Get out
MAXRCWTO WTO 'SYNCHRON(W): -Invalid MAXRC specified, ignored', RETCODE=11
XR  R15,R15            .Acceptable, clear the return code
ST  R15,RETCODE        .Plug it
GETROUTN L   R7,PARMLENG  .Length of passed parameter
SH  R7,=H'8'           .Length of 'ROUTINE='
LTR  R7,R7             .Long enough?
BNP  GETPARMX          .No
LENGOK2 L  R4,FRSTCHAR    .Point to first parm byte
ROUTNLPM CLC Ø(R8,R4),=C'ROUTINE=' SEE IF ROUTINE= WAS SPECIFIED
BE   FNDROUTN          .Yes, it was
LA  R4,1(R4)           .No, bump up pointer
BCT  R7,ROUTNLPM       .Scan entire text
WTO 'SYNCHRON(I): ROUTINE not specified',RETCODE=11
B    GETPARMX          .ROUTINE= not specified, get out
FNDROUTN LA  R8,8(R4)    .Start of actual ROUTINE= value
XR  R10,R10            .Length of qualifier
SRCROUTN C  R8,LASTCHAR  .Have we reached the end?
BH   ROUTNEND          .Yes, routine value ends here
CLI  Ø(R8),C','        .Do we have a comma
BE   ROUTNEND          .End of routine value
LA  R10,1(R10)       .Add 1 to length
LA  R8,1(R8)         .Bump up search pointer
B    SRCROUTN         .Redo the loop
ROUTNEND CH  R10,=H'1'  .Length must be at least 1 byte
BL RoutNWTO .Invalid routn value specified
CH R10,=H'8' .Length must not exceed 8
BH RoutNWTO .Invalid routn value specified
BCTR R10,0 .Reduce by 1 to update move instr
STC R10,MOROUTN+1 .Update the move instruction
MOVROUTN MVC ROUTINE,8(R4) .Move the routine value in
OC ROUTINE,=8X'40' .Make sure upper case & blanks
B GETPARMX .Get out
RoutNWTO WTO 'SYNCHRON(W): -Invalid ROUTINE= parameter specified', X
ROUTCDE=11
LA R15,12 .Unacceptable
ST R15,RETCODE .Plug it
GETPARMX PR .Reload caller's status
**********************************************************************
*        This routine analyses the existing time stamp in the file
**********************************************************************

CHKSTAMP BAKR R14,0 .Preserve caller's status
LA R1,JFCBAREA .Pointer to JFCB workarea
ST R1,EXLST .Plug the address into exit list
OIJ FCBAREA=07
RDJFCB CNTLFIL1
LTR R15,R15 .Is the DD-card present?
BZ GETDSNAM .Yes
WTO 'SYNCHRON(E): -CNTLFILE DD-card missing, cannot proceed.X
'ROUTCDE=11
LA R15,16 .Set return code to 16
ST R15,RETCODE .Plug it
B GETSTAMX .Get out
GETDSNAM LA R1,JFCBAREA .Point to JFCB data
USING INFMJFCB,R1
MVC RNAME,JFCBDSNM .Dataset name becomes RNAME
GETTIME TIME DEC, TIMEDATE, DATETYPE=DDMMYYYY, LINKAGE=SYSTEM
MVC WORK1(8), TIMEDATE+8 .First-half-of-byte workarea
NC WORK1,=4X'0F' .Turn off second half of each byte
MVC WORK2(8), TIMEDATE+8 .Second-half-of-byte workarea
NC WORK2,=4X'0F' .Turn off first half of each byte
TR WORK1,FRSTBYTE .Make first halves printable
TR WORK2,SECBYTE .Make second halves printable
LA R1,WORK1 .Area containing all first halves
LA R2,WORK2 .Area containing all second halves
LA R3,4 .Number of bytes in each
LA R4,PRTDATE .Where we will move it to
MOVELOOP MVC 0(1,R4),Ø(R1) .The first half of the byte
MVC 1(1,R4),Ø(R2) .The second half of the byte
LA R4,2(R4) .Pointer to target area up by 2
LA R1,1(R1) .Pointer to first half of bytes
LA R2,1(R2) .Pointer to second half of bytes
BCTR R3,MOVELOOP .Do for entire date (8 bytes)
LA R14,OPENCNTL
BSM 0,R14

OPENCNTL OPEN CNTLFIL1

TM CNTLFIL1+48,X'10'

BO GETREC

WTO 'SYNCHRON(E): -Could not open control file.',ROUTCDE=11

LA R15,16

ST R15,RETCODE

B CLOSCNTL

GETREC GET CNTLFIL1

LR R2,R1

USING CNTLINFO,R2

CLC LASTDATE,PRTDATE

BNE MUSTDO

CHKSTATS CLC STATUS,=C'OK'

BNE CHKDASH

WTO 'SYNCHRON(I): -CNTLFILE status indicates task has already run today.',ROUTCDE=11

LA R15,4

ST R15,RETCODE

B CLOSCNTL

CHKDASH CLC STATUS,=C'—'

BNE MUSTDO

TM GOTENQ,YES

BNO MAYDO

WTO 'SYNCHRON(I): -Previous task left control file in invalid status, file being reset',ROUTCDE=11

B MUSTDO

MAYDO LA R15,0

ST R15,RETCODE

WTO 'SYNCHRON(I): -CNTLFILE status indicates update currently in progress.',ROUTCDE=11

B CLOSCNTL

MUSTDO LA R15,0

ST R15,RETCODE

DONEx XR R15,R15

NODATA TM NEWFILE,YES

BO CLOSCNTL

WTO 'SYNCHRON(W): -Control file empty, will be initialized.'X ,ROUTCDE=11

OI NEWFILE,YES

CLOSCNTL CLOSE CNTLFIL1

GETSTAMX PR

**********************************************************************

This routine obtains the exclusive ENQ

**********************************************************************

GETENQ BAKR R14,0

MODESET MODE=SUP,KEY=ZERO .Required for wait on ENQ

ENQ (QNAME,RNAME,E,44,SYSTEMS),RET=HAVE

LTR R15,R15 .Did we get it (always should)
BE GETENOQ .Yes
WTO 'SYNCHRON(E): Unexpected ENQ failure',ROUTCD=11
ABEND ØØØ3,DUMP
GETENOQ OI GOTEQ,YES .Set the flag
PR .Reload caller's status
**********************************************************************
* This routine calls the dataset allocation routine
**********************************************************************
CALLROUT BAKR R14,Ø .Preserve caller's status
DROP R2
USING CNTLINFO,OUTREC
MVC STATUS,=C'OK' .Default status
CLC ROUTINE,=8X'Ø0' .Do we have a program to call?
BE CALLUPDT .No, get out
MVC STATUS,=C'-' .Mark file as in-progress
BAS R14,UPDTFILE .Update the file
LA R14,SETAB .Start from here after an abend
STM R1,R14,RUBLRSQS .Preserve all our registers
MVC RUBLIST+2(2),=B'Ø1111111111111ØØ' Reload r1-r13
ST R13,ESTAEPRM .R13 required by recovery routine
La R2,ESTAEPRM .Pass R13 contents as a parameter
ESTAEMAC ESTAE RECOVER,PARAM=((2)),ASYNCH=NO
LA R1,JFCBAREA .Pointer to JFCB workarea
ST R1,EXLST .Plug the address into exit list
OI EXLST,X'Ø7'
RDJFCB PARMFILE .See if we have a parmfile DD-card
LTR R15,R15 .Is the DD-card present?
BZ READPARM .Yes
NOPARM LA R1,NULLFLD .Point to a null field
B LINKPNT .Go call the routine
READPARM OPEN (PARMFILE,INPUT)
GETLOOP GET PARMFILE .Get the first record
CLC Ø(6,R1),=C'NOPARM' ."noparm" specified?
BE NOPARM .Yes
CLC Ø(2,R1),=C'/*' .Comment?
BE GETLOOP .Yes, ignore
CLI Ø(R1),C' ' .Blank?
BE GETLOOP .Yes, ignore
MVC CALLDATA,Ø(R1) .Move the data in
ENDDATA CLOSE PARMFILE
LA R1,CALLDATA .Start of data
LA R2,8Ø .Length of data
BLNKLOOP CLI Ø(R1),C' ' .Look for a space
BE PREPPNTR .Found
LA R1,1(R1) .Bump up the pointer
BCT R2,BLNKLOOP .Do for entire text
PREPPNTR LA R1,CALLPRM .Address of jcl-format parameter
ST R1,CALLPRM@ .Plug it into pointer
LA R1,8Ø .Length of input card
SR R1,R2
STH R1,CALLPLNG .Update the length field
LA     R1, CALLPRM@ 
LINKPNT LA R2, ROUTINE .Point to the name of routine
LINK EPLOC=((R2)), ERRRET=NOLINK
ST R15, RETCODE .Plug the return code
B SETSTATS .Update control file

NoLink WTO 'SYNCHRON(E): -Could not call module.', ROUTCDE=11
MVC STATUS, =C'AB'
LA R15, 12 .Set the return code to 12
ST R15, RETCODE .Plug it
B CALLUPDT .Go update control file

SETSTATS CH R15, MAXRC .Compare to maximum allowable rc
BH ERRWTO .Mark file as unsuccessful

SETOK MVC STATUS, =C'OK' .Indicate success
B CALLUPDT .Go update file

SETAB ST R15, RETCODE .Store the passed return code
WTO 'SYNCHRON(E): -An ABEND has been detected', ROUTCDE=11
ERRWTO WTO 'SYNCHRON(E): -Timestamp in file set to AB as return code from routine too high.', ROUTCDE=11
MVC STATUS, =C'AB' .Indicate failure
MVC RETCODE, =F'16' .Set the return code to 16
CALLUPDT BAS R14, UPDTFILE .Go update the control file
CALLROUX ESTAE Ø .Remove the estae routine
PR .Reload caller's status

**********************************************************************
*        This routine timestamps the file
**********************************************************************

UPDTFILE BAKR R14, Ø .Preserve caller's status
BAS R14, GETENV .Go get JOBNAME etc. into record
OPEN (CNTLFIL2, OUTPUT) .Did the file open?
TM CNTLFIL2+48, X'10' .Yes
BO PUTREC
WTO 'SYNCHRON(E): -Could not open control file for output', ROUTCDE=11
LA R15, 16 .Return code of 4
ST R15, RETCODE .Plug the return code
B UPDTFILX .Get out
PUTREC PUT CNTLFIL2, OUTREC .Write the record
CLOSE CNTLFIL2 .and close the file
UPDTFILX PR .Reload caller's status

**********************************************************************
*        This routine DEQ's the exclusive ENQ
**********************************************************************

DODEQ BAKR R14, Ø .Preserve caller's status
DEQ (QNAME, RNAME, 44, SYSTEMS), RET=HAVE
LTR R15, R15 .Did we release it (always should)?
BE DODEQX .Yes
WTO 'SYNCHRON(E): Unexpected DEQ failure', ROUTCDE=11
ABEND ØØØ3, DUMP
DODEQX PR .Reload caller's status

**********************************************************************
*        This routine gets our own JOBNAME & job number
**********************************************************************

GETENV  BAKR R14,Ø               .Preserve caller's status
MVC   LASTDATE,PRTDATE    .Move datastamp into record
MVC   LASTDATE,PRTDATE    .Move datastamp into record
L     R1,16               .Cvt address
DROP  R1
USING CVT,R1               .Addressability to cvt
MVC   SYSNAME,CVTSNAME    .Move system name into record
XR    R4,R4               .Psa starts at zero
USING PSA,R4
L     R5,PSATOLD          .Pointer to current tcb
USING TCB,R5               .Addressability to tcb
L     R6,TCBJSCB          .JSCB of this step
L     R5,TCBTO           .Pointer to TIOT
DROP  R5
USING TIOT1,R5              .Addressability to TIOT
LA    R2,ADDRSPC
EXTRACT ((2)), 'S', FIELDS=(TSO)
L     R2,ADDRSPC
LTR   R2,R2               .In use?
BNZ   BATCH
L     R4,PSAANEW          .Pointer to ascb
DROP  R4
USING ASCB,R4
L     R4,ASCBJBNS         .Pointer to JOBNAME
MVC   JOBNAME,Ø(R4)       .Move JOBNAME in
B     GETJSCB
BATCH  MVC   JOBNAME,TIOCNJOB    .Pick up our JOBNAME
USING IEZJSCB,R6          .Addressability to JSCB
GETJSCB  L     R6,JSCBACT          .Pointer to active JSCB
ICM   R6,15,JSCBSSIB      .Addr of subsystem id block
USING SSIB,R6             .Addressability to SSIB
MVC   JOBNUM,SSIBJBID     .Pick up our job number
GETENVX  PR                        .Reload caller's status
**********************************************************************
*        Constants follow
**********************************************************************
FRSTBYTE DS ØCL24Ø
DC    X'F0 ',15X'Ø0',X'F1',15X'Ø0',X'F2',15X'Ø0',X'F3'
DC    15X'Ø0',X'F4',15X'Ø0',X'F5',15X'Ø0',X'F6',15X'Ø0',X'F7'
DC    15X'Ø0',X'F8',15X'Ø0',X'F9',15X'Ø0',X'C1',15X'Ø0',X'C2'
DC    15X'Ø0',X'C3',15X'Ø0',X'C4',15X'Ø0',X'C5',15X'Ø0',X'C6'
SECBYTE DC X'F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6'
CNTLFIL1 DCB DDNAME=CNTLFILE,DSORG=PS,MACRF=GL,EODAD=NODATA, X
EXLST=EXLST
PARMFILE DCB DSORG=PS,LRECL=ØØ,MACRF=GL,EODAD=ENDDATA,EXLST=EXLST, X
DDNAME=PARMFILE
CNTLFIL2 DCB DDNAME=CNTLFILE,DSORG=PS,MACRF=PM,LRECL=ØØ
EXLST DS F
QNAME DC CLB'DATESTMP'
RNAME DS CL44
LTORG
Error recovery routine follows

```
DS  OF
RECOVER EQU *
BAKR R14,Ø
LR R12,R15
DROP R12
USING RECOVER,R12
CH R0,=H'12'
BNE SDWAVAIL
BNE NOTAVAIL XR R15,R15
SDWAVAIL EQU *
USING SDWA,R1
LR R3,R1
L R1,SDWAPARM
L R13,Ø(1)
ST R0,SDWASTOR
L R2,RUBLRSRG+52
LR R1,R3
LA R3,RUBLIST+2
SETRP RC=4,RETADDR=(2),RETREGS=YES,RUB=(3),DUMP=YES
RECOVERX PR

DSECTs follow
```

```
STORAREA DSECT
SAVEAREA DS 18F
RET.CODE DS F
PARMADDR DS F
NEWFILE DS F
TIMEDATE DS CL16
PRTDATE DS CL8
WORK1 DS CL4
WORK2 DS CL4
LASTCHAR DS F
FRSTCHAR DS F
PARMLENG DS F
ROUTINE DS CL8
MAXRC DS H
DOUBLE DS D
JFCBAREA DS CL176
OUTREC DS CL80
ADDRSPC DS F
SAVER14 DS F
NULLFLD DS F
SDWASTOR DS F
RUBLIST DS F
RUBLRSRGs DS 15F
GOTENQ DS C
```

ESTAEPRM DS F .Parameter passed to estae routine
CALLPRM@ DS F .Pointer to build jcl-style parm
CALLPARM DS ØF .Parm to call routine with
CALLPLNG DS H .Length of passed parameter
CALLDATA DS CL8Ø .Data area for parmfile
STORSIZE EQU *-STORAREA .Size of area to getmain/freemain
RØ EQU Ø
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R1Ø EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
YES EQU X'8Ø'
NO EQU X'ØØ'

CNTLINFO DSECT
STATUS DS CL2 .Can be OK, AB or — (active)
LASTDATE DS CL8 .Date file was updated
JOBNAME DS CL8 .Name of job that did last update
JOBNUM DS CL8 .Number of job that did last update
SYSNAME DS CL3 .Name of system that did last update

JFCBDESC DSECT ,
  IEFJFCBN
CVT DSECT=YES
IHASDWA
IHAPSA
IHAAASC
IEZJSCB
DSECT ,
IEFJSSIB
DSECT ,
IEFTIOT1
IKJTCB
END

Systems Programmer (UK) © Xephon 2000
Managing dynamic dump datasets

BACKGROUND
When MVS used dump datasets named SYS1.DUMP00 through DUMP99, it was easy to identify dumps and when they were taken by using the operator command ‘D D,T/E’ or the TSO command ‘SYSDSCAN’. However, this has changed since the arrival of dynamically allocated dump datasets, which were first provided in MVS/ESA SP 5.1. Even with OS/390 Version 2 Release 8, ‘D D,T/E’ will show only titles or error data for the latest dump produced, and even ‘D D,T,AUTODSN=ALL’ shows only information for the system dumps created since the last IPL.

To manage system dump datasets effectively we need to be able to identify dumps created, know their title, and when and where they were taken, regardless of whether an IPL has taken place since the dump dataset was allocated. The DUMPDS dialog described in this article makes this possible.

DYNAMIC DUMP DATASETS
Dynamic dump datasets are defined to the system using the DUMPDS command, DD for short. The benefits are that you no longer need to pre-allocate the space for system dumps, they no longer need be named SYS1.DUMPxx, and they can be managed by SMS. The datasets are allocated dynamically as and when required and the format of the name is selected by the installation to fit in with the site’s dataset naming standards, SMS, security, and the rest. MVS provides a variety of options, including system symbols, to be used when defining the name format, or pattern, and you can select from these options so that the dataset name tells you most of what you need to know about a dump. The dataset name can include the name of the system which created the dump, the date and time of creation, and a sequential number relating to the number of dumps created since the IPL. Full documentation is in the OS/390 MVS System Commands manual.
Once a system dump is generated and the dump dataset is allocated, you can use automation to notify the systems programmers or raise a problem record in your incident management software. Then you can use SMS and other storage management software to migrate or delete the datasets after use. However, the only supplied method to display information about dumps created since the last IPL is the operator command ‘DISPLAY DUMP’, and there is no supplied method of displaying data for dumps created prior to the IPL.

THE DUMPDS DIALOG
The following ISPF dialog provides a mechanism to list the dump datasets on a system and provides information about the title of the dump, the age, and whether it was created before or since the last IPL. It also makes it easy to delete these datasets manually, if required. The DUMPDS dialog consists of one REXX EXEC and five panels. It has been tested on OS/390 Version 2 Releases 5 and 8.

The exact information available will depend on the dump dataset name pattern in use. The section on customization describes what tailoring may be required but the rest of this article will describe the information when the dump dataset format is SYSMVS.DUMP.Dyymmdthhmmss.sysname.Snnnnn as specified in the operator command ‘DUMPDS NAME=SYSMVS.DUMP&YYMMDD..T&HHMMSS..&SYSNAME..S&SEQ’ This is the same as the system default name pattern except for the high-level qualifier.

DUMPDS REXX
The DUMPDS REXX EXEC drives the whole process. It finds all the system dump datasets from the system catalog using the ISPF library management services, and builds a temporary ISPF table of information relating to each dump. While this process is underway, panel DUMPD1 is displayed and, when the table is complete, panel DUMPD2 is displayed. DUMPD2 displays a scrollable list of all dumps found, as shown in the example below:
MVS1/MVS2 system dumps

DUMPDS Report 06/07/2000  Row 1 to 5 of 10
Command ---->  Scroll ----> CSR
MVS1: 3  MVS2: 7
Dump Datasets found . . : 10

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>Created</th>
<th>Volser</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSMVS.DUMP.D000705.T110128.MVS2.S00002</td>
<td>Wed 5 Jul 2000</td>
<td>M1T002</td>
</tr>
<tr>
<td>ABDUMP ERROR,COMPON=ABDUMP,COMPID=SCMP, ISSUER=IEAVTABD2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMPON=ASM,COMPID=SC1CW, ISSUER=ILRDRV01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSMVS.DUMP.D000704.T155330.MVS1.S00001</td>
<td>Tue 4 Jul 2000</td>
<td>M1T000</td>
</tr>
<tr>
<td>GTF TERMINATING ON ERROR CONDITION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSMVS.DUMP.D000701.T090000.MVS2.S00001</td>
<td>Sat 1 Jul 2000</td>
<td>M20012</td>
</tr>
<tr>
<td>TCPIP V2R5: Job(FTP ) ( ) +?????? S0C4/</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSMVS.DUMP.D000630.T071754.MVS2.S00002</td>
<td>Fri 30 Jun 2000</td>
<td>M2T003</td>
</tr>
<tr>
<td>COMPON=BPX,COMPID=SCPX1,ISSUER=BPXMIPC,MODULE=BPXVRSRB+???? ,ABEND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSMVS.DUMP.D000629.T023311.MVS2.S00001</td>
<td>Thu 29 Jun 2000</td>
<td>M2T005</td>
</tr>
<tr>
<td>(No title - dump older than 7 days)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSMVS.DUMP.D000623.T012933.MVS2.S00001</td>
<td>Fri 23 Jun 2000</td>
<td>M2T001</td>
</tr>
<tr>
<td>(No title - dump older than 7 days)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSMVS.DUMP.D000612.T192044.MVS2.S00004</td>
<td>Mon 12 Jun 2000</td>
<td>MIGRAT</td>
</tr>
<tr>
<td>(No title - dump migrated)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSMVS.DUMP.D000609.T155135.MVS1.S00002</td>
<td>Fri 9 Jun 2000</td>
<td>MIGRAT</td>
</tr>
<tr>
<td>(No title - dump migrated)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSMVS.DUMP.D000606.T164346.MVS1.S00004</td>
<td>Tue 6 Jun 2000</td>
<td>MIGRAT</td>
</tr>
<tr>
<td>(No title - dump migrated)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The information displayed is largely taken from the dataset name so is dependant on the dataset name format in use. More usefully, it includes the first portion of the dump title. It does this by opening each dump dataset and locating the title from there. The table is displayed in a pop-up window, but if you would prefer to have the panel displayed in the full screen size you can use the RESIZE command on the panel command line. To return to the pop-up format, use the RESIZE command again. RESIZE is a little-used ISPF command that can prove useful, especially when viewing a table in a pop-up window.
It can be used on any pop-up window with a command line, and even those without a command line when the RESIZE command has been assigned to a PF-key.

Recognizing that you may not want all dump datasets to be opened every time the DUMPDS dialog is run, thereby altering the last referenced date, there are two parameters available. ‘DUMPDS SHORT’ will display the dumps without their titles using panel DUMPD3, as shown below.

Another option is used to limit the display of titles to those dumps fewer than a specified number of days old. For example, ‘DUMPDS 5’ will display all dump datasets found, but only the titles for those fewer than five days old. As supplied, the dialog will also not attempt to open or allocate an archived or migrated dataset.

From panels DUMPD2 and DUMPD3 there are three line commands available:

- I – to display further information about the selected dump, including the full title and whether the dump was created before or since the last IPL – if created on the active system.
- B – to browse the dataset under ISPF.
- D – to delete the dataset.
The ‘I’ command displays the information about the dump in a pop-up window DUMPD4, as shown below. This display shows the full title of the dump as on the main panel the title field is truncated in order to fit on a single line:

```
MVS1 Dump details
Command ===> 

DSN. . . : SYSMVS.DUMP.D000630.T071754.MVS2.S00002
Title. . . : COMPON=BPX,COMPID=SCPX1,ISSUER=BPXMIPCE,MODULE=BPX
VRSRB+????,ABEND=S00C4,REASON=0000011

System . . : MVS1        Dump taken prior to IPL
Creation Date . : Fri 30 Jun 2000 Age . . : 6  days
Julian Date . : 2000.182   Time . . : 01:29:30

IPL Date . . : 2000.184    Time . . : 21:30:49
```

The ‘B’ command merely takes you into ISPF browse on the dump dataset. The D command brings up another pop-up, panel DUMPD5 shown below, asking you to confirm the dataset is to be deleted. Pressing <ENTER> will delete the dump.

```
DUMPDS Confirm Delete
Command ===> 

DUMP dataset to be deleted:

DSN. . . : SYSMVS.DUMP.D000614.T030619.MVS2.S00015
Title. . . : COMPID=?????,CSECT=????????+FFFF,DATE=????????,
Creation Date. . : 14 Jun 2000  System. . : MVS2

Enter-Confirm F12-Cancel
```

Syntax
The DUMPDS command takes a variety of parameters, summarized below:

- **DUMPDS** – take all default values.
- **DUMPDS SHORT** – show short display only (ie no dump titles).
- **DUMPDS n** – show titles, but only for dumps fewer than ‘n’ days old.
• DUMPDS BATCH – write dump details to SYSOUT (ie do not use ISPF panels).
• DUMPDS ? – will display the syntax explanation.
• DUMPDS DEBUG – will enable the REXX trace option.

The batch option displays the dump information in a report rather than an ISPF table. The sample JCL shown below can be used to run the DUMPDS dialog in batch. All parameters can be abbreviated and can be specified singly or together, in any order, with the following exceptions:

• ? – will display the syntax explanation and end – no further processing is done so any other parameters are superfluous.
• When a SHORT display is requested, the ‘n’ number of days parameter is meaningless as no titles will be displayed.

SAMPLE JCL

//jobcard JOB . . .
//*
//** DOC: RUN ISPF IN BATCH
//**
//S1 EXEC PGM=IKJEFT01,DYNAMNBR=300,REGION=8M,
// PARM='ISPSTART CMD(%DUMPDS BATCH)'
//SYSPROC DD DISP=SHR,DSN=SYS1.ISPF.CLIST
//ISPPLIB DD DISP=SHR,DSN=SYS1.ISPF.ISPPLIB
// ISPMLIB DD DISP=SHR,DSN=SYS1.ISPF.ISPMLIB
// ISPSLIB DD DISP=SHR,DSN=SYS1.ISPF.ISPSLIB
//ISPTLIB DD DDNAME=ISPTABL
// DD DISP=SHR,DSN=SYS1.ISPF.ISPTLIB
//ISPTABL DD DISP=(,DELETE),DSN=&ISPTABL,UNIT=SYSDA,
// LRECL=80,RECFM=FB,BLKSIZE=0,SPACE=(TRK,(1,1,1))
//ISPPROF DD DISP=(,DELETE),DSN=&ISPPROF,UNIT=SYSDA,
// LRECL=80,RECFM=FB,BLKSIZE=0,SPACE=(TRK,(1,1,1))
//ISPLIB DD DUMMY
//SYSTSPRT DD SYSOUT=*
• DUMPDS – REXX EXEC.
• DUMPD1 – ‘Please Wait’ message panel, while table is processing.
• DUMPD2 – Main panel showing dumps information for all dump datasets.
• DUMPD3 – Short display panel, similar to DUMPD2 but without the titles.
• DUMPD4 – Detail panel of information about a single dump.
• DUMPD5 – Delete confirmation panel.

CUSTOMIZATION
Before running this dialog, you must tailor it to fit your installation’s dump dataset name format. If your name standard includes date, time, and system name, you need only alter the DUMPDS REXX EXEC. If your dump dataset names do not include this information then you will need to make some modifications to the panels as well.

Instructions in the REXX EXEC, which may need to be tailored, can be identified in the code by the comment /*<— dsn format */. Just change these lines to specify the positions of the date, time, and system fields in your dataset name format. Dates displayed by this dialog are generally shown in the European format of day-month-year.

If you use DFDSShsm to manage dump datasets, the dialog is already set not to open any dump with a volser of ‘MIGRAT’ in order to avoid unwanted recalls. If, however, you use an alternative product or use a different volser to represent a migrated or archived dataset, you must change the constant MIGVOL. This constant is found at the beginning of the REXX EXEC in a section entitled DEFAULT VALUES. This is where all defaults are coded and each is described with a comment. Change these as desired.

To assist in testing any changes, use the DEBUG option – ‘DUMPDS DEBUG’ executes the REXX TRACE instruction to display each instruction as it processes.

The DUMPDS REXX EXEC assumes that the panels are available in the ISPPLIB concatenation. If this is not the case then you will need to add your own LIBDEF statement to make the panels available.
ENHANCEMENTS

As written, the DUMPDS dialog uses a temporary ISPF table to store the data for the duration of the dialog. It could be enhanced to build a permanent table improving performance and providing an easy record of all dumps that have occurred on a number of systems.

Another suggested enhancement would be to add an option to the DUMPD2/3 panels to take the user into IPCS to examine the dump, and perhaps an option to generate a batch job to copy the dump to tape to send to a vendor.

To make the dialog more portable, you could set a variable to the dataset name format as specified in the DUMPDS operator command, and then analyse the format in the DUMPDS REXX EXEC. This would avoid hard-coding the high level qualifier and the positions of the date, time, and system fields.

DUMPDS REXX EXEC

/*— REXX ————————*/
/* DUMPDS: 
* DUMPDS displays all cataloged MVS dynamic dump datasets, */
* dataset name, creation date and volser and, where */
* dump is less than five days old, the dump title. */
*/ */
*/ Syntax: */
*/ 'DUMPDS' with no parameter shows the default display */
*/ 'DUMPDS BATCH' shows display in-line, without ISPF panel */
*/ 'DUMPDS SHORT' shows the dsnames and dates only (no titles) */
*/ 'DUMPDS 2' shows titles less than two days old */
*/ 'DUMPDS DEBUG' traces REXX instructions */
*/ */
*/ Standard TSO rules of abbreviation apply */
*/ */
*/ Related components: */
*/ DUMPD1 'Please wait' message panel */
*/ DUMPD2 Main panel, full display with titles */
*/ DUMPD3 Short display panel */
*/ DUMPD4 Detail panel */
*/ DUMPD5 Delete confirmation panel */
*/ */
*/ Amendment History: */
*/ 1.Ø MMMyy ID First implementation */
*/ */

arg prm1 prm2 prm3 prm4 rest
titles = 'Y'
agelim = 4  /* age limit */
/* DD NAME=SYSMVS.DUMP.D&YYMMDD..T&HHMMSS..&SYSNAME..S&SEQ */
hlq = 'SYSMVS.DUMP'  /* dump dsname pfx */
migvol = 'MIGRAT'  /* migrated volser */
/* initialize counters */
d = Ø
numdump = Ø  /* number of dumps */
/* initialize constants */
if sysvar('SYSISPF') = 'ACTIVE' then ispf = 'Y'
address "ISPEXEC"
paname = 'DUMPD2'  /* panel name */
sysid = mvsvar('SYSNAME')
delparm = ' * * * deleted * * * '
syslist = ''
if sysvar('SYSISPF') ¬= 'ACTIVE' then
  do
    say 'DUMPDS Dialog requires ISPF environment'
  exit 16
  end
if ispf = 'Y' then "ISPEXEC CONTROL ERRORS CANCEL"
if batch ¬= 'Y' then  /* if NOT equal */
do
  /* indicate that process may take a few seconds */
  ADDRESS "ISPEXEC" "CONTROL DISPLAY LOCK"  /* lock terminal */
  if (rc > 8) then
    do
      /* error? */
      say 'Lock error   rc = ' rc  /* yes- message */
      exit(rc)  /* and quit */
    end
  ADDRESS "ISPEXEC" "ADDPOP ROW(6) COLUMN(10)"  /* pop up position */
  ADDRESS "ISPEXEC" "DISPLAY PANEL(DUMPD1)"
  if (rc > 8) then
do /* error? */  
say 'Display error rc = ' rc  /* yes- show message*/  
extit(rc)  /* and quit */
end

"ISPEXEC LMDINIT LISTID(IDVAR) LEVEL("hlq")"
if rc = Ø then /* if NOT equal */
do /* error? */
say 'LMDINIT error rc = ' rc  
extit(rc)  /* and quit */
end

"ISPEXEC LMDLIST LISTID("idvar") OPTION(LIST)", "DATASET(DSVAR) STATS(YES)"
listrc = rc

do while listrc = Ø /* process each dsn */
d = d + 1
dsn.d = dsvar
vol.d = zdlvol
"ISPEXEC LMDLIST LISTID("idvar") OPTION(LIST)", "DATASET(DSVAR) STATS(YES)"
listrc = rc
end

"ISPEXEC LMDLIST LISTID("idvar") OPTION(FREE)"
if rc = Ø then /* if NOT equal */
do /* error? */
say 'LMDLIST FREE error rc = ' rc  
extit(rc)  /* and quit */
end

"ISPEXEC LMDFREE LISTID("idvar")"
if rc = Ø then /* if NOT equal */
do /* error? */
say 'LMDFREE error rc = ' rc  
extit(rc)  /* and quit */
end

numdump = d
if numdump = Ø then /* no dumps found */
do
  zedlmsg = 'No dump datasets found'
  if ispf = 'Y' then /* use ISPF message */
    address ISPEXEC "SETMSG MSG(ISRZØØ1)"
  else say zedlmsg
  exit 4
end

/*————————————*/
/* Format dataset details */
do i = 1 to numdump
    yy.i = substr(dsn.i,14,2) /* <- dsn format */
    mm.i = substr(dsn.i,16,2) /* <- dsn format */
    dd.i = substr(dsn.i,18,2) /* <- dsn format */
    ymd.i = yy.i||'/'||mm.i||'/'||dd.i /* yy/mm/dd */
    ddat.i = date(,ymd.i,'O') /* dd Mon yyyy */
    dday.i = substr(date('W',ymd.i,'O'),1,3) /* Mon/Tue/Wed/etc. */
    sdat.i = date('S',ymd.i,'O') /* sorted yyyymmdd */
    bdat.i = date('B',ymd.i,'O') /* days since base */
    age.i = date('B') - bdat.i /* age of dump */
    jdat.i = subword(ddat.i,3)'.'right(date('D',ddat.i),3,'Ø') /* Julian yyyy.dddd */
    sys.i = substr(dsn.i,29,4) /* <- dsn format */
    sys = sys.i /* count of systems */
    if pos(sys,syslist) > Ø then do
        COUNT.sys = COUNT.sys + 1
        end
    else do
        syslist = syslist sys
        COUNT.sys = 1
        end
/* ______________________*/
/* Open dataset and extract dump title */
/* ______________________*/
if titles = 'Y' then do
    address "TSO"
    select
        when vol.i = migvol then, /* Don't open migrated dump */
            title.i = '(No title - dump migrated)'
        when (age.i > agelim) then, /* Don't open old dump*/
            title.i = '(No title - dump older than ' agelim 'days)'
        otherwise
            do
            "ALLOC FI(DUMPDS) DA('dsn.i') SHR REUSE"
            if rc ¬= Ø then, /* if NOT equal */
            title.i = '(No title - error allocating dump)'
            else,
                do /* read first record */
                "EXECIO 1 DISKR DUMPDS (stem line. finis"
                if rc ¬= Ø then, /* if NOT equal */
                title.i = '(No title - error opening dump)'
            else
                title.i = strip(substr(line.1,89,100))
        "FREE FI(DUMPDS)"
    end
end /* select */
address "ISPEXEC"
end
title.i = ''
end /* processing dump datasets */

/*—————————*/
/* Format report headers */
/*—————————*/
today = substr(date('S'),1,4)'.'right(date('D'),3,'Ø')
heading1 = strip(syslist) /* drop extra blanks*/
heading1 = translate(heading1,'/',' ') 
heading1 = heading1 'system dumps'
heading2 = ''
do i = 1 to words(syslist)
a = subword(syslist,i,1)
heading2 = heading2 a':' COUNT.a
end
heading2 = strip(heading2) /* drop extra blanks*/
heading3 = numdump 'dump datasets found'

/*—————————*/
/* IPL date/time */
/*—————————*/
IPLDATE:
smca = d2x(c2d(storage(10,4))+197) /* Get SMCA address */
ipicent = d2x(c2d(storage(smca,3))+340)
cent = d2x(c2d(storage(ipicent,1)))
cent = cent + 19 /* calculate century*/
iplyear = d2x(c2d(storage(smca,3))+341)
iyear = d2x(c2d(storage(iplyear,1)))
iyear = right(iyear,2,'Ø') /* force two digits */
ipiday = d2x(c2d(storage(smca,3))+342)
iday = d2x(c2d(storage(ipiday,2)))
iday = strip(iday,'F') /* remove sign */
iday = right(iday,3,'Ø') /* min. three digits*/
idade = cent||iyear.'iday /* IDATE variable */

IPLTIME:
ipltime = d2x(c2d(storage(smca,3))+336)
dec = c2d(storage(ipltime,4)) /* stored in binary */
ss = dec//60000
ss = ss%100
ss = right(ss,2,'Ø') /* make two digits */
mins = dec%60000
mm = mins//60
mm = right(mm,2,'Ø') /* make two digits */
hh = mins%60
hh = right(hh,2,'Ø') /* make two digits */
itime = hh':'mm':'ss                            /* ITIME variable   */

/***********/
/* Non-ISPF display    */
/***********/
if batch = 'Y' then
  do
    say heading1
    say heading2
    say heading3
    do i=1 to numdump
      say ''
      say i||'. ' dsn.i '-' ddow.i ddat.i '-' vol.i
      if titles = 'Y' then say '   ' title.i
      say ''
    end
  end
/***********/
/* ISPF panel display */
/***********/
else                                  /* put report into ISPF table */
  do
    rows = numdump                            /* no of rows in table */
    address "ISPEXEC"
    "CONTROL ERRORS RETURN"
    "TBCLOSE DUMPDTBL"
    tbnames = 'NAMES(ddow,ddat,dsn,vol,title,jdat,age,sys)'
    "TBCREATE DUMPDTBL" tbnames ,
    "NOWRITE"
    if rc = 8 then,                       /* try again, with REPLACE */
      "TBCREATE DUMPDTBL" tbnames ,
      "NOWRITE REPLACE"
    if rc ¬= Ø then                              /* if NOT equal     */
      do
        zedlmsg = 'Error creating table rc=' rc
        "SETMSG MSG(ISRZØØ1)"
        exit
      end
    do i = 1 to numdump                       /* — build table —*/
      ddow = ddow.i                             /* day of week      */
      ddat = ddat.i                             /* creation date    */
      dsn  = dsn.i                              /* dataset name     */
      vol  = vol.i                              /* DASD volser      */
      jdat = jdat.i                             /* Julian date      */
      age  = age.i                              /* age in days      */
      sys  = sys.i                              /* creation system  */
      if titles = 'Y' then title = title.i
      "TBADD DUMPDTBL MULT("rows")"
if rc ≠ Ø then                          /* if NOT equal */
do
zedlmsg = 'Error adding row' i 'to table rc=' rc
"SETMSG MSG(ISRZ001)"
exit
end
end
"TBTOP DUMPDTBL"
"TBSORT DUMPDTBL FIELDS(jdat,N.D,dsn,c,d)"  /* sort by date */
if rc ≠ Ø then                          /* if NOT equal */
do
zedlmsg = 'Error sorting table rc=' rc
"SETMSG MSG(ISRZ001)"
end
if words(syslist) > 1 then
zedlmsg = heading2
message = 'MSG(ISRZ001)'
cursor = 'CURSOR()'
csrrrow = 'CSRROW()'
ztdtop = '1'
panrc = Ø
"REMPPO"                     /* remove popup */
"ADDPOP"                      /* do until END key pressed */
/* do until END key pressed */
do while panrc < 8
"TBVCLEAR DUMPDTBL"
lcmd = ''                     /* nullify any pre-selected lines */
"TBSKIP DUMPDTBL NUMBER("ztdtop")"
"TBDISPL DUMPDTBL PANEL("paname")" message cursor csrrrow,
   "AUTOSEL(NO) POSITION(CRP)"
panrc = rc
zedmsmsg = ''
zedlmsg = ''
message = ''
cursor = ''
/* Process line commands */
/* Loop while there are selected rows */
do while ztdsels > Ø
   select
     when lcmd = 'D' then call DELCMD
     when lcmd = 'B' then call BROWSE
     when lcmd = 'I' then call INFO
     otherwise
       do
         zedmsmsg = 'Invalid select code'
         zedlmsg = 'Must be D (delete), B (browse) or I (info)'
   end
message = 'MSG(ISRZ001)'
cursor = 'CURSOR(LCMD)'
csrrow = 'CSRROW('CRP')'
end

if ztdsels > 1 then
    do
        "TBDISPL DUMPDTBL POSITION(CRP)"
    end /* process next row */
else
    ztdsels = Ø
    /* no rows left */
end

"TBTOP DUMPDTBL"
end /* of main panel display loop */

if panrc > 8 then
    do
        zedlmsg = 'Error displaying panel rc=' rc
        "SETMSG MSG(ISRZ001)"
        exit
    end

message = '
"TBVCLEAR DUMPDTBL"
lcmd = '
"TBEND DUMPDTBL"
if rc \= Ø then /* if NOT equal */
    do
        zedlmsg = 'Error with TBEND rc=' rc
        "SETMSG MSG(ISRZ001)"
        exit
    end
"REMPUP"
end /* of ISPF procesing */

EXIT

/*****
/* SUB- ROUTINES */
/*****

/*****
/* Process parameters passed */
/*****
VERPARM:
parse arg parm
parm = strip(parm)
select
when parm = '' then nop
when abbrev('DEBUG ',parm,1) then
do
  trace r
end
when abbrev('SHORT ',parm,1) then
do
  paname = 'DUMPD3'                         /* short panel name */
  titles = 'N'
end
when abbrev('BATCH ',parm,1) then batch = 'Y'
when datatype(parm,'N') then agelim = parm
when parm = '?' then
do
  say "DUMPDS displays all cataloged MVS dynamic dump datasets, "
  say "dataset name, creation date and volser and, where         "
  say "dump is less than five days old, the dump title.         "
  say "
  say "'DUMPDS' with no parameter shows the default display"
  say "'DUMPDS BATCH' shows display in-line, without ISPF panel "
  say "'DUMPDS SHORT' shows the dsnames and dates only (no titles)"
  say "'DUMPDS 2' shows titles less than 2 days old         "
  say "'DUMPDS DEBUG' traces REXX instructions               "
  say "
  say "Standard TSO rules of abbreviation apply               "
  exit
end
otherwise say parm 'invalid option - ignored'
end /* select */
return /*——————— end of VERPRM routine ————————*/

INFO:
drop cdate dtim dinfo title1 title2 title3

cdate = ddow ddat
  /* v— dsn format */
dtim = substr(dsn,22,2)':'substr(dsn,24,2)':'substr(dsn,26,2)
if sys ¬= mvsvsvar('SYSNAME') then,     /* if NOT equal x */
  dinfo = 'Dump not taken on this system'
else
  do
    if jdat > idate then dinfo = 'Dump taken after IPL'
  else dinfo = 'Dump taken prior to IPL'
  end
len1 = 50
len2 = 61
if length(title) <= len1 then
do
title1 = title
end
else
do
title1 = substr(title,1,len1)
title2 = substr(title,len1+1,len2)
if length(title) > len1+len2 then
title3 = substr(title,len1+len2+1)
end
zwinttl = sys 'Dump details'
"DISPLAY PANEL(DUMPD4)"
if rc > 4 then zedsmsg = 'Error displaying panel DUMPD4'
return /*——— end of INFO routine ————*/

/*-----------------------------*/
/* Browse DUMP dataset */
/*-----------------------------*/
BROWSE:
if dsn = delparm then
do
  zedsmsg = 'Already deleted'
return
end
"REMPOP"
"CONTROL DISPLAY SAVE"
"BROWSE DATASET(''dsn'')''" if rc > Ø then zedsmsg = 'BROWSE failed'
"CONTROL DISPLAY RESTORE"
"ADDPOP"
message = 'MSG(ISRZ001)'
csrrow = 'CSRROW('CRP')' return /*——— end of BROWSE routine ————*/

/*-----------------------------*/
/* DELETE a dump dataset */
/*-----------------------------*/
DELCMD:
if dsn = delparm then
do
  zedsmsg = 'Already deleted'
  zedlmsg = 'Cannot select this row - dataset already deleted'
end
else
do
  "CONTROL DISPLAY SAVE"
  "DISPLAY PANEL(DUMPD5)"
  if rc = Ø then
do
    address "TSO" "DELETE '"||dsn||"' PURGE"
    if rc = Ø then
do
dsn = delparm
title = ''
ddat = '——'
ddow = '—'
vol = '——'
sys = '——'
"TBPUT DUMPDTBL" /* update row */
zedmsg = 'Dataset deleted'
end
else zedmsg = 'Delete failed RC= ' rc
end
else zedmsg = 'Delete cancelled'
"CONTROL DISPLAY RESTORE"
end
message = 'MSG(ISRZ001)'
return /*——————— end of DELETE routine ———————*/

DUMPD1
The ‘Please Wait’ message panel.

)ATTR DEFAULT(%*_.)
  _ TYPE(INPUT)  INTENS(HIGH) COLOR(YELLOW)
  * TYPE(TEXT)  COLOR(RED)
  % TYPE(TEXT)  COLOR(YELLOW)
)BODY WINDOW(53,5)
%
% +——————————————————+
% | DUMP scan in progress. Please wait.   |
% +——————————————————+
%
)INIT
&ZCMD= ' '
&ZWINTTL= 'DUMPDS Scan'
)REINIT
&ZCMD= ' '
)PROC
)END
/* DUMPDS dialog:
/* DUMPD1 panel - initial message panel

DUMPD2
The main panel.

)ATTR
  + TYPE(TEXT) COLOR(BLUE)
  œ TYPE(TEXT) COLOR(GREEN) CAPS(OFF)
  @ TYPE(TEXT) COLOR(TURQ)  CAPS(OFF)
DUMPDS Report &ddmmyy4
@DUMPDS Report &ddmmyy4
@Command ===>_ZCMD                           / / @Scroll ===>_Z +
+ @Dump Datasets found . . :$numdump+
+ @ Dataset Name                             Created           Volser
+ ———————————   ———————   ———
)MODEL
?z#z                                        !z  %z          + !z
  #title
+ )INIT
  &ZWINTTL = &heading1
  &DVARS = '(ZSCML LCMD DSN DDOW DDAT VOL)'
  &DDMMYY4 = '&ZDAY/&ZMONTH/&ZSTDYEAR'
  VGET (ZSCML) PROFILE     /* Fill Scroll Vars if       */
  IF (&ZSCML = ' ') &ZSCML = 'PAGE'
)PROC
  VPUT (ZSCML) PROFILE
)END
/* DUMPDS dialog:
/*  DUMPD2 panel - main panel showing all dump information

DUMPD3
The short display panel.

)ATTR
+ TYPE(TEXT)   COLOR(BLUE)
@ TYPE(TEXT)   COLOR(GREEN) CAPS(OFF)
@ TYPE(TEXT)   COLOR(TURQ)   CAPS(OFF)
+ TYPE(INPUT)   COLOR(TURQ)
? TYPE(INPUT)   COLOR(TURQ)   CAPS(ON)
! TYPE(OUTPUT)   COLOR(BLUE)   CAPS(OFF)
% TYPE(OUTPUT)   COLOR(BLUE)   CAPS(OFF) JUST(RIGHT)
# TYPE(OUTPUT)   COLOR(GREEN)   CAPS(OFF)
$ TYPE(OUTPUT)   COLOR(TURQ)   CAPS(OFF)
+ TYPE(OUTPUT)   COLOR(WHITE)   CAPS(OFF) JUST(LEFT)
)BODY EXPAND(///) CMD(ZCMD) WINDOW(69,21)
@DUMPDS Report &ddmmyy4

@z#z

**DUMPDS**

The detail panel.

```plaintext
)PANEL KEYLIST(ISPSNAB,ISP)
)ATTR DEFAULT(%+_)
   _ TYPE(NEF) PADC(USER)
   + TYPE(TEXT) COLOR(GREEN)
   # TYPE(OUTPUT) INTENS(HIGH) COLOR(TURQ) CAPS(OFF)
   @ TYPE(OUTPUT) COLOR(WHITE) CAPS(OFF)
)BODY EXPAND(///) WINDOW(65,12)
+Command ===> _ZCMD
 +
 +  * DSN. . . :#dsn
 +  Title. . :#title1
 +  #title2
 +  #title3
 +  System . :#sys + @dinfo
 +  Creation Date . :#cdate + Age . . :#age +days
 +  Julian Date . :#jdat + Time . :#dtim +
 +
 +  IPL Date . . . :#idate + Time. . :#itime +
 +
)END
/* DUMPDS dialog:
/*    DUMPD4 panel - detailed dump information
```
DUMPDS

The delete confirmation panel.

)PANEL KEYLIST(ISPSNAB,ISP)
)ATTR DEFAULT(%+_.)
  _ TYPE(NEF) PADC(USER)
  @ TYPE(TEXT) COLOR(TURQ)
  + TYPE(TEXT) COLOR(GREEN)
  # TYPE(OUTPUT) INTENS(HIGH) COLOR(TURQ) CAPS(OFF)
)BODY EXPAND(///) WINDOW(65,9)
+Command ===> _ZCMD
+
+ DUMP dataset to be deleted:
+
+  DSN. . . . :#dsn  
+  Title. . . :#title  
+  Creation Date. . :#ddat  
+  System. . :#sys  
+
+  / / Enter-Confirm  F12-Cancel / /
)INIT
  &zwinttl = 'DUMPDS Confirm Delete'
)REINIT
)PROC
)END
/*  DUMPDS dialog:
/*
*     DUMPDS panel - delete confirmation panel

Moira Hunter
Systems Programmer (UK) © Xephon 2000

---

Subscribers to *MVS Update* will no doubt be interested to know that Xephon publishes a quarterly journal devoted exclusively to TSO and ISPF topics. If you have a site license, it is very easy to add a subscription to *TSO/ISPF Update*. To find out more, visit our Web site at www.xephon.com/tsoispfupdate.html, or contact us at any of the addresses on page 2 of this issue.
REXPDSM REXX function

This function returns the members of a PDS or PDSE, with or without the statistic information, in a stem variable.

This REXX function accepts three arguments, the third being optional. The first argument is either the dataset name of a PDS or PDSE (fully-qualified without quotes) or the name of a DD statement which has been allocated to a PDS or PDSE. If a DDname is used, the string FILE must also be coded following the DDname (this syntax is similar to the REXX LISTDSI function). The second argument is the name of a stem variable (with the terminating period) in which the member names will be returned. The number of member names, as is common REXX practice, will be stored in stem.0. The third, optional, argument specifies whether ISPF statistics are to be returned or not (statistics are not applicable to load libraries).

If the DDname option is used, the DD statement may have two or more datasets concatenated in the statement; the datasets may be any combination of PDS and PDSE and any combination of load and non-load libraries. If multiple datasets are concatenated, the second and subsequent identically-named members will be discarded.

Following execution, the content of the stem variable is dependent on the presence/absence of the third argument.

If an input dataset is a load library, or Stats is not requested, the content of each instance of the stem variable will be just the member name.

If an input dataset is not a load library and Stats is requested, the content of each instance will be:

```
name
vv
mm
created yyyy/mm/dd
changed yyyy/mm/dd
changed hh:mm:ss
size
init
mod
id
```
Each field is space delimited to aid in parsing. The syntax of the function is:

```
<--REXPDSM(--dsname------,---stem_name.----,---------)-->  
+ddname—FILE++                  +-Stats—+
```

In keeping with standard REXX practices, the Stats argument requires that only the first character be provided, and that character may be in upper or lower case. The function returns an integer. This integer will indicate success or failure. An example of the function being invoked:

```
RC = REXPDSM(STEPLIB FILE, STEM., S);
```

The different values that may be returned are as follows:

- 2 – IRXEXCOM - lack of storage
- 1 – IRXEXCOM - error condition
- 0 – Normal
- 4 – DDname specified > 8 bytes
- 8 – DSName specified > 44 bytes
- 12 – Stem name specified > 32 bytes
- 16 – No period at end of stem name
- 20 – Stem name contains invalid characters
- 24 – Stats specification invalid
- 28 – Invalid number of arguments
- 32 – File (DDname) open failure – possible incorrect DDname
- 36 – Catalog locate failure – possible incorrect DSName
- 40 – Allocation failure – possible sharing conflict.

An example of the function being used:

```
/* REXX ******************************************************/
address tso "ALLOC F(INPUT) DA(my.input.pds) SH REU";
RC = REXPDSM(INPUT FILE, MEMBER., S);
```
if RC = Ø then
    do I = 1 to MEMBER.Ø;
        say MEMBER.I;
    end;
address tso "FREE F(INPUT)"
exit;

This example shows, with statistics (if present), the members of the PDS or PDSE allocated to the DDname INPUT. Similar results may have been produced by:

/* REXX ******************************************************/
say "Enter PDS name:";
pull PDSNAME;
if substr(PDSNAME,1,1) <> "" then
    PDSNAME = sysvar(SYS_PREF)".PDSNAME;
else
    PDSNAME = substr(PDSNAME,2,length(PDSNAME)-2);
RC = REXPDSM(PDSNAME, MEMBER., Stats);
if RC = Ø then
    do I = 1 to MEMBER.Ø;
        say MEMBER.I;
    end;
exit;

When the DDname option is used, multiple datasets may be concatenated:

/* REXX ******************************************************/
address tso "ALLOC F(INPUT) DA(my.input.pds my.pdse) SH REU"
RC = REXPDSM(INPUT FILE, MEMBER., S);
if RC = Ø then
    do I = 1 to MEMBER.Ø;
        say MEMBER.I;
    end;
address tso "FREE F(INPUT)"
exit;

This will show the members of both input datasets. If the same member name occurs in both datasets, then only the member data from the first will be returned in the stem variable.

SOURCE
REXPDSM TITLE 'REXX FUNCTION RETURNING MEMBERS OF PDS OR PDSE'
PRINT NOGEN
*
* PROGRAM: REXPDSM
* RETURN IN STEM VARIABLE MEMBER DATA

© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
ATTRIBUTES:
    REENTRANT
    AMODE: 31
    RMODE: ANY

ABSTRACT:
REXX FUNCTION THAT DETERMINES ALL MEMBERS WITHIN THE PDS
OR PDSE ALLOCATED TO A SPECIFIED DDNAME OR DSNAME. IF THE
DDNAME IS NOT ALLOCATED OR THE DATASET IS NOT A PDS OR PDSE
THE STEM.Ø WILL BE INITIALIZED TO ZERO AND A RETURN CODE OF Ø.
THE NUMBER OF MEMBER NAMES RETURNED WILL BE STORED IN STEM.Ø

USAGE:
RETURN_CODE = REXPDSM(--DDNAME FILE--, STEM., --------_);
                        |             |       |         |
                        -DSNAME--------       ---STATS---

RETURN_CODE VALUES:
-2                      . IRXEXCOM - LACK OF STORAGE
-1                      . IRXEXCOM - ERROR CONDITION
0                        . NORMAL
4                        . DDNAME SPECIFIED > 8 BYTES
8                        . DSNAME SPECIFIED > 44 BYTES
12                       . STEM NAME SPECIFIED > 32 BYTES
16                       . NO PERIOD AT END OF STEM NAME
20                       . STEM NAME INVALID CHARACTERS
24                       . STATS SPECIFICATION INVALID
28                       . INVALID NUMBER OF ARGUMENTS
32                       . FILE OPEN FAILURE
-36                      . POSSIBLE INCORRECT DDNAME
-40                      . POSSIBLE INCORRECT DSNAME
-48                      . ALLOCATION FAILURE
-56                      . POSSIBLE SHARING CONFLICT

EJECT
TITLE 'EQUATES, MACROS && CONTROL BLOCKS USED'
R0   EQU   0                      . BAS RETURN REGISTER
R1   EQU   1
R2   EQU   2
R3   EQU   3
R4   EQU   4
R5   EQU   5
R6   EQU   6
R7   EQU   7
R8   EQU   8
R9   EQU   9
R10  EQU   10
R11  EQU   11
R12  EQU   12                      . CSECT BASE REGISTER
R13  EQU   13                      . -> DYNAMIC AREA
R14  EQU   14                      . -> RETURN
R15  EQU  15                      . -> ENTRY POINT
*  
*  * MACROS USED:
*  *  CLOSE                   . TERMINATE DATASET PROCESSING
*  *  DELETE                  . DECREMENT USAGE OF LOAD MODULE
*  *  DESERV                  . DIRECTORY ENTRY SERVICES
*  *  DYNALLOC                . DYNAMIC ALLOCATION
*  *  FREEPOOL                . RELEASE BUFFER POOL
*  *  IRXARGTB                . MAP ARGUMENT TABLE
*  *  IRXEPFL                 . MAP EXTERNAL FUNCTIONS PLIST
*  *  IRXEVALB                . MAP EVALUATION BLOCK
*  *  IRXSHVB                 . MAP SHARED VARIABLE BLOCK
*  *  LOAD                    . DYNAMICALLY LOAD MODULE
*  *  OPEN                    . OPEN DATASET
*  *  STORAGE                 . STORAGE ACQUIRE AND RELEASE
*  *  WTO                     . WRITE TO OPERATOR

EJECT

TITLE 'MAIN CSECT PROCESS'

REXPDSM  CSECT
REXPDSM  AMODE 31
REXPDSM  RMODE ANY

LA    R14,Ø(R14)             . VALIDITY OF R14
BSM   R14,R0                 . CURRENT ADDRESSING MODE
BAKR  R14,R0                 . ESTABLISH LINKAGE
LR    R12,R15                . 12 -> EPA
USING REXPDSM,R12            . CSECT ADDRESSABILITY
STORAGE OBTAIN,              . ACQUIRE DYNAMIC AREA
   ADDR=(R13),
   LENGTH=DYNLEN,
   LOC=(BELOW,ANY),
   SP=Ø
MVC   4(4,R13),=C'F1SA'       . INDICATE FORMAT OF SAVE AREA
USING DYNAREA,R13            . DSECT ADDRESSABILITY
XC    @IRXEXCOM,@IRXEXCOM    . INDICATE IRXEXCOM NOT LOADED
BAS   R10,REXXVECT           . REXX VECTOR PROCESSING
BAS   R10,ARGUMENT           . PROCESS ARGUMENTS
LTR   R8,R8                  . Q. ARGUMENTS VALID?
BNZ   AØØØ1                  . A. NO
BAS   R10,STEMDEL            . DELETE STEM
BAS   R10,PROCPSD            . PROCESS PDS

AØØØ1 EQU *
BAS   R10,TERMINAT           . TERMINATION
STORAGE RELEASE,             . RELEASE DYNAMIC STORAGE
   ADDR=(R13),
   LENGTH=DYNLEN,
   SP=Ø
SLR   R15,R15                . 15 - RETURN CODE
PR                             . ADIOS
PROCESS THE TWO ARGUMENTS PASSED TO REXX FUNCTIONS
THE ADDRESS OF THE REXX ENVIRONMENT BLOCK (OPTIONAL)
THE ADDRESS OF THE EXTERNAL FUNCTION PARAMETER LIST

REGISTER USAGE

0 . -> ENVIRONMENT BLOCK
1 . -> EXTERNAL FUNCTION PLIST
2 . -> PARSED PARAMETER LIST

ARGUMENT EQU *

L R2,ARGTAB . 2 -> ARGUMENT TABLE
USING ARGTABLE_ENTRY,R2 . DSECT ADDRESSABILITY
SLR R1,R1  . 1 - ZERO (ARGUMENT COUNT)
MVI STATS,OFF  . SET STATS OFF BY DEFAULT

CØØØ1 EQU *
*  . 4 -> ARGUMENT STRING
*  . 5 - ARGUMENT STRING LENGTH
LM R4,R5,ARGTABLE_ARGSTRING_PTR
LTR R5,R5  . Q. LENGTH NEGATIVE?
BM CØØØ8  . A. YES - LAST ARGUMENT
LA R1,1,(R1)  . INCREMENT ARGUMENT COUNT
CH R1,=H'1'  . Q. ARGUMENT TWO?
BNE CØØØ4  . A. YES
LA R6,Ø(R5,R4)  . 6 -> END OF FIRST ARGUMENT
SH R6,=H'5'  . BACK UP 5 BYTES
CLC =C' FILE',Ø(R6)  . Q. FILE SPECIFIED?
BNE CØØØ2  . A. NO
LA R8,4  . SET ERROR CODE
SH R5,=H'5'  . DECREMENT LENGTH
CH R5,=H'8'  . Q. ARGUMENT LENGTH > EIGHT?
BH CØØØ9  . A. YES - ERROR
MVI PROCFLAG,DDPROC  . SET DDNAME PROCESS
MVI DDNAME,C' '  . INITIALIZE DDNAME TO SPACES
MVC DDNAME+1(L'DDNAME-1),DDNAME
LA R6,DDNAME  . 6 -> DDNAME VALUE
B CØØØ3  . CONTINUE

CØØØ2 EQU *
*  . 2 -> NEXT ARGUMENT DATA
*  . 2 -> NEXT ARGUMENT DATA
LA R2,ARGTABLE_NEXT-ARGTABLE_ENTRY(,R2)
B CØØØ1  . PROCESS NEXT ARGUMENT

CØØØ3 EQU *
*  . 3 -> DDNAME/DSNAME
*  . 3 -> DDNAME/DSNAME
BCTR R5,RØ  . DECREMENT LENGTH FOR EXECUTE
EX R5,MVC  . SAVE DDNAME/DSNAME

CØØØ4 EQU *
*  . 4 -> DDNAME/DSNAME
CH R1,=H'2'  . Q. ARGUMENT THREE?
BNE CØØØ6  . A. YES
LA R8,12  . SET ERROR CODE
CH R5,=Y(L'STEM)  . Q. VARIABLE NAME TOO GREAT?
BH CØØØ9  . A. YES - ERROR
LA R6,Ø(R5,R4)       . 6 -> AFTER LAST BYTE OF NAME
LA R8,16           . SET ERROR CODE
BCTR R6,R0       . 6 -> LAST BYTE OF STEM NAME
CLI Ø(R6),C'.'    . Q. PERIOD PRESENT?
BNE C0009       . A. NO - ERROR
LA R8,20           . SET ERROR CODE
MVC STEM,SPACES   . INITIALIZE SAVED STEM VALUE
LA R6,STEM       . 6 -> SAVED STEM NAME VALUE
SLR R7,R7         . LENGTH OF STEM NAME

*  
C0005 EQU *  
*
SLR R3,R3       . 3 - ZERO
IC R3,Ø(R4)     . 3 - BYTE OF STEM VARIABLE
LA R3,TRTABLE(R3) . 3 - CHARACTER FROM TABLE
CLI Ø(R3),X'ØØ' . Q. VALID CHARACTER?
BE C0009       . A. NO
MVC Ø(1,R6),Ø(R4)   . MOVE BYTE TO SAVE STEM
LA R4,1(R4)     . 4 -> NEXT BYTE OF STEM NAME
LA R6,1(R6)     . 6 -> NEXT BYTE OF SAVED NAME
LA R7,1(R7)     . INCREMENT BYTES IN STEM NAME
BCT R5,C0005   . LOOP THROUGH STEM NAME
ST R7,#STEM    . SAVE LENGTH
*  
C0006 EQU *  
*
LA R2,ARGTABLE_NEXT-ARGTABLE_ENTRY(,R2)    . 2 -> NEXT ARGUMENT DATA
SLR R8,R8
B C0001       . PROCESS NEXT ARGUMENT

*  
C0007 EQU *  
*
MVI STATS,ON    . SET STATS ON
*
LA R2,ARGTABLE_NEXT-ARGTABLE_ENTRY(,R2)    . 2 -> NEXT ARGUMENT DATA
*
LA R4,R5,ARGTABLE_ARGSTRING_PTR    . 4 -> ARGUMENT STRING
*
LM R4,R5,ARGTABLE_ARGSTRING_PTR    . 5 - ARGUMENT STRING LENGTH
SLR R8,R8   . VALID RETURN
LTR R5,R5   . Q. LENGTH NEGATIVE?
BM C0009   . A. YES
LA R8,28                . SET ERROR CODE
B C0009                . OUT OF HERE
C0008 EQU *

CH R1,=H'2'            . Q. VALID NUMBER OF ARGUMENTS?
BE C0009                . A. YES
LA R8,28                . SET ERROR CODE

C0009 EQU *

DROP R2                . DSECT NOT REQUIRED
ST R8,RETCODE          . SAVE RETURN CODE
BR R1Ø

MVC *-**(R6),**(R4)    . EXECUTED MOVE
EJECT

TITLE 'DELETE ANY EXISTING STEM VARIABLE'

* LOAD REXX SERVICE ROUTINE IRXEXCOM
* SET UP PARAMETER LIST FOR IRXEXCOM
* INVOKE IRXEXCOM TO DROP STEM VARIABLE

* REGISTER USAGE
* Ø                . MACRO - EPA IRXEXCOM
* 1                . -> PARAMETER LIST
* 2                . -> SHARED VARIABLE BLOCK
* 3                . WORK
* 10               . RETURN
* 14               . CALL
* 15               . CALL

STEMDEL EQU *

LOAD EP=IRXEXCOM      . LOAD IRXEXCOM
ST RØ,@IRXCOM        . SAVE EPA
LA R2,SHVARBLK       . 2 -> SHARED VARIABLE BLOCK
XC ø(L'SHVARBLK,R2),ø(R2) . INITIALIZE
USING SHVBLOCK,R2    . DSECT ADDRESSABILITY

MVI SHVCODE,SHVDROPV  . SPECIFY ACTION
LA R3,STEM            . 3 -> STEM NAME
ST R3,SHVNAMA         . SAVE IN DSECT
MVC SHVNAML,#STEM     . LENGTH OF STEM NAME

LA R3,CIRXCOM        . 3 -> CHARACTER STRING IRXEXCOM
ST R3,@CSTR          . SAVE IN PARAMETER LIST
XC @DUMMY1(L'@DUMMY1+L'@DUMMY2),@DUMMY1
ST R2,@SHVB            . -> SHARED VARIABLE REQ BLOCK
OI @SHVB,X'80'        . FLAG END OF ARGUMENTS

L RØ,0REXX           . Ø -> REXX ENVIRONMENT BLOCK
LA R1,PIRXCOM        . 1 -> PARAMETER LIST
L R15,@IRXCOM        . 15 - EPA IRXEXCOM
BASSM R14,R15 . INVOKE IRXEXCOM

* LTR R15,R15 . Q. RETURN CODE LESS THAN ZERO?
BM D0001 . A. YES - ERROR
CH R15,-H'28' . Q. RETURN CODE 28?
BE D0001 . A. YES - ERROR
CH R15,-H'32' . Q. RETURN CODE 32?
BE D0001 . A. YES - ERROR
CLI SHVRET,SHVCLEAN . Q. EXECUTION OKAY?
BER R10 . A. YES - EXIT
CLI SHVRET,SHVNEWV . Q. NON-EXISTENT STEM?
BER R10 . A. YES - EXIT

* D0001 EQU *
*
DROP R2 . DSECT NOT REQUIRED
ST R15,RETCODE . SAVE 15
BR R10
EJECT

TITLE 'PROCESS PDS /POSE USING DIRECTORY ENTRY SERVICES'
* SET UP PARAMETERS FOR IRXEXCOM
* DETERMINE MODE - DDNAME OR DSNAME
* IF DSNAME - DYNAMICALLY ALLOCATE DATASET
* SAVE COUNT OF MEMBERS IN STEM.
*
* REGISTER USAGE
* 1 . WORK
* 2 . -> DCB
* 3 . WORK
* 4 . -> DESERV BUFFER
* 5 . WORK
* 6 . -> SHARED VARIABLE BLOCK
* 5 . -> SMDE IN DESERV BUFFER
* 6 . # OF SMDE IN DESERV BUFFER
*
PROCPS EQU *
*
ST R10,ESAVE . SAVE RETURN ADDRESS
LA R4,SHVARBLK . 4 -> SHARED VARIABLE BLOCK
XC 0(L'SHVARBLK,R4),0(R4) . INITIALIZE IT
USING SHVBLOCK,R4 . DSECT ADDRESSABILITY
MVI SHVCODE,SHVSYSET . SPECIFY ACTION
LA R1,NEWSTEM . 1 -> NEW STEM NAME
ST R1,SHVNAMA . SAVE IN DSECT
LA R1,L'STEMMEM . 1 - LENGTH OF MEMBER DATA
CLI STATS,OFF . Q. STATISTICS DATA REQUIRED?
BE E0001 . A. NO
LA R1,STEMDLEN . LENGTH OF MEMBER DATA
*
E0001 EQU *
*
ST R1,SHVVALL . SAVE IN DSECT
*
ZAP #VARS,=P'+'Ø' . INITIALIZE NUMBER OF VARIABLES
CLI PROCFLAG,DDPROC . Q. PROCESS BY DDNAME?
BE E0003 . A. YES - NO DYNALLOC
*
LA R2,2Ø+S99RBLEN . 2 - AMOUNT OF STORAGE NEEDED
*
* STORAGE OBTAIN, . ACQUIRE STORAGE *
ADDR=(R1),
LENGTH=(R2),
SP=Ø
ST R1,@S99AREA . SAVE ADDRESS OF STORAGE
USING S99RBP,R1 . MAP FIRST PART OF AREA
LA R2,2Ø(,R1) . 2 -> AREA FOR REQUEST BLOCK
ST R2,S99RBPTR . SAVE -> OF REQUEST BLOCK
OI S99RBPTR,S99RBPN . END OF PARAMETER LIST
DROP R1 . DSECT NOT REQUIRED
USING S99RB,R2 . MAP SVC99 REQUEST BLOCK
ST R2,@S99RB . SAVE ADDRESS OF REQUEST BLOCK
XC S99RB(S99RBLEN),S99RB . INITIALIZE REQUEST BLOCK
LA R3,S99RBLEN . 3 - LENGTH OF REQUEST BLOCK
STC R3,S99RBLN . SAVE LENGTH IN REQUEST BLOCK
LA R3,4(,R1) . 3 -> TEXT UNIT POINTERS
ST R3,S99TXTPP . SAVE ADDRESS IN REQUEST BLOCK
*
MVI S99VERB,S99VRBAL . INDICATE DSNAMES ALLOCATION
MVI S99FLG11,S99NOMNT+S99NOCNV . INDICATE NO MOUNT AND NO
* . USE OF EXISTING ALLOCATION
*
DROP R2 . REQUEST BLOCK NOT REQUIRED
*
L R1,@S99AREA . 1 -> ACQUIRED AREA
LA R3,4(,R1) . 3 -> TEXT UNIT POINTERS
USING S99TUPL,R3 . MAP THE TEXT UNIT POINTER
*
MVC DSNTUE,DSNTU . MOVE DSNAMES TU TO DYNAMIC
LA R2,DSNTUE . 2 -> DSNAMES TEXT UNIT
USING S99TUNIT,R2
MVC S99TUAPAR(L'DSNAME),DSNAME . OUTPUT DSNAMES
DROP R2
ST R2,S99TUPTR . SAVE IN TEXT UNIT POINTER
LA R3,L'S99TUPTR(,R3) . 3 -> NEXT TEXT UNIT POINTER
*
LA R2,STATTU . 2 -> STATUS TEXT UNIT
ST R2,S99TUPTR . SAVE IN TEXT UNIT POINTER
LA R3,L'S99TUPTR(,R3) . 3 -> NEXT TEXT UNIT POINTER
*
LA R2,DISPTU . 2 -> DISPOSITION TEXT UNIT
ST R2,S99TUPTR . SAVE IN TEXT UNIT POINTER
LA R3,L'S99TUPTR(,R3) . 3 -> NEXT TEXT UNIT POINTER
* MVC DDNTUE,DDNTU
  LA R2,DDNTUE
  ST R2,99TUPTR
  OI 99TUPTR,99TUPLN
  . MOVE DD NAME TU TO DYNAMIC
  . 2 -> DDNAME TEXT UNIT
  . SAVE IN TEXT UNIT POINTER
  . INDICATE LAST TEXT UNIT

* DROP R3
  . DSECT NOT REQUIRED

* DYNALLOC
  . ISSUE DYNAMIC ALLOCATION

* B *+4(R15)
  B EØØ2
  B EØ015
  B EØ019
  B EØ019
  . BRANCH DEPENDENT ON R15
  . RETURN CODE - Ø
  . RETURN CODE - 4
  . RETURN CODE - 8
  . RETURN CODE - 12

* EØØ2 EQU *

* LA R2,DDNTUE
  USING S99TUNIT,R2
  LH R3,S99TULNG
  MVC DDNAME,S99TUPAR
  DROP R2
  . 2 -> DYNAMIC TEXT UNIT
  . DSECT ADDRESSABILITY
  . 3 - LENGTH OF TEXT (DDNAME)
  . SAVE DDNAME USED

* EØØ3 EQU *

* MVC DCBE,DCBL
  LA R2,DCBE
  USING IHADCBE,R2
  MVC DCBDDNAM,DDNAME
  . MOVE DCB TO DYNAMIC
  . 2 -> DCB
  . DSECT TO CHECK OPEN
  . INITIALIZE DDNAME

* MVC OPENE,OPENL
  OPEN ((R2),INPUT),
  MODE=31,
  MF=(E,OPENE)
  TM DCOFLGS,DCBOFOPN
  BNO EØØ2Ø
  DROP R2
  . MOVE LIST FORM TO DYNAMIC
  . OPEN
  . Q. OPEN OKAY?
  . A. NO

* MVC DESERVE,DESERVL
  DESERV AREAPTR=DESERVBUFF,
  ISSUE GET_ALL
  CONCAT=ALL,
  DCB=(R2),
  FUNC=GET_ALL,
  RETCODE=DESERVRTRN,
  RSNCODE=DESERVREAS,
  SUBPOOL=Ø,
  MF=(E,DESERVE)
  CLC DESERVTRNR,+F'Ø'
  BE EØØ05
  CLC DESERVTRNR,+F'8'
  . MOVE TO DYNAMIC
  . ISSUE GET_ALL
  . Q. RETURN CODE VALID?
  . A. YES - PROCESS BUFFER
  . Q. RETURN CODE 8?
BNE E0004 . A. NO
CLC =H'1012',DESERVREAS+2 . Q. REASON CODE 1012?
BE E0010 . A. YES - NO MEMBERS

* E0004 EQU * . DESERV ERRORS
* 
MVC WTODESE,WTODESL . MOVE WTO TO DYNAMIC
MVC WTODESE+30(L'MASK5),MASK5 . MOVE IN EDIT MASK
MVC WTODESE+48(L'MASK5),MASK5 . MOVE IN EDIT MASK
L R1,DESERVRTRN . 1 -> RETURN CODE
CVD R1,DWORD . CONVERT TO DECIMAL
ED WTODESE+30(L'MASK5),DWORD+5 . AND OUTPUT
LH R1,DESERVREAS+2 . 1 -> REASON CODE
CVD R1,DWORD . CONVERT TO DECIMAL
ED WTODESE+48(L'MASK5),DWORD+5 . AND OUTPUT
WTO MF=(E,WTODESE) . DO THE WTO
B E0010 . CLOSE AND EXIT

* E0005 EQU * . DESERV - RETURN CODE Ø
* 
L R3,DESERVBUFF . 3 -> BUFFER AREA HEADER
LTR R3,R3 . Q. PRESENT?
BZ E0010 . A. NO - EOF
USING DESB,R3

* E0006 EQU * . PROCESS BUFFER
* 
LA R5,DESB_DATA . 5 -> START OF DATA
L R6,DESB_COUNT . 6 - ENTRIES IN BUFFER
USING SMDE,R5

* E0007 EQU * . PROCESS BUFFER
* 
TM SMDE_FLAG,SMDE_FLAG_ALIAS . Q. ALIAS ENTRY?
BO E0009 . A. YES - IGNORE
LH R1,SMDE_NAME_OFF . 1 -> NAME OFFSET
LA R1,SMDE(R1) . 1 -> NAME LENGTH
LA R1,2(,R1) . 1 -> NAME
MVI STEMDATA,C' ' . INITIALIZE STEM DATA
MVC STEMDATA+1(STEMDLLEN-1),STEMDATA
MVC STEMMEM.Ø(R1) . SAVE MEMBER NAME
TM SMDE_FLAG,SMDE_FLAG_LMOD . Q. LOAD MODULE?
BO E0008 . A. YES - NO STATS
CLI STATS,OFF . Q. STATISTICS REQUIRED?
BE E0008 . A. NO
CLC SMDE_USRD_LEN,=H'0' . Q. USER DATA LENGTH ZERO?
BE E0008 . A. YES - NO STATISTICS
LH R1,SMDE_USRD_OFF . 1 -> USER DATA OFFSET
LA R7,SMDE(R1) . 7 -> USER DATA
USING USERDATA,R7
*
SLR R8,R8                   . PREPARE FOR ICM
ICM R8,1,VV                  . PROCESS VERSION
CVD R8,DWORD
UNPK ZONWORK,DWORD+4(4)
OI ZONWORK+6,X'F0'          . ENSURE ABSOLUTE
MVC STEMVV,ZONWORK+5        . OUTPUT VERSION
*
ICM R8,1,MM                   . PROCESS MM
CVD R8,DWORD
UNPK ZONWORK,DWORD+4(4)
OI ZONWORK+6,X'F0'          . ENSURE ABSOLUTE
MVC STEMMM,ZONWORK+5        . OUTPUT MM
CLC SMDE_USRD_LEN,=H'5'     . Q. USER DATA < FIVE BYTES?
BL E0008                   . A. YES
*
ZAP PACKWORK,DATECR         . MOVE CREATE DATE TO WORK
BAS R10,PROCDATE           . DATE PROCESS
MVC STEMCDT,OUTDATE       . MOVE DATE TO OUTPUT
*
ZAP PACKWORK,DATECHG        . MOVE CHANGE DATE TO WORK
BAS R10,PROCDATE           . DATE PROCESS
MVC STEMMDT,OUTDATE       . MOVE DATE TO OUTPUT
*
XC PACKWORK,PACKWORK       . CHANGE TIME PROCESS
MVC PACKWORK(2),HHMM       . MOVE IN HOURS AND MINUTES
MVC PACKWORK+2(1),SECONDS . TACK ON SECONDS
L R8,PACKWORK             . PREPARE TO SHIFT
SRL R8,4                   . DIVIDE BY 10
ST R8,PACKWORK            . PUT IT BACK
OI PACKWORK+3,X'0F'        . TACK ON A SIGN
MVC OUTTIME,MASKTM         . MOVE MASK
ED OUTTIME,PACKWORK        . FORMAT THE TIME
MVC STEMMTM,OUTTIME+1      . MOVE INTO STEM
*
MVC STEMSIZE,MASK5         . PROCESS SIZE
SLR R8,R8                  . PREPARE FOR ICM
ICM R8,3,SIZE              . LOAD SIZE
CVD R8,DWORD               . PACK
ED STEMSIZE,DWORD+5        . OUTPUT
*
MVC STEMINIT,MASK5         . PROCESS INIT
ICM R8,3,INIT              . LOAD INIT
CVD R8,DWORD               . PACK
ED STEMINIT,DWORD+5        . OUTPUT
*
MVC STEMOD,MASK5          . PROCESS MOD
ICM R8,3,MOD               . LOAD MOD
CVD R8,DWORD               . PACK
ED STEMOD,DWORD+5         . OUTPUT
*
MVC STEMUSER,USERID       . OUTPUT MODIFIED BY
DROP R7

EØØ8 EQU *

LA R1,STEMDATA           . 1 -> STEM DATA
ST R1,SHVVALA            . SAVE IN DSECT
AP #VARS,=P'+1'          . INCREMENT VARIABLES PROCESSED

BAS R10,BLDVARNM         . BUILD VARIABLE NAME
MVC SHVNAML,#NEWSTEM     . LENGTH OF VARIABLE NAME
L R0,REXX                . Ø -> REXX ENVIRONMENT BLOCK
LA R1,PIRXEXCOM         . 1 -> PARAMETER LIST
L R15,IRXEXCOM          . 15 - EPA IRXEXCOM
BASSM R14,R15           . INVOKE IRXEXCOM

LTR R15,R15             . Q. RETURN CODE ZERO?
BZ EØØ9                 . A. YES - CONTINUE
CH R15,=H'1'           . Q. RETURN CODE ONE?
BZ EØØ9                 . A. YES - CONTINUE
ST R15,RETCODE         . SAVE RETURN CODE
B EØØ21

EØØ9 EQU *

AL R5,SMDE_LEN           . 5 -> NEXT ENTRY IN BUFFER
BCT R6,EØØ7             . PROCESS

LR R6,R3                . 6 -> BUFFER
L R7,DESB_LEN           . 7 - LENGTH OF BUFFER
ICM R3,15,DESB_NEXT     . 3 -> NEXT BUFFER
STORAGE RELEASE,        . RELEASE BUFFER
                        * ADDR=(R6),
                        * LENGTH=(R7)

LTR R3,R3               . Q. NEXT BUFFER PRESENT?
BNZ EØØ6                . A. YES - PROCESS
DROP R3
DROP R5

EØØ1Ø EQU *

MVC OPENE,OPENL         . MOVE LIST FORM TO DYNAMIC
CLOSE ((R2),),         . CLOSE
                        * MODE=31,
                        * MF=(E,OPENE)
FREEPOOL (R2)           . FREE BUFFER POOL

MVC VARWORK,MASK8       . MOVE EDIT MASK TO WORK AREA
ED VARWORK,#VARS        . EDIT THE DATA
LA R1,VARWORK           . 1 -> EDITED DATA
LA R2,L'VARWORK         . 2 - LENGTH OF EDITED DATA
* EØØ11 EQU *
 *
 CLI Ø(R1),C' ' . Q. SIGNIFICANT?
 BNE EØØ12 . A. YES
 LA R1,1(.R1) . 1 -> NEXT BYTE
 BCT R2,EØØ11 . LOOP
 EØØ12 EQU *
 ST R2,SHVVALL . LENGTH OF COUNT
 ST R1,SHVVALA . -> COUNT
 ZAP #VARS,=P'+'Ø' . INSTANCE NUMBER
 BAS R1Ø,BLDVARNM . BUILD VARIABLE NAME
 MVC SHVNAML,#NEWSTEM . LENGTH OF VARIABLE NAME
 DROP R4 . DSECT NOT REQUIRED
 L RØ,@REXX . Ø -> REXX ENVIRONMENT BLOCK
 LA R1,PIRXEXCOM . 1 -> PARAMETER LIST
 L R15,@IRXEXCOM . 15 - EPA IRXEXCOM
 BASSM R14,R15 . INVOKE IRXEXCOM
 LTR R15,R15 . Q. RETURN CODE ZERO?
 BZ EØØ13 . A. YES - CONTINUE
 CH R15,=H'1' . Q. RETURN CODE ONE?
 BZ EØØ13 . A. YES - CONTINUE
 ST R15,RETCODE . SAVE RETURN CODE
 B EØØ14 . CONTINUE
 EØØ13 EQU *
 CLI PROCFLAG,DDPROC . Q. PROCESS BY DDNAME?
 BE EØØ14 . A. YES - NO DYNALLOC
 L R2,@S99RB . 2 -> SVC99 REQUEST BLOCK
 USING S99RB,R2 . DSECT ADDRESSTABILITY
 MVI S99VERB,S99VRBU . INDICATE UNALLOCATION
 DROP R2 . REQUEST BLOCK NOT NEEDED
 L R1,@S99AREA . 1 -> ACQUIRED AREA
 LA R3,4(.R1) . 3 -> TEXT UNIT POINTERS
 USING S99TUPL,R3 . DSECT ADDRESSTABILITY
 LA R2,DSNTUE . 2 -> DSNAME TEXT UNIT
 ST R2,S99TUPTR . SAVE IN PARAMETER LIST
 OI S99TUPTR,S99TUPLN . INDICATE LAST TEXT UNIT
 DROP R3 . TEXT UNIT DSECT NOT REQUIRED
 DYNALLOC . DE-ALLOCATE FILE
 *
 B *+4(R15) . BRANCH DEPENDENT ON R15
 B EØØ14 . RETURN CODE Ø
 B EØØ15 . RETURN CODE 4
 B EØØ19 . RETURN CODE 8
 B EØØ19 . RETURN CODE 12
 *
 EØØ14 EQU *
 *
 L R1,@S99AREA . ADDRESS OF STORAGE
 LA R2,2Ø+S99RBLEN . 2 - AMOUNT OF STORAGE TO
 * . RELEASE
STORAGE RELEASE, . RELEASE STORAGE * 
ADDR=(R1), * 
LENGTH=(R2) * 
B E0021 . CONTINUE * 

E0015 EQU * 
LA R8,40 . SET RETURN CODE 
L R1,@S99RB . 1 -> SVC99 REQUEST BLOCK 
USING S99RB,R1 . DSECT ADDRESSABILITY 
LH R3,S99ERROR . 3 - DYNALLOC ERROR CODE 
CH R3,=H'+772' . Q. CLASS 2 ERROR? (SYSTEM * 
. RESOURCE NOT AVAILABLE) 
* 
BNL E0018 . A. NO 
* 
SH R3,=H'+516' . DECREMENT BY 516 - X'0204' * 
. X'0204' - START OF CLASS 2 
CH R3,=H'+12' . Q. GREATER THAN TWELVE? 
BH E0019 . A. YES - ABEND 
B *+4(R3) . BRANCH DEPENDENT ON R3 * 
. X'0204' 
B E0019 . ABEND - REAL STORAGE DEFICIT * 
. X'0208' 
* 
B E0019 . ABEND - RESERVED * 
. X'020C' 
* 
B E0016 . REQUEST FOR EXCLUSIVE USE * 
. CAN NOT BE HONORED * 
. X'0210' 
* 
B E0017 . DATASET NOT AVAILABLE - * 
. ALLOCATED TO ANOTHER USER * 

E0016 EQU * . DATASET USAGE CONFLICT 
WTO MF=(E,WTO1L) . FREE UP STORAGE 
B E0021 E0017 EQU * . DATASET ALLOC. TO OTHER TASK 
WTO MF=(E,WTO2L) . FREE UP STORAGE 
B E0021 E0018 EQU * . PROCESS CLASS 3, 4, 7 CODES 
CLI S99ERROR,X'17' . Q. CATALOG LOCATE FAILURE? 
BNE E0019 . A. NO - ABEND 
LA R8,36 . SET RETURN CODE 
WTO MF=(E,WTO3L) . FREE UP STORAGE 
B E0021 E0019 EQU * . ABEND TASK 
L R1,@S99RB . 1 -> SVC99 REQUEST BLOCK 
LR R2,R15 . 2 - CONTENT OF 15 
LH R3,S99ERROR . 3 - DYNALLOC ERROR CODE 
LH R4,S99INFO . 4 - DYNALLOC INFORMATION 
DROP R1 
ABEND 100, . DUMP * 

E0020 EQU * 
LA R8,32 . SET RETURN CODE
TITLE 'DEVELOP STEM NAME'
* CREATE STEM NAME FOR VARIABLE ABOUT TO BE ADDED
* TAKE SPECIFIED STEM AND APPEND THE OCCURRENCE NUMBER
* REGISTER USAGE
* 1 . -> INSTANCE NUMBER
* 7 . LENGTH OF NEW STEM
* 8 . LENGTH OF INSTANCE NUMBER
* 9 . LENGTH OF STEM
* . -> NEW STEM (COMPOUND)

BLDVARNM EQU *
MVC STEMQUAL, MASK8 . MOVE EDIT MASK TO WORK AREA
ED STEMQUAL, #VARS . EDIT THE DATA
LA R1, STEMQUAL . 1 -> EDITED DATA
LA R8, L'STEMQUAL . 8 - LENGTH OF EDITED DATA
F0001 EQU *
CLI Ø(R1), C' ' . Q. SIGNIFICANT?
BNE F0002 . A. YES
LA R1, 1(.R1) . 1 -> NEXT BYTE
BCT R8, F0001 . LOOP
F0002 EQU *
MVC NEWSTEM, SPACES . INITIALIZE NEW STEM
L R9, #STEM . NUMBER OF BYTES IN STEM
LR R7, R9 . 7 - SAME
BCTR R9, R0 . DECREMENT FOR EXECUTE
EX R9, MVCSTEM . MOVE STEM INTO NEW STEM
LA R9, NEWSTEM . 6 -> NEW STEM
LA R9, Ø(R7, R9) . 6 -> AFTER STEM IN NEW STEM
F0003 EQU *
MVC Ø(1, R9), Ø(R1) . MOVE COUNT BYTE BY BYTE
LA R1, 1(.R1) . 1 -> NEXT BYTE OF COUNT
LA R9, 1(.R9) . 6 -> NEXT BYTE OF COUNT
LA R7, 1(.R7) . INCREMENT LENGTH
BCT R8, F0003 . LOOP
ST R7, NEWSTEM . SAVE LENGTH
BR R1Ø

MVCSTEM MVC NEWSTEM(*-*), STEM
EJECT
TITLE 'TERMINATION ROUTINE'
* DELETE IRXEXCOM IF LOADED
* SET UP REXX FUNCTION RETURN CODE
* PUT RETURN VALUE INTO REXX EVALUATION BLOCK
* REGISTER USAGE
* 1 . LENGTH OF RETURN VALUE
* 2 . -> RETURN VALUE
* . -> EVAL BLOCK

* 3 . BINARY RETURN VALUE
* 4 . EVAL BLOCK SIZE
* 4 . LENGTH OF EDITED RETURN VALUE

TERMINATE EQU *

ICM R8,B'1111'.@IRXEXCOM . Q. IRXEXCOM LOADED?
BZ G0001 . A. YES
DELETE EP=IRXEXCOM . DECREMENT RESPONSIBILITY

G0001 EQU *

SLR R1,R1 . 1 - ZERO
LA R2,RCDATA . 2 -> OUTPUT DATA
MVC RCDATA,SPACES . INITIALIZE OUTPUT
L R3,RETCODE . 3 - RETURN CODE
LTR R3,R3 . Q. RETURN CODE NEGATIVE?
BNM G0002 . A. NO
MVI Ø(R2),C'-' . OUTPUT NEGATIVE SIGN
LA R1,1(.R1) . INCREMENT BYTES OUTPUT
LA R2,1(.R2) . 2 -> NEXT OUTPUT BYTE

G0002 EQU *

CVD R3,DWORD . PACK IT
MVC VARWORK,MASK8 . MOVE EDIT MASK TO WORK AREA
ED VARWORK,DWORD+4 . EDIT THE DATA
LA R3,VARWORK . 3 -> EDITED DATA
LA R4,L'VARWORK . 4 - LENGTH OF EDITED DATA

G0003 EQU *

CLI Ø(R3),C ' ' . Q. SIGNIFICANT?
BNE G0004 . A. YES
LA R3,1(.R3) . 3 -> NEXT BYTE
BCT R4,G0003 . LOOP

G0004 EQU *

MVC Ø(1,R2),Ø(R3) . MOVE OUT BYTE
LA R1,1(.R1) . INCREMENT BYTES OUTPUT
LA R2,1(.R2) . 2 -> NEXT OUTPUT BYTE
LA R3,1(.R3) . 3 -> NEXT INPUT BYTE
BCT R4,G0004 . LOOP
ST R1,#RCDATA . NUMBER OF BYTES
L R2,@EVALBLK . 2 -> EVAL BLOCK
USING EVALBLOCK,R2 . DSECT ADDRESSABILITY
L R3,EVALBLOCK_EVSIZE . 3 - LENGTH
CH R3,=H'3' . Q. AT LEAST THREE DOUBLES?
BL G0005 . A. NO
MVC EVALBLOCK_EVDATA(4),RCDATA . SET RESULT
MVC EVALBLOCK_EVLEN(4),#RCDATA
DROP R2

G0005 EQU *

BR R1Ø

TITLE 'SMDE DATE FIELD PROCESS'

AP PACKWORK,=P'+1900000' . SET CENTURY
UNPK  ZONEWORK,PACKWORK
MVC  OUTDATE(4),ZONEWORK  . OUTPUT YEAR
MVI  OUTDATE+4,C'/'
MVI  OUTDATE+7,C'/'
PACK  PACKYEAR,ZONEWORK(4)  . DETERMINE IF LEAP YEAR
DP  PACKYEAR,=P'+4'  . THIS SHOULD DO
LA  R8,1  . 8 - ONE
CP  PACKYEAR,=P'+Ø'  . Q. LEAP YEAR?
BE  HØØØ1  . A. YES
LA  R1,NORMYEAR  . 1 -> NORMAL YEAR
B  HØØØ2  . CONTINUE
HØØØ1  EQU  *
LA  R1,LEAPYEAR  . 1 -> LEAP YEAR
HØØØ2  EQU  *
CP  Ø(2,R1),PACKWORK+2(2)  . Q. THIS MONTH?
BH  HØØØ3  . A. YES
LA  R8,1(,R8)  . INCREMENT MONTH
LA  R1,2(,R1)  . 1 -> NEXT ENTRY
B  HØØØ2  . CONTINUE
HØØØ3  EQU  *
BCTR  R8,RØ  . DECREMENT MONTH NUMBER
CVD  R8,DWORD  . CONVERT MONTH TO DECIMAL
UNPK  ZONEWORK,DWORD+4(4)
OI  ZONEWORK+6,X'FØ'  . ENSURE ABSOLUTE
MVC  OUTDATE+5(2),ZONEWORK+5  . OUTPUT MONTH
SH  R1,=H'2'  . BACKUP ONE MONTH
SP  PACKWORK+2(2),Ø(2,R1)  . DETERMINE DAY
UNPK  ZONEWORK,PACKWORK+2(2)
OI  ZONEWORK+6,X'FØ'  . ENSURE ABSOLUTE
MVC  OUTDATE+8(2),ZONEWORK+5  . OUTPUT DAY
BR  R1Ø
DROP  R13
TITLE 'DYNAMIC AREA'

DYNAREA  DSECT
  DS    18F
  DWORD  DS    D  . FOR CVD
  ESAVE  DS    F  . REGISTER SAVE AREA
  @ARGTAB  DS    F  . -> ARGUMENT TABLE
  @EFPL  DS    F  . -> REXX EXT FUNCTION PLIST
  @EPAREA  DS    F  . -> EXTERNAL PARAMETER AREA
  @EVALBLK  DS    F  . -> EVAL BLOCK
  @IRXEXCOM  DS    F  . -> ENTRY POINT IRXEXCOM
  @REXX  DS    F  . -> REXX ENVIRONMENT BLOCK
  #NEWSTEM  DS    F  . LENGTH OF NEW STEM NAME
  #RCDATA  DS    F  . LENGTH OF RETURNED DATA
  #STEM  DS    F  . LENGTH OF STEM VARIABLE NAME
  RETCODE  DS    F  . RETURN CODE
  PIRXEXCOM  DS    ØF  . IRXEXCOM PARAMETER LIST
  @CSTR  DS    F  . -> CHARACTER STRING IRXEXCOM
  @DUMMY1  DS    F  . -> DUMMY ARGUMENT
  @DUMMY2  DS    F  . -> DUMMY ARGUMENT
@SHVB DS F . -> FIRST SHARED VARIABLE BLOCK
@S99AREA DS F . -> SVC 99 WORK AREA
@S99RB DS F . -> SVC 99 REQUEST BLOCK
DESERVBUFF DS F . DESERV BUFFER POINTER
DESERVRTRN DS F . DESERV RETURN CODE
DESERVREAS DS F . DESERV REASON CODE
DDNAME DS CL8 . DDNAME ARGUMENT VALUE
DSNAME DS CL44 . DSNAME ARGUMENT VALUE
PROCFLAG DS CL1 . PROCESS FLAG
DDPROC EQU C'D' . DDNAME SUPPLIED
DSNPROC EQU C'S' . DSNAME SUPPLIED
NEWSTEM DS CL44 . NEW STEM NAME
#VARS DS PL4 . NUMBER OF INSTANCES OF STEM
RCDATA DS CL8 . RETURN DATA
STATS DS XL1 . STATISTICS REQUESTED
OFF EQU X'ØØ'
ON EQU X'Ø1'
STEMQUAL DS CL8 . STEM QUALIFIER WORK
VARWORK DS CL8 . VARIABLE NUMBER WORK
DS ØF
PACKWORK DS PL4 . WORK - PACKED
ZONWORK DS CL7 . WORK DATE - ZONED
PACKYEAR DS PL4 . WORK YEAR
OUTDATE DS CL10 . FORMATTED JULIAN DATE
OUTTIME DS CL9 . FORMATTED TIME
DS ØF
STEMDATA DS ØF . STEM DATA
STEMMEM DS CL8 . MEMBER NAME
DS CL1 . DELIMITER
STEMVV DS CL2 . VERSION
DS CL1 . DELIMITER
STEMMM DS CL2 . MODIFICATION LEVEL
DS CL1 . DELIMITER
STEMCDT DS CL10 . CREATION DATE
DS CL1 . DELIMITER
STEMMDT DS CL10 . MODIFICATION DATE
DS CL1 . DELIMITER
STEMMTM DS CL8 . MODIFICATION TIME
STEMSIZE DS CL6 . SIZE
STEMINIT DS CL6 . INITIAL
STEMMOD DS CL6 . MOD
DS CL1 . DELIMITER
STEMUSER DS CL8 . MODIFIED BY
STEMDLEN EQU *-STEMDATA . LENGTH OF STEM DATA
DS ØF
SHVARBLK DS CL(SHVBLEN) . SHARED VARIABLE BLOCK AREA
DS ØF
DSNTUE DS CL(DSNTULEN) . DSNAME TEXT UNIT
DS ØF
DDNTUE DS CL(DDNTULEN) . RETURN DD NAME TEXT UNIT
DS  ØF
OPENE DS  CL(OPENLEN) . DATA MANAGEMENT LIST OPEN AREA
DS  ØF
DCBE DS  CL(DCBLEN)   . DATA CONTROL BLOCK
DS  ØF
DESERVE DS  CL(DESERVLEN) . DESERV GET_ALL
WTODESE DS  CL(WTODESLEN) . WTO WORK AREA
DYNLEN EQU  *-DYNAREA
TITLE 'IBM SUPPLIED DSECTS'
IRXARGTB . ARGUMENT TABLE
IRXEFPL . EXTERNAL FUNCTION PARAM LIST
IRXEVAB . EVALUATION BLOCK
IRXSHVB . SHARED VARIABLE REQUEST BLOCK
DCBD  DEVD=DA, . DCB MAPPING
DSORG=PS
IEFZB4DØ . SVC99 DSECTS
S99RBLEN EQU  (S99RBEND-S99RB) . LENGTH OF SVC99 REQUEST BLK
IEFZB4D2 . SVC99 EQUATES AND MNEMONICS
IGWDES . DIRECTORY ENTRY SERVICES
IGWSMDE . SYSTEM MANAGED DIRECTORY ENTRY
USERDATA DSECT
VV DS  XL1 . VV
MM DS  XL1 . MM
DS  XL1
SECONDS DS  XL1 . CHANGE SECONDS
DATECR DS  PL4 . CREATE DATE
DATECHG DS  PL4 . CHANGE DATE
HHMM DS  XL2 . CHANGE HOURS MINUTES
SIZE DS  XL2 . SIZE
INIT DS  XL2 . INITIAL SIZE
MOD DS  XL2 . MODIFIED
USERID DS  CL8 . USERID
TITLE 'LIST FORM MACROS, CONSTANTS'
REXPDSM CSECT
DS  ØF
DDNTU DC  AL(L'S99TUKEY)(DALRTDDN) . RETURN DD NAME SPEC TEXT UNIT
DC  XL(L'S99TUNUM)'Ø001'
DC  XL(L'S99TULNG)'Ø008' . LENGTH OF DD NAME
DS  CL8 . DATA DEFINITION NAME
DDNTULEN EQU  *-DDNTU
DS  ØF
DISPTU DC  AL(L'S99TUKEY)(DALNDISP) . DISPOSITION SPECIFICATION TU
DC  XL(L'S99TUNUM)'Ø001'
DC  XL(L'S99TULNG)'Ø001' . LENGTH OF DISP VALUE
DC  XLI'Ø8' . KEEP - MAY BE CHANGED
DS  ØF
DSNTU DC  AL(L'S99TUKEY)(DALDSNAM) . DSN SPECIFICATION TEXT UNIT
DC  XL(L'S99TUNUM)'Ø001'
DC  XL(L'S99TULNG)'Ø02C' . LENGTH OF DSNNAME
DS  CL44 . DSNNAME
DSNTULEN EQU  *-DSNTU
DS 0F
STATTU DC AL(L'S99TUKEY)(DALSTATS) . STATUS SPECIFICATION TU
DC XL(L'S99TNUM)'0001'
DC XL(L'S99TULNG)'0001' . LENGTH OF STATUS VALUE
DC XL1'08' . SHARE
OPENL OPEN (.INPUT),
MODE=31,
MF=L
OPENLEN EQU *-OPENL
DCBL DCB DDNAME=?????????, . FORM DCB
DSORG=PO,
EODAD=E0010,
MACRF=R
DCBLLEN EQU *-DCBL
DESERVL DESERV AREAPTR=*-* , . GET_ALL DESERV
DCB=*-* ,
FUNC=GET_ALL,
SUBPOOL=22,
MF=L
DESERVLEN EQU *-DESERVL . LENGTH
WTODESL WTO 'DESERV FAILED RETURN CODE XXXX REASON CODE XXXX',
ROUTCDE=11,
MF=L
WTODESLEN EQU *-WTODESL
WT01L WTO 'DATASET REQUESTED SHARED USE - CONFLICT',
ROUTCDE=11,
MF=L
WT02L WTO 'DATASET ALLOCATED TO ANOTHER USER(S)',
ROUTCDE=11,
MF=L
WT03L WTO 'DATASET NOT CATALOGED',
ROUTCDE=11,
MF=L
WT04L WTO 'DATASET OPEN FAILED',
ROUTCDE=11,
MF=L
MASK5 DC X'402020202120' . EDIT MASK FOR WTO/SIZE
MASK8 DC X'4020202020202120' . EDIT MASK FOR STEM
MASKTM DC X'2120207A20207A2020' . EDIT MASK FOR TIME
* SPACES DC 32C' ' . SPACES FOR INITIALIZATION
STATISTIC DC C'STATS' . ARGUMENT VALUE
CIRXEXCOM DC C'IRXEXCOM' . NAME OF REXX SERVICE ROUTINE
* TRTABLE DC 256X'00' . TRANSLATE TABLE
ORG TRTABLE+X'4B' . VALIDATE CONTENT OF STEM NAME
DC X'4B'
ORG TRTABLE+X'5B'
DC X'5B'
ORG TRTABLE+X'6D'
DC X'6D'
*  
NORMYEAR DC    PL2'0'                 . JANUARY
DC    PL2'31'                 . JANUARY
DC    PL2'59'                 . FEBRUARY
DC    PL2'90'                 . MARCH
DC    PL2'120'                . APRIL
DC    PL2'151'                . MAY
DC    PL2'181'                . JUNE
DC    PL2'212'                . JULY
DC    PL2'243'                . AUGUST
DC    PL2'273'                . SEPTEMBER
DC    PL2'304'                . OCTOBER
DC    PL2'334'                . NOVEMBER
DC    PL2'365'                . DECEMBER
LEAPYEAR DC    PL2'0'                 . JANUARY
DC    PL2'31'                 . JANUARY
DC    PL2'60'                 . FEBRUARY
DC    PL2'91'                 . MARCH
DC    PL2'121'                . APRIL
DC    PL2'152'                . MAY
DC    PL2'182'                . JUNE
DC    PL2'213'                . JULY
DC    PL2'244'                . AUGUST
DC    PL2'274'                . SEPTEMBER
DC    PL2'305'                . OCTOBER
DC    PL2'335'                . NOVEMBER
DC    PL2'366'                . DECEMBER
LTORG
END   REXPDSM

Dave Loveluck  
Consultant (USA)  © Xephon 2000
ASPG has announced version 4.2 of Megacryption, an OS/390 tool that provides a way to encrypt and decrypt any file in the MVS environment.

The principal new feature is the support of public-key encryption. This means that OS/390 sites can now exchange encrypted files without previously exchanging secret data (encryption keys). This works on a file-by-file basis, or even for multiple files packaged into one by DFSMSdss.

MegaCryption will also support PGP Version 6 and above. It is now possible to encrypt on OS/390 and decrypt on the PC with PGP (or on Unix with GnuPG).

Also supported: MD5 and SHA-1 for file integrity control, DH-ElGamal (1024, 1536 and 2048 bits) for public-key cryptography, passphrases ‘à la PGP’, CFB encryption mode, storage of keys into RACF, interface with DFSMSdss or DB2, etc.

For further information contact:
www.aspg.com
www.megacryption.com

* * *

Xepphon has published its new Report, OS/390: Strategies and Tactics. A ‘must’ for all MVS systems specialists, the report covers a whole range of performance and operational issues, and gives some initial views on z/OS.

Further details at:
www.xepphon.com/otrz.html

* * *

BEA has announced a technology and marketing alliance with SofTouch Systems, whose CrossPlex development and run-time toolkit accelerates the re-use of IBM 3270 applications by eliminating the need to modify legacy code.

As part of the deal, BEA and SofTouch have integrated the WebLogic Java Adapter for Mainframe 4.1 online EAI tool with CrossPlex for Web-enabling IBM 3270 legacy applications, including those whose source code is no longer available.

The combined products require no programming on the mainframe side, and only a minimal amount of data needs to be transmitted between server and mainframe.

The product resides on the mainframe and uses standard OS/390 services to interact with the online 3270 applications. All 3270 manipulations are done in memory on OS/390. And only the data requested by WebLogic Java Adapter for Mainframe is sent from the mainframe, not each and every 3270 buffer.

The announcement coincides with BEA’s launch of version 4.1 of WebLogic Java Adapter for Mainframe. New features include a configuration option that supports a Distributed Communications Resource Manager, said to enhance overall performance and optimize workload distribution between Java Adapter’s gateway and the mainframe itself.

For further information, contact:
www.bea.com

* * *