September 2003

In this issue

3 DFHSM error report from daily SMF extract
7 Archival data warehousing designs
10 Locating strings in files
22 Calling the ANTRQST macro
33 Using Pretty Good Privacy (PGP) to encrypt/decrypt and sign your MVS data
54 Using TAR and JAR files on MVS
56 Finding CSECTs within LPA load modules in virtual storage
63 Reorganization of datasets
70 October 1999 – September 2003 index
73 MVS news
Subscriptions and back-issues
A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1999 issue, are available separately to subscribers for £29.00 ($43.50) each including postage.

*MVS Update* on-line
Code from *MVS Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/mvs; you will need to supply a word from the printed issue.

© Xephon plc 2003. 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.*
After publishing the article entitled *A simple DFHSM report writer* (*MVS Update*, issue 191, August 2002) I received several requests to make the code more applicable. The key point of these requests was that the code was written in the SAS language, which is not available at all mainframe sites.

What was really needed was a simple, flexible, and yet fast reporting tool that would quickly identify work that has not been completed by DFHSM.

In order to provide a simple and effective way to gather the data needed to monitor, analyse, and correct what went wrong during DFHSM activity, an easy-to-use reporting procedure has been created.

**CODE**

The code is a two-part stream. In the first part, COPYSMF, selected DFHSM SMF records of the FSR type are copied from the SMF dataset to a file that can be used as a base for the archived records. By default, DFHSM writes SMF id 241 for functional statistics records (FSR and WWFSR). A detailed description of the layout of FSR records and its fields can be obtained from the DFSMS manual *DFSMShsm Implementation and Customization Guide*.

In the second part of the code, HSMERR, the captured records are formatted and two reports are produced.

The first report (DFHSM Function Error Statistics report) shows the summary of DFHSM errors along with a breakdown of this summary data by DFHSM function (BACKUP, SPILL, Primary to Level 1, etc), date, and return/reason codes, as well as the totals of the numbers of each error. The second report is a detailed dataset-level activity, reporting all activity that failed.

Note: a non-zero return code indicates that the function did not
complete successfully. The meanings of the return codes are documented in *DFHSM Messages*, SH35-0094. Use this field along with the function field to find out what function was running. Then, see message ARC0734I to determine what the return code means.

```plaintext
// DEL EXEC PGM=IDCAMS
// SYSPRINT DD SYSOUT=X
// SYSIN DD *
   DELETE hlq.SMF241
   SET MAXCC=0
/
// COPIESMF EXEC PGM=IFASMFDP, REGION=DM
// INDA1 DD DSN=sysid.SMFDUMPW, DISP=SHR  * weekly/daily SMF file
// OUTDA DD DSN=hlq.SMF241, DISP=(NEW,CATLG), UNIT=SYSDA,
//        SPACE=(CYL,(x,y),RLSE),
//        DCB=(sysid.SMFDUMPW)
// SYSPRINT DD SYSOUT=X
// SYSIN DD *
   INDD(INDA1, OPTIONS(DUMP))
   OUTDD(OUTDA, TYPE(241))
/
// HSMERR EXEC PGM=ICETOOL, REGION=DM
// TOOLMSG DD SYSOUT=X
// DFSMSG DD SYSOUT=X
// FUNCERR DD SYSOUT=X
// DETAIL DD SYSOUT=X
// SMF DD DSN=hlq.SMF241, DISP=SHR
// SHOW DD DSN=&&TEMPD, SPACE=(CYL,(15,15)), UNIT=SYSDA
// SMFC DD DSN=&&TEMPV, SPACE=(CYL,(15,15)), UNIT=SYSDA
// TOOLIN DD *
   COPY FROM(SMF) TO(SMFC) USING(SMFI)
*
*  Print a summary report showing total number of errors
*  OCCUR FROM(SHOW) LIST(FUNCERR)
  TITLE('DFHSM Function Error Statistic Report') DATE TIME
  HEADER('Function') ON(1,13,CH) -
  HEADER('Total number of errors') ON(VALCNT) -
  BLANK
*
*  Print a summary report showing number of errors by RC/RS
*  OCCUR FROM(SHOW) LIST(FUNCERR)
  TITLE('DFHSM Function Error Report by RC & RS') DATE TIME
  HEADER('Function') ON(1,13,CH) -
  HEADER('Ret. Code') ON(26,4,BI) -
  HEADER('Reas. Code') ON(30,4,BI) -
```
HEADER('Total number of errors') ON(VALCNT) BLANK

* Print a summary report showing number of function errors by date

OCCUR FROM(SHOW) LIST(FUNCERR)
TITLE('DFHSM Function Error Report by Function') DATE TIME
HEADER('Function') ON(1,13,CH)
HEADER('Date') ON(18,4,DT1,E'9999/99/99')
HEADER('Number of errors') ON(VALCNT)

* Print a summary report showing number of function errors by date

OCCUR FROM(SHOW) LIST(FUNCERR)
TITLE('DFHSM Function Error Report by Date') DATE TIME
HEADER('Date') ON(18,4,DT1,E'9999/99/99')
HEADER('Function') ON(1,13,CH)
HEADER('Number of errors') ON(VALCNT)

* Print a detailed report of function errors by date

DISPLAY FROM(SHOW) LIST(DETAIL)
TITLE('DFHSM Function Error Report - Detailed') DATE TIME
HEADER('Sid') ON(14,4,CH)
HEADER('Time') ON(22,4,TM1,E'99:99:99')
HEADER('Function') ON(1,13,CH)
HEADER('RC') ON(26,4,BI)
HEADER('REAS') ON(30,4,BI)
HEADER('Last Ref.') ON(34,4,DT1,E'9999/99/99')
HEADER('Last Moved') ON(38,4,DT1,E'9999/99/99')
HEADER('DS Age') ON(62,2,BI)
HEADER('Request by') ON(42,8,CH)
HEADER('From Vol.') ON(56,6,CH)
HEADER('To Vol.') ON(50,6,CH)
HEADER('Dsorg ') ON(116,4,CH)
HEADER('Data Set Name') ON(64,44,CH)
HEADER('Mclass') ON(108,8,CH)
BREAK(18,4,DT1,E'9999/99/99')
BTITLE(': Daily Error Report')
BLANK

/*
* //SMFICNTL DD *
* OPTION COPY, VLSHRT
* OUTFIL FNames=SHOW, CONVERT, * Build a new output record
* INCLUDE=(6, 1, BI, EQ, 'X'F1', AND, 109, 4, BI, GT, 'X'0000'),
* OUTREC=(43, 1, CHANGE=(13,
* 'X'01', 'C' Migrate l0/l1', * primary to level 1 migration

It is to be hoped that DASD/Storage Management personnel now have the ability to easily identify and analyse any action taken by DFHSM that failed.
Archival data warehousing designs

The underlying objectives of data warehouse designs can be summarized as:

- Storing a large volume of data.
- The fastest and cheapest retrieval of data.
- OLTP interface for user-friendly querying.
- Controlling system load.

Business applications running on legacy systems often need such a design to query archival/back-up data, without restrictions on size. Ad hoc reporting techniques available on MVS systems (see Handling ad hoc reports querying large volumes of data – a case study, MVS Update, issue 201, June 2003) can be used to build a primitive data warehouse that meets these underlying principles.

DATA ARCHIVAL DESIGN

Generation data groups can be used for taking back-ups daily. Because GDGs support a maximum of only 255 versions, we must collect data differently, based on a volume of data that we can browse sequentially. Below are two ways to do this.

- Monthly, appending data daily to the current version. We use a file layout with the date of archival as the file’s key, which helps to search data by date. This is when a month’s volume of data is suitable for sequentially browsing.

- Separate GDGs for each month with 28/29/30/31 versions based on month and year. We need two-tier JCL with INCLUDE groups (see previous article). First the JCL runs a program with built-in logic to read the date of the data archival/system date and converts it to a GDG version number. Thus the correct generation is located to store the back-up data. This information is passed to the second JCL
with the INCLUDE feature. The second JCL uses this INCLUDE member and stores the data. This pair runs daily to collect archival data. This JCL pair helps when a month’s volume is too great for sequential browsing during data retrieval. We can have as a file key the back-up hour too, for those applications where back-ups are taken hourly.

Thus the design of the data storage layer aims at zero maintenance by storing archival data in GDGs.

It directly influences the speed of data retrieval by:

- Using two-tier JCL with INCLUDE groups.
- Choosing a file key of date/hour.

Figure 1 shows a data archival design diagram.

```
Read system date
Convert to GDG version
Write to incld.ods(m1)
Submit JCL

//DD DSN H(-n)
Or
//EXEC SQL LOAD

//J1 JOB
//JCLLIB=incl.pds(m1)
//P1 EXEC PGM=LOAD
//IN1 INCLUDE=m1
```

**Figure 1: Data archival design diagram**

**DATA RETRIEVAL DESIGN**

As discussed above, we can embed in an ISPF/CLIST front end, built-in date-version conversion logic, prepare INCLUDE
members, and trigger the JCL to run using the INCLUDE dataset group. Such an application runs queries single-threaded. Using a design with two-tier JCL, using job scheduler switches (see previous article), means that JCL waits in the MVS input queue until the preceding job pair completes. This multi-threads keying-in requests. Batch query jobs still run single-threaded. By preparing separate INCLUDE members instead, a common member query can be multi-threaded.

Queries can be made much more flexible by preparing INCLUDE members with DB2 load control statements instead of JCL DD statements. Thus data warehousing designs can be created with much more advanced front-end designs on top of layers of the fastest and cheapest archival/retrieval designs.

![Data retrieval design diagram]

Data retrieval design is illustrated in Figure 2.

*Sreenivas Peddiraju*
*IT Analyst (India)*

© Xephon 2003
Locating strings in files

For most purposes, the Searchfor utility is an acceptable way to search for strings within files. However, it lacks support for VSAM files and a way to limit the scope of the search. For example, I need to search for a name that should exist in a name field, but eventually might also exist in other fields, like address, etc. If I know the position of the field I want to search within each record, then I may wish to restrict my search to that area. Or I may choose to search only a limited set of records, and not the entire file.

To satisfy these needs, I wrote a search program that accepts both sequential and VSAM files, allows the search zone to be limited to a range of columns within each record or to a range of records, permits the search string to be specified in hexadecimal, and also permits case to be ignored when performing comparisons (this applies only to strings specified as characters and to standard a-z characters, but you can change the translation table within the program, if you like).

The application consists of an Assembler program and a front-end formed by a REXX EXEC and an ISPF panel. It looks like this:

```
+---------------- Locate a string in a file ---------------------+
|                                                              |
|    File......: AARCF3.TEST.BA300                             |
|                                                              |
|    Search for: x'03f5f4c1a3                                  |
|       (Enter text string or begin with X' for hexadecimal)   |
|                                                              |
|       Ignore case - only valid for text (Y,N).: Y            |
|                                                              |
|    Search columns              Search records                |
|       First column.  12    First record.  10000              |
|       last column.  25      Last record.  15000               |
|                                                              |
|   Enter - execute                              PF3/15 cancel   |
+----------------------------------------------------------------+
```
In this example, we search for a hexadecimal string (since it begins with `x`, the ending quote is optional and can be omitted) with column boundaries 12 through to 25 and record boundaries 10,000 through to 15,000.

Since the string is hexadecimal, the *ignore case* field has no meaning. The program launches a job, and the result of the search can be found in sysprint: it states the record numbers where a match was found, the total number of record matches, and the total number of records searched:

String found in record number 000012034
String found in record number 000012035
String found in record number 000012036
Number of records where string found: 0000000003
Number of records searched...: 000005001

**LOCATE ASSEMBLER PROGRAM**

*====================================================================*
* LOCATE - Locates and counts the number of record matches of a      *
*          string in a file. The file can be sequential or VSAM.     *
*                                                                    *
* Format of the parameter received:                                  *
* Offset Name and meaning                                            *
*   0  Col1 initial position within records                          *
*   4  Col2 final position within records                            *
*   8  Rec1 initial search record                                    *
*  16  Rec2 final search record                                      *
*  24  Flag X-Hexa string  Y-ignore case  otherwise respect case    *
*  25  String to search.                                            *
*                                                                    *
* DDnames: Infile, Sysprint                                          *
*                                                                    *
* This program reads an input file and searches for a string in each *
* record. The search can be limited to a column range within each    *
* record (Col1 to Col2) or to a set of records (Rec1 to Rec2).       *
* If flag = 'X', string is represented in hexadecimal.               *
* If flag = 'Y', string and searched areas are uppercased before     *
* comparison takes place (case is ignored). This applies only to     *
* a-z standard EBCDIC characters.                                   *
*====================================================================*

&PROGRAM SETC 'LOCATE'
&PROGRAM AMODE 31
&PROGRAM RMODE 24
&PROGRAM CSECT
SAVE (14, 12)
LR R12, R15
USING &PROGRAM, R12
ST R13, SAVEA+4
LA R11, SAVEA
ST R11, 8(R13)
LR R13, R11
B GETPARMS
DC CL16'&PROGRAM 1.1'
DC CL8'&SYSDATE'

*====================================================================*
* Check and validate parameters                                      *
*====================================================================*
*
GETPARMS DS ØH
LR R2, R1            Copy parm pointer to R2.
L R2, Ø(Ø, R2)      Load parm address
LH R3, Ø(R2)        Load parm length in R3
OPEN (SYSPRINT, OUTPUT)  Open sysprint (for error msgs)
LTR R3, R3          Any parm entered?
BZ EXIT1            No, error
*
LR R6, R2
AR R6, R3           R6: point after end of parms
LA R6, 2(Ø, R6)     Skip 2 bytes of parmlength
LA R2, 2(Ø, R2)
XR R9, R9           Clear length counter
*
CONVERT EQU *
LA R9, 3            4 byte length parms
EX R9, EXPACK       Execute pack
CVB R7, PARMPACK    Convert to binary into R7
S R7, =F'1'        Turn limit to offset
ST R7, PARM1        And store it
*
LA R2, 4(Ø, R2)     Inc parm pointer
EX R9, EXPACK       Execute pack
CVB R7, PARMPACK    Convert to binary into R7
ST R7, PARM2        And store it
*
LA R2, 4(Ø, R2)     Inc parm pointer
LA R9, 7            8 byte length parms
EX R9, EXPACK       Execute pack
CVB R7, PARMPACK    Convert to binary into R7
ST R7, PARM3        And store it
*
LA R2, 8(Ø, R2)     Inc parm pointer
EX R9, EXPACK       Execute pack
CVB R7, PARMPACK  Convert to binary into R7
ST R7, PARM4  And store it

LA R2, 8(Ø, R2)  Inc parm pointer
MVC FLAG, Ø(R2)  Store flag parameter

LA R2, 1(Ø, R2)  Inc parm pointer to string
SR R6, R2  Length of string to search
SH R6, =H'1'  Length of string to search
EX R6, EXMOVE  Move to string
STH R6, STRINGL  Keep string length (-1)
CLI FLAG, C'X'  Hexadecimal string?
BNE VALIDPR  No, jump ahead

MVC STR1, STRING1  Convert string to real hexadecimal
MVC STR2, STRING2  characters in three 12 byte parts
MVC STR3, STRING3
NC STR1, XAND
TR STR1, XTRN
NC STR2, XAND
TR STR2, XTRN
NC STR3, XAND
TR STR3, XTRN
PACK STRP1, STR1(13)
PACK STRP2, STR2(13)
PACK STRP3, STR3(13)
MVC STRING(6), STRP1
MVC STRING+6(6), STRP2
MVC STRING+12(6), STRP3

LH R6, STRINGL  Get original length (-1)
LA R6, 1(Ø, R6)  Add 1
SRL R6, 1  Divide length by two
SH R6, =H'1'  Ready for comparisons (-1)
STH R6, STRINGL  Store it

VALIDPR EQU  Validate parameters

CLC PARM1, PARM2
BH ERRMSG1
CLC PARM3, PARM4
BH ERRMSG2
L R7, PARM2
S R7, PARM1
CR R6, R7
BH ERRMSG3

CLI FLAG, C'Y'  Ignore case specified?
BNE OPENACB1  No, jump ahead
LH R10, STRINGL  Load string length (-1)
LA R10, 1(Ø, R10)  Reset correct length
LA R5, STRING  Execute uppercase translation

*====================================================================*
* Check whether file is VSAM or sequential. If VSAM, check for ESDS *
*====================================================================*

OPENACB1 EQU *  Open ACB for VSAM input file
OPEN INFILEA  If error, go open the file as
LTR R15, R15  sequential.
BNZ OPENDCB1
TESTCB ACB=INFILEA,  File is VSAM, check for ESDS
ATRB=ESDS
BNE READFILE

ESDSFILE1 EQU *  ModCB RPL=INFILER, Set RPL for ESDS
MODCB RPL=INFILER,
     OPTCD=ADR  X
B READFILE

OPENDCB1 EQU *  Open sequential file
OPEN (INFILED, INPUT)
LTR R15, R15
BNZ ERRMSG4
MVI FILETYP1, C'S'  Set flag sequential
LA R2, INFILED  R2: IHADCB of input file
USING IHADCB, R2

*====================================================================*
* Read loops (VSAM loop or sequential loop) and compare subroutine *
*====================================================================*

READFILE EQU *  Record counter
XR R7, R7  First record
L R8, PARM3  Last record
CLI FILETYP1, C'V'  VSAM file?
BNE LOOPSEQ  No, go to sequential

*LOOPVSA EQU *  VSAM loop
LA R7, 1(Ø, R7)  Increment record counter
GET RPL=INFILER  Read VSAM file
LTR R15, R15  End of file?
BNZ EXITØ
CR R7, R8  First record attained?
BL LOOPVSA  No, read again
CR R7, R9  Last record attained?
BH EXITØ  Yes, exit

© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
* L R4, VAREA1 Get address of data in R4.
SHOWCB RPL=INFILER,
AREA=LRECL1,
LENGTH=4,
FIELDS=RECL
L R3, LRECL1 Get record length in R3
BAL R10, COMPARE Call compare
B LOOPVSA

* LOOPSEQ EQU * Sequential loop
LA R7, 1(Ø, R7) Increment record counter
GET INFILED Read sequential
LR R4, R1 R4: address of record
CR R7, R8 First record attained?
BL LOOPSEQ No, read again
CR R7, R9 Last record attained?
BH EXITØ Yes, exit
LH R3, DCBLRECL Load R3 with record length.
BAL R10, COMPARE Call compare
B LOOPSEQ

* COMPARE EQU * Compare subroutine
L R5, PARM1 Left limit (offset)
L R6, PARM2 Right limit
CR R3, R6 Record smaller than last position?
BNE COMPARE9 No, jump ahead
BNE COMPARE3 If strings not equal, exit

* COMPARE9 EQU *
SH R6, STRINGL R6: number of searches in the line
C R6, =F'Ø' If R6<Ø, string is greater than
BNH COMPARE3 search area, so skip this record

* COMPARE1 EQU *
LH R11, STRINGL length of search string
EX R11, EXCOMPAR Execute compare
BNE COMPARE2 If strings not equal, exit
LR R0, R7
BAL R11, UNPACK
MVC MSGFND2, OUT1Ø
LR R0, R9
PUT SYSPRINT, MSGFND String found, send message
L     R11, TOTFOUND     Inc rec found counter  
LA    R11, 1(Ø, R11)    
ST    R11, TOTFOUND     
B     COMPARE3          And return  

*  
COMPARE2 EQU  *     
LA    R5, 1(Ø, R5)      Increment compare position  
BCT   R6, COMPARE1      Loop to next  

*  
COMPARE3 EQU  *     
BR    R10              return  

*  
*====================================================================*  
* Send final messages, close files, and exit                        *  
*====================================================================*  

*  
EXITØ    EQU  *     
L     RØ, TOTFOUND    
BAL   R11, UNPACK     
MVC   MSGTOT2, OUT1Ø  
PUT   SYSPRINT, MSGTOT  
SR    R7, R8         
LR    RØ, R7         
BAL   R11, UNPACK     
MVC   MSGFIM2, OUT1Ø  
PUT   SYSPRINT, MSGFIM  

*  
EXIT1    EQU  *     
CLOSE INFILED      
CLOSE INFILEA     
CLOSE SYSPRINT     
L     R13, SAVEA+4    
LM    R14, R12, 12(R13)  
XR    R15, R15  
BR    R14  

*  
*====================================================================*  
* Other subroutines, execute instructions and work areas            *  
*====================================================================*  

*  
EXCOMPAR EQU  *     
CLC    Ø(Ø, R5), STRING  

*  
EXMOVE EQU  *     
MVC    STRING, Ø(R2)  

*  
EXPACK EQU  *     
PACK   PARMPACK, Ø(Ø, R2)  

*  
EXTRAN EQU  *  

© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
TR Ø(Ø, R5), TRTAB

UNPACK EQU *
CVD RØ, REGDECIM
UNPK OUT12, REGDECIM
BR R11

ERRMSG1 EQU *
PUT SYSPRINT, MSG1
B EXITØ

ERRMSG2 EQU *
PUT SYSPRINT, MSG2
B EXITØ

ERRMSG3 EQU *
PUT SYSPRINT, MSG3
B EXITØ

ERRMSG4 EQU *
PUT SYSPRINT, MSG4
B EXITØ

MSG1 DC CL8Ø 'Error: Parm2 smaller than parm1'
MSG2 DC CL8Ø 'Error: Parm3 smaller than parm4'
MSG3 DC CL8Ø 'Error: String length smaller than search interval'
MSG4 DC CL8Ø 'Error opening input file'
MSG5 DC CL8Ø 'Record smaller than compare position'
MSGFND DS ØCL8Ø
MSGFND1 DC C' String found in record number ' 
MSGFND2 DC CL50 ' 
MSGTOT DS ØCL8Ø
MSGTOT1 DC C' Number of records where string found: ' 
MSGTOT2 DC CL50 ' 
MSGF1 M DS ØCL8Ø
MSGF1 M1 DC C' Number of records searched. . . . . : ' 
MSGF1 M2 DC CL50 ' 

STRINGL DS H
DS CL2
STRING DS ØCL36
STRING1 DS CL12
STRING2 DS CL12
STRING3 DS CL12
DS CL4
STR1 DS CL12
DC C' ' 
STR2 DS CL12
DC C' ' 

STR3 DS CL12
DC C ' ' 
STRP1 DS CL7 
STRP2 DS CL7 
STRP3 DS CL7  
STRP16 DS CL6 
STRP26 DS CL6  
STRP36 DS CL6  
* 
DS ØF 
XAND DC X'1F1F1F1F1F1F1F1F1F1F1F1F1F1F1F1F' leave bits 00011111 
XTRN DC X'000A0B0C0D0E0F000000000000000000' bits 01111 A thru F 
* 
TRTAB DC X'000102030405060708090A0B0C0D0E0F' Uppercase 
DC X'101112131415161718191A1B1C1D1E1F' translation 
DC X'202122232425262728292A2B2C2D2E2F' table 
DC X'303132333435363738393A3B3C3D3E3F' 
DC X'404142434445464748494A4B4C4D4E4F' 
DC X'505152535455565758595A5B5C5D5E5F' 
DC X'606162636465666768696A6B6C6D6E6F' 
DC X'707172737475767778797A7B7C7D7E7F' 
DC X'808182838485868788898A8B8C8D8E8F' 
DC X'909192939495969798999A9B9C9D9E9F' 
DC X'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF' 
DC X'B0B1B2B3B4B5B6B7B8B9BBABBBCBDBEBEF' 
DC X'C0C1C2C3C4C5C6C7C8C9CACBCCDDCECF' 
DC X'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF' 
DC X'E0E1E2E3E4E5E6E7E8E9EAEBECDEEEFF' 
DC X'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF' 
* 
INFILEA ACB DDNAME=INFILE 
INFILER RPL ACB=INFILEA, 
OPTCD=LOC, 
AREA=VAREA1, 
ARG=CHAVE1 
INFILED DCB DSORG=PS,MACRF=(GL), 
EODAD=EXITØ, 
DDNAME=INFILE 
SYSPRINT DCB DSORG=PS,MACRF=(PM), 
LRECL=8Ø, 
DDNAME=SYSPRINT 
* 
SAVEA DS 18F 
VAREA1 DS F 
CHAVE1 DS F 
LRECL1 DS F 
TOTFOUND DC F' Ø' 
FILETYPE1 DC C' V'
LOCATE REXX EXEC

/* REXX MVS */
/* LOCATE - Locates a string within a file. */
/* Optional argument: file to search. */
/* */
/* This application consists of this EXEC, LOCATE ISPF panel, */
/* and LOCATE Assembler program. The load module should reside */
/* in the library indicated by the loablib variable below. */
/* */
arg file.
file = strip(file,"""")
loadlib = "loadlib.with.locate.module"
tempfile = userid()||".TEMP.FILE"
I = "Y"
do forever
  address ispexec
  'addpop row(1) column(1)'
  'display panel(locate)'
  if rc = 8 then exit
  'rempop'
  address tso
  msg = ""
  hexastring = Ø
  if col1 = "" then col1 = 1
  if col2 = "" then col2 = 9999
  if rec1 = "" then rec1 = 1
  if rec2 = "" then rec2 = 99999999
  if col2 < col1 then do
    msg="Column 2 cannot be smaller than column 1"
iterate
if rec2 < rec1 then do
    msg="Record2 cannot be smaller than record1"
    iterate
end
str = strip(stri)
lstr = length(str)
if left(str,2) = "x'" | left(str,2) = "X'" then do
    hexasring = 1
    str = strip(str,"T","'")
    str = translate(substr(str,3))
lstr = length(str)
if datatype(str, "X") <> 1 then do
    msg="Invalid hexadecimal string"
    iterate
end
lok = lstr // 2
if lok <> Ø then do
    msg="Odd number of characters in hexadecimal string"
    iterate
end
lstr = lstr / 2
if lstr > col2 - col1 + 1 then do
    msg="String is longer than column search zone"
    iterate
end
if msg = "" then leave
end
if hexasring = 1 then I = 'X'
parm = ""right(col1,4,"Ø") || right(col2,4,"Ø") ||,
       right(rec1,8,"Ø") || right(rec2,8,"Ø") ||,
       I || str"'
xx = msg(off)
"free dd (temp1)"
"alloc da ('tempfile') dd(temp1) new reuse blksize(8000),
   lrecl(80) recfm(f,b) dsorg(ps) space(1 1) tracks delete"
if rc <> Ø then do
    say "Error "rc" allocating" tempfile
    exit
end
queue "/"userid()" Ø JOB LOCATE,MSGCLASS=X,CLASS=A"
queue "/"STEPØ EXEC PGM=LOCATE,"
queue "/" PARM="parm"
queue "/"STEPLIB DD DISP=SHR,DSN="loadlib"
queue "/"INFILE DD DISP=SHR,DSN="file"
queue "///SYSPRINT DD SYSOUT=*"
queue ""

"execio * diskw temp1 (finis"
"submit "tempfile"
"free dd (temp1)
say "Job" userid("Ø submitted"
exit

LOCATE ISPF PANEL

)ATTR
  _ TYPE(INPUT) CAPS(ON) JUST(LEFT) COLOR(RED)
  $ TYPE(INPUT) CAPS(OFF) JUST(LEFT) COLOR(RED)
 # TYPE(INPUT) CAPS(ON) JUST(RIGHT) COLOR(RED)
  ? TYPE(TEXT) INTENS(HIGH) SKIP(ON) COLOR(PINK)
  % TYPE(TEXT) INTENS(HIGH) SKIP(ON) COLOR(YELLOW)
 + TYPE(TEXT) INTENS(LOW) SKIP(ON) COLOR(GREEN)
  ! TYPE(OUTPUT) CAPS(OFF) SKIP(ON) COLOR(WHITE)
)BODY WINDOW(70,17)
+
  ? File.......: _FILE                                        +
  +
  ? Search for:$STRI                                 +
  ?         (Enter text string or begin with X' for hexadecimal)
  +
  +      Ignore case - only valid for text (Y,N):_I
  +
  % Search columns       Search records
  +     First column.#COL1+     First record.#REC1    +
  +     last column.#COL2+     Last record.#REC2    +
  +
  ! MSG
  + Enter - execute                              PF3/15 cancel
)INIT
&ZWINTTL = 'Locate a string in a file'
)PROC
&ver='Y,N'
VER(&FILE,nonblank,dsname)
VER(&STRI,nonblank)
VER(&STRI,nonblank)
VER(&I,NONBLANK,listv,&ver)
VER(&COL1,num)
VER(&COL2,num)
VER(&REC1,num)
VER(&REC2,num)
)END

Systems Programmer (Portugal)  © Xephon 2003
Calling the ANTRQST macro

The following program, CQPROG, is an example of how to call the ANTRQST macro. This macro issues calls to the System Data Mover API, and can be used to issue commands for XRC, PPRC, SNAPSHOT, and FLASHCOPY functions.

This particular program issues PPRC CQUERY commands. These can also be issued as TSO CQUERY commands (or alternatively ISMF can obtain the same information), but the program has several advantages over this:

• The TSO command can be quite slow if you are issuing a lot of commands.
• The TSO command mirrors all of its displays on the system console – this can be quite a problem if you are repeatedly issuing a lot of these commands.
• This program uses an easily-tailored input file to select the volumes you want to report on.

The program reads a list containing addresses (required) and volume names (optional) that you want to issue CQUERY commands against, while using PPRC to mirror DASD volumes. TSO variables are updated and displayed using a small piece of REXX. CQPROG will need to be authorized in the IKJTSOx member.

CQUERY PROGRAM

TITLE 'CQPROG - ISSUE "CQUERY" COMMANDS VIA "ANTRQST" MACRO'
***********************************************************************
* CQPROG: ISSUE CQUERY COMMANDS AGAINST SELECTED DEVICES, USING     *
* THE 'ANTRQST' MACRO (SEE 'DFSMSDFP ADVANCED SERVICES').           *
*                                                                     *
* THIS WILL PREVENT THE OUTPUT OF 'CQUERY' COMMANDS FROM               *
* FLOODING THE SYSLOG.                                               *
*                                                                     *
* THE CALLER WILL NEED READ ACCESS TO THE FACILITY CLASS              *
* PROFILE 'STGADMIN.ANT.PPRC.COMMANDS'.                              *
***********************************************************************
* THE 'ANTASØØØ' ADDRESS SPACE MUST BE ACTIVE.
* PARMS: NONE
* INPUT: DATASET CONTAINING ADDRESSES OF DISKS TO BE 'CQUERY'ED.
* OUTPUT: TSO 'PPRQNNN' VARIABLES, FOR PANEL OR REXX DISPLAY.
* RETURN CODES:
* Ø  - SUCCESSFUL COMPLETION
* 4  - AT LEAST ONE NON-ZERO RETURN CODE FROM 'ANTRQST'
* 8  - UNABLE TO OPEN INPUT FILE 'VOLIN'
* 12 - UNABLE TO UPDATE TSO VARIABLE
* PRINT NGEN
* HOUSEKEEPING...

CQPROG CSECT
CQPROG AMODE 31
CQPROG RMODE 24
BAKR R14,Ø
LR R12,R15
USING CQPROG,R12
OPEN (VOLIN,(INPUT))
LTR R15,R15
BNZ BADOPEN
LA R7,VOLINREC
USING VOLINDSC,R7
LA R6,QRYINFO
USING ANTDATA,R6
READLOOP DS ØH
GET VOLIN,VOLINREC
AP PPRTOT,PØ1
MVC VALDEVN, VOLADDR
BAL R9, CONVCUU
ANTRQST ILK=PPRC,
REQUEST=PQUERY,
DEVN=DEVN,
QRYSIZE=QRYSIZE,
QRYINFO=QRYINFO,
RETIINFO=RETIINFO,
ALET=Ø,
ASYNCH=NO,
ECB=NO_ECB,
BITMAP=NOBITMAP,
PATHS=NO,
WAITTIME=Ø

* 
MVC RETCD, RETINFO SAVE RETURNCODE
MVC RSNCD, RETINFO+4 SAVE REASONCODE
CLC RETCD, =A(RQST_PQUERY_QRYSIZE_BIG_ENOUGH) CALL OK?
BE CALLOK

* ***************************************************************************
* CALL TO 'ANTRQST' GAVE NONZERO RETURN CODE. THERE WILL BE AN ERROR *
* MESSAGE IN THE RESPONSE - DISPLAY THAT...                             *
* ***************************************************************************

ANTFAIL DS ØH
MVC RETC, RC4 SET RC=4
MVC VALDEVN+5(5), =C'ANTPØ' SET UP MSG PREFIX
XR R2, R2 CLEAR R2
IC R2, RETINFO+8 GET LENGTH OF FOLLOWING MSG
C R2, =F'56' TOO LONG FOR MSG AREA?
BL MOVEIT NO...MOVE MSG
LA R2, 56 YES...TRUNCATE...

MOVEIT DS ØH
EX R2, EXMVC MOVE MSG TO VARIABLE AREA
B UPDATEIT NOW GO AND UPDATE

* ***************************************************************************
* CALL TO 'ANTRQST' OK - UPDATE TSO VARIABLES...                        *
* ***************************************************************************

CALLOK DS ØH
CLC ANTSTAT(6), =CL6' DUPLEX' #VOLUMES IN DUPLEX STATE
BE UPDTDUPL
CLC ANTSTAT(7), =CL7'SIMPLEX' #VOLUMES IN SIMPLEX STATE
BE UPDTSIMP
CLC ANTSTAT(7), =CL7'SUSPEND' #VOLUMES IN SUSPEND STATE
BE UPDTSUSP
CLC ANTSTAT(7), =CL7'PENDING' #VOLUME IN SUSPEND STATE
BE UPDTPEND
B UPDTTSO UNKNOWN STATUS - SKIP COUNTS

UPDTDUPL DS ØH
MVC VALASTER(3), =C'(*)' HIGHLIGHT DUPLEX PAIRS
AP DUPLECT, PØ1 BUMP DUPLEX COUNT
B UPDTTSO

UPDTSIMP DS ØH
AP SIMPCNT, PØ1 BUMP SIMPLEX COUNT
B UPDTTSO

UPDTPEND DS ØH
AP PENDCT, PØ1 BUMP PENDING COUNT
B UPDTTSO

UPDTSUSP DS ØH
AP SUSPCT, PØ1 BUMP SUSPEND COUNT
UPDTSO DS ØH
MVC VALDEVN, ANTDEVN
MVC VALLEVEL, ANTLEVEL
MVC VALSTAT,ANTSTAT
MVC VALVOLID, VOLVOLID
BAL R9, CHKPATHS CHECK PATHS
UPDATEIT DS ØH
UNPK NAMESTEM(3), PPRTOT
OI NAMESTEM+2, X'F0' SET CORRECT SIGN
BAL R9, UPDTVAR
LTR R15, R15 OK?
BNZ SETRC12 NO... SET RC=12
MVI VALDEVN, C' ' CLEAR OUT DETAIL AREA
MVC VALDEVN+1(VALUELN-1), VALDEVN
B READLOOP
*---------------------------------------------------------------------*
* CLOSE INPUT DATASET...                                         *
*---------------------------------------------------------------------*
CLOSE DS ØH
CLOSE VOLIN
*---------------------------------------------------------------------*
* RETURN TO CALLER WITH RELEVANT RC...                            *
*---------------------------------------------------------------------*
RETURN DS ØH
L R15, RETC LOAD RETURN CODE
PR , RESTORE CALLER DATA, RETURN
*---------------------------------------------------------------------*
* COULDN'T OPEN INPUT FILE 'VOLIN'...                             *
*---------------------------------------------------------------------*
BADOPEN DS ØH
MVC RETC, RC8 SET RC=8
B RETURN
*---------------------------------------------------------------------*
* ERROR UPDATING TSO VARIABLE(S)... RC=12...                      *
*---------------------------------------------------------------------*
SETRC12 DS ØH
MVC RETC, RC12 SET RC=12
B CLOSE
*---------------------------------------------------------------------*
* E-O-F ON INPUT - SET TSO VARIABLES FOR # IN SIMPLEX, # IN DUPLEX, *
* # PENDING, # SUSPENDED AND TOTAL INPUT RECORDS READ...           *
*---------------------------------------------------------------------*
EOFVOLIN DS ØH
MVC NAME(5), =C' PPSIM M'
MVC NAMELEN, =A(5)
UNPK VALDEVN(3), SIMPCT
OI VALDEVN+2, X'FO' SET CORRECT SIGN
MVC VALUELEN, =A(3)
BAL R9, UPDTVAR UPDATE PPSIM
MVC NAME(5), =C'PPDUP'
MVC NAMELEN, =A(5)
UNPK VALDEVN(3), DUPLC
OI VALDEVN+2, X'FØ'
MVC VALUELEN, =A(3)
BAL R9, UPD'TVAR
MVC NAME(5), =C'PPND'
MVC NAMELEN, =A(5)
UNPK VALDEVN(3), PENDCT
OI VALDEVN+2, X'FØ'
MVC VALUELEN, =A(3)
BAL R9, UPD'TVAR
MVC NAME(5), =C'PPSUS'
MVC NAMELEN, =A(5)
UNPK VALDEVN(3), SUSPECT
OI VALDEVN+2, X'FØ'
MVC VALUELEN, =A(3)
BAL R9, UPD'TVAR
MVC NAME(5), =C'PPTOT'
MVC NAMELEN, =A(5)
UNPK VALDEVN(3), PPTOT
OI VALDEVN+2, X'FØ'
MVC VALUELEN, =A(3)
BAL R9, UPD'TVAR
MVC NAME(7), =C'PPRPATH'
MVC NAMELEN, =A(7)
MVC VALDEVN(1), PPRPATH
MVC VALUELEN, =A(1)
BAL R9, UPD'TVAR
B CLOSE

***********************************************************************
*                    + + S U B R O U T I N E + + +                    *
* CONVERT CHARACTER CUU (EG 'Ø94F') INTO ITS BINARY EQUIVALENT, THEN  *
* PLACE IT IN THE 'DEVN' FIELD...                                     *
***********************************************************************
CONVCUU DS ØH
TR VOLADDR(4), TRTAB
XC DWORD, DWORD
PACK DWORD+4(4), VOLADDR(5)
L R8, DWORD+4
SRL R8, 8
STH R8, DEVN
BR R9

***********************************************************************
*                    + + S U B R O U T I N E + + +                    *
* CHECK THE PATHS' FILES IN CASE THERE IS A POTENTIAL PROBLEM. AS WE   *
* ARE USING ONLY 2 PATHS WE'LL CHECK ONLY PATH1 AND PATH2...          *
***********************************************************************
CHKPATHS DS ØH
MVC VALPATHS,=15C ' '          RESET FIELD
CLC ANTSTAT1,=C ' '         STATUS1 = SPACES?
BE PATH1OK                  YES..GOOD
CLC ANTSTAT1,ESTABLISHED    STATUS1 = ESTABLISHED?
BE PATH1OK                  YES..GOOD
MVC VALPATHS,PATHCHEK        NO...SHOW POSSIBLE ERROR
MVI PPRPATH, C'Y'            SHOW POSSIBLE ERROR
BR R9                       RETURN FROM SUBROUTINE

PATH1OK  DS ØH

CLC ANTSTAT2,=C ' '         STATUS2 = SPACES?
BER R9                       YES..RETURN
CLC ANTSTAT2,ESTABLISHED    STATUS2 = ESTABLISHED?
BER R9                       YES..RETURN
MVC VALPATHS,PATHCHEK        NO...SHOW POSSIBLE ERROR
MVI PPRPATH, C'Y'            SHOW POSSIBLE ERROR
BR R9                       RETURN FROM SUBROUTINE

***********************************************************************
*                    + + S U B R O U T I N E + + +                    *
* CALL 'IKJCT441' TO UPDATE TSO VARIABLES...                         *
***********************************************************************
UPDTVAR  DS ØH

* LINK EP=IKJCT441,       PUT VALUE INTO VARIABLE             X
   PARAM=(ECODE,                                           X
   NAMEPTR,                                                X
   NAMELEN,                                                X
   VALUEPTR,                                               X
   VALUELEN,                                               X
   TOKEN),                                                 X
   VL=1

* BR R9                       RETURN

*---------------------------------------------------------------------*
* LTORG                      LITERAL POOL
*---------------------------------------------------------------------*
* ReginaEQUATES, ETC...                                             *
*---------------------------------------------------------------------*
EXMVC  MVC VALDEVN+1Ø(Ø), RETINFO+9 EXECUTED MOVE
SIMPCT DC PL2 '0'
DUPLCT DC PL2 '0'
PENDCT DC PL2 '0'
SUSPCT DC PL2 '0'
PPRTOT DC PL2 '0'
PØ1  DC PL1 '1'
DEVN  DC XL2 '0000'
NO    DC CL3 'NO'
RETCMD DS F
RSNCD  DS  F
FWORD  DS  F
DWORD  DS  D
RETC   DC  F'Ø'
RC4    DC  F'4'
RC8    DC  F'8'
RC12   DC  F'12'
RETI信息系统  XL10Ø
QRY信息系统  XL19ØØ
QRY信息系统 EQU  *·QRY信息系统
QRY信息系统 DC  Y(QRY信息系统)
NOBITMAP DC  CL3'NO'
PATHCHEK DC  CL13'＜CHECK PATHS＞'
PRRPATH DC  CL1'N'
ESTABLISHED DC  CL2'01'
*
  Ô 1 2 3 4 5 6 7 8 9 A B C D E F
TRTAB   DC  X'FFFFFFFFFFFFFFFFFFFFFFF' 0
         DC  X'FFFFFFFFFFFFFFFFFFFFFFF' 1
         DC  X'FFFFFFFFFFFFFFFFFFFFFFF' 2
         DC  X'FFFFFFFFFFFFFFFFFFFFFFF' 3
         DC  X'FFFFFFFFFFFFFFFFFFFFFFF' 4
         DC  X'FFFFFFFFFFFFFFFFFFFFFFF' 5
         DC  X'FFFFFFFFFFFFFFFFFFFFFFF' 6
         DC  X'FFFFFFFFFFFFFFFFFFFFFFF' 7
         DC  X'FFFFFFFFFFFFFFFFFFFFFFF' 8
         DC  X'FFFFFFFFFFFFFFFFFFFFFFF' 9
         DC  X'FFFFFFFFFFFFFFFFFFFFFFF' A
         DC  X'FFFFFFFFFFFFFFFFFFFFFFF' B
         DC  X'FF000C000C000C000C000C' C (ABCDEF)
         DC  X'FFFFFFFFFFFFFFFFFFFFFFF' D
         DC  X'FFFFFFFFFFFFFFFFFFFFFFF' E
         DC  X'00001234567890123456789' F (0123456789)
*
*--------------------------------------------------------------------*
* IKJCT441 PARMLIST (TSO VARIABLE ACCESS ROUTINE)...
*--------------------------------------------------------------------*
NAME     DC  C'PPRQ'                  TSO variable name (fixed)
NAMESTEM DS  CL3                      TSO variable name (variable)
NAMELN   EQU   *·NAME                  variable length
NAMELEN  DC  A(NAMELN)                   " "
VALDEVN  DC  CL4' '                   device number (IE address)
         DC  CL6' '  
VALLEVEL DC  CL9' '                   level
         DC  CL5' '  
VALSTAT  DC  CL7' '                   status
         DC  CL3' '  
VALASTER DC  CL3' '                   highlight if duplex
         DC  CL10' '  

VALVOLI D DC CL6',
   VOL I D
DC CL2',
WHEREPAT EQU '* VALDEVN OFFSET TO VALPATHS FIELD
VALPATHS DC CL13', POSSIBLE PATH PROBLEM
VALUELN EQU '* VALDEVN VALUE LENGTH
VALUELEN DC A(VALUELN)
"","" NAMEPTR DC A(NAME) POINTER TO VARIABLE NAME
VALUEPTR DC A(VALDEVN) POINTER TO VARIABLE VALUE
TOKEN DC F'Ø' TOKEN (UNUSED HERE)
ECODE DC A(TSVEUPDT) ENTRY CODE FOR 'SET'

*---------------------------------------------------------------------*
* INPUT DCB AND BUFFER...                                             *
*---------------------------------------------------------------------*
VOLIN DCB DDNAME=VOLIN, X
   RECFM=FB, X
   DSORG=PS, X
   MACRF=(GM), X
   EODAD=EOFVOLIN
VOLINREC DS CL8Ø

*---------------------------------------------------------------------*
* REGISTER EQUATES, DSECTS, ETC...                                    *
*---------------------------------------------------------------------*
*
* YREGS
*
  ANTRQSTL
*
  IKJTSVT
*
   ANTDATA DSECT
   ANTDEVN DS CL4 DEVICE NUMBER
   DS CL1
   ANLEVEL DS CL9 LEVEL
   DS CL1
   ANTSTAT DS CL10 STATUS
   DS CL1
   ANTPTHST DS CL8 PATH STATUS
   DS CL1
   ANTPSSID DS CL4 PRIMARY SSID
   DS CL1
   ANTPCCA DS CL2 PRIMARY CCA
   DS CL1
   ANTPSER# DS CL12 PRIMARY SERIAL NUMBER
   DS CL1
   ANTSSSID DS CL4 SECONDARY SSID
   DS CL1
   ANTSCCA DS CL2 SECONDARY CCA
   DS CL1
   ANTSSER# DS CL12 SECONDARY SERIAL NUMBER
   DS CL1
The following REXX code will execute the CQUERY program. It expects a sysid (from a valid list), which it then uses as a qualifier for a dataset name to hold the list of addresses that you want to issue the PPRC commands for.

```rexx
/* Rexx tso issue CQUERY-type commands */
lastct = Ø

sys = 'PRD1 PRD2 PRD3 TST1 TST2'
```
lin = "========================"

Parse Upper Arg dsn1 whatsthis
If whatsthis <> "" Then Do
   Say "Extraneous info passed: '"whatsthis'""
   Return 8
End
If Length(dsn1) = 4 Then
   sysid = dsn1
Else
   Parse Value dsn1 WITH pref '.' pprc '.' sysid '.' stuff
If Pos(sysid,sys) = Ø Then Do
   Say "==> Invalid SYSid passed: '"sysid'"
   Return 8
End

Address "TSO"

x = Outtrap("null.",10,"NOCONCAT")
"FREE FI(VOLIN)"
dsn = "'SYSG.PPRC."sysid".INPUT'"
"ALLOC FI(VOLIN) DA("dsn") SHR" /* Alloc list of addresses */

Do Forever
   Address "TSO"
   "CALL 'SYSG.LINKLIB(CQPROG)'"
cqrc = rc
   If cqrc = 8 Then
      pprqØØ1 = "Unable to open input dataset "dsn"..."
   If cqrc = 12 Then Do
      lin = "NB: Unable to update at least one TSO variable..."
      aa = pptot + 3
      aa = Right("ØØ"aa,3)
      Interpret "PPRQ"aa" = lin
   End
   pprdt = Date('E')
   pprtm = Time()
   ppsy = sysid
   If cqrc = Ø Then Do
      ppch = ppdup¦¦"("ppdup-lastct")" /* Calc change since last time */
      ppch = Strip(ppch,"L","Ø") /* Remove leading Øs */
      lastct = ppdup
      aa = pptot + 1
      aa = Right("ØØ"aa,3)
      Interpret "PPRQ"aa" = lin /* Underline last one */
ppsim = Strip(ppsim,"L","Ø")   /* Remove leading Øs */
pppnd = Strip(pppnd,"L","Ø")
ppsus = Strip(ppsus,"L","Ø")
pptot = Strip(pptot,"L","Ø")
End
pathmsg = " "
If pprpath = "Y" Then    /* If prog found path problem, flag this */
    pathmsg = "**** POSSIBLE PATH PROBLEMS - PLEASE CHECK ****"
Address "ISPEXEC"
"ISPEXEC DISPLAY PANEL(PPRCQPAN)"
If rc = 8 Then Return    /* PF3 = exit */
End
Address "TSO"
x = Outtrap("null.",1Ø,"NOCONCAT")
"FREE FI(VOLIN)"
Return    /* That's it... */

PPRCQPAN PANEL

)ATTR DEFAULT(%+_)  
! TYPE(OUTPUT) INTENS(LOW) CAPS(OFF)
¬ TYPE(OUTPUT) INTENS(HIGH)
$ TYPE(OUTPUT) INTENS(HIGH) COLOR(RED) HILITE(BLINK)
# AREA(SCRL) EXTEND(ON)
) BODY WIDTH(8Ø) EXPAND(@@)
¬pprdt %@-@ PPRC QUERY DISPLAY (~ppsy%) @-@¬pprtm % 
+    $pathmsg +
+    SIMPLEX:¬ppsim+DUPLEX:¬ppch
+PENDING:¬pppnd+SUSPEND:¬ppsus+TOTAL:¬pptot
+    =DEVICE=    ==LEVEL==     ===STATUS====         =VOLSER=
#SCRLAREA  
#  
#  
#  
#  
#  
#  
#  
#  
#  
#  
#  
+    +Press%PF3+to%End,%Enter+to Refresh...
) AREA SCRLAREA  
+    ¬PPRQ001 +
INPUT

Format of the input dataset. The addresses start in column 2, the volids in column 20; all other information is ignored.

```
+ -PPRQ002
+ -PPRQ003
+ -PPRQ004
+ -PPRQ005
  ...

and so on up to

+ -PPRQ398
+ -PPRQ399
+ -PPRQ400
)` INIT
` PROC
` END
```

Using Pretty Good Privacy (PGP) to encrypt/decrypt and sign your MVS data

INTRODUCTION TO CRYPTOGRAPHY

Pretty Good Privacy (PGP) is a freeware public key cryptography program developed by Phillip Zimmerman (1991) under the GNU licence, which is now available for z/OS.
Cryptography is the science of using mathematics to encrypt and decrypt data. Cryptography enables you to store sensitive information or transmit it across insecure networks (like the Internet) so that it cannot be read by anyone except the intended recipient.

A cryptographic algorithm, or cipher, is a mathematical function used in the encryption and decryption process. A cryptographic algorithm works in combination with a key (a word, number, or phrase) to encrypt the plaintext. The same plaintext encrypts to different ciphertext with different keys. The security of encrypted data is entirely dependent on two things – the strength of the cryptographic algorithm and the secrecy of the key.

Cryptanalysis is the science of analysing and breaking secure communication.

Encryption

Symmetric-key encryption
In conventional cryptography, also called secret-key or symmetric-key encryption, one key is used both for encryption and decryption.

The Data Encryption Standard (DES) in an example of a conventional cryptosystem that has been widely deployed by the US Government and the banking industry.

An extremely simple example of conventional cryptography is a substitution cipher. A substitution cipher substitutes one piece of information for another. This is most frequently done by offsetting letters of the alphabet.

So, starting with ABCDEFGHIJKLMNOPQRSTUVWXYZ and sliding everything up by three, you get DEFGHIJKLMNOPQRSTUVWXYZABC – where D=A, E=B, F=C, and so on.

Using this scheme, the plaintext SECRET encrypts as VHFUHW. To allow someone else to read the ciphertext, you tell them that the key is 3.
Conventional encryption has benefits — it is quite simple and very fast.

But it has some major limitations — for a sender and recipient to communicate securely using conventional encryption, they must agree on a key and keep it secret between themselves. If they are in different physical locations, they must trust a courier, the phone, or some other secure communications medium to prevent the disclosure of the secret key during transmission. Anyone who overhears or intercepts the key in transit can later read, modify, and forge all information encrypted or authenticated with that key.

The persistent problem with conventional encryption is key distribution — how do you get the key to the recipient without someone intercepting it?

**Asymmetric-key encryption**

The problems of key distribution are solved by public-key cryptography, the concept of which was introduced by Whitfield Diffie and Martin Hellman in 1975.

Public-key cryptography uses a pair of keys — a public key, which encrypts data, and a corresponding private key, for decryption. Because it uses two keys, it is also called asymmetric cryptography. You publish your public key to the world while keeping your private key secret. Anyone with a copy of your public key can then encrypt information that only you can read.

It is computationally infeasible to deduce the private key from the public key.

Anyone who has a public key can encrypt information but cannot decrypt it. Only the person who has the corresponding private key can decrypt the information.

The main benefit of public-key cryptography is that it allows people who have no pre-existing security arrangement to exchange messages securely.
The need for sender and receiver to share secret keys via some secure channel is eliminated; all communications involve only public keys, and no private key is ever transmitted or shared.

Some examples of public-key cryptosystems are:

- Elgamal (named for its inventor, Taher Elgamal).
- RSA (named for its inventors, Ron Rivest, Adi Shamir, and Leonard Adleman).
- Diffie-Hellman (named, you guessed it, for its inventors).
- DSA, the Digital Signature Algorithm (invented by David Kravitz).

*How PGP works*

PGP combines some of the best features of both conventional and public-key cryptography – PGP is a hybrid cryptosystem.

When a user encrypts plaintext with PGP, PGP first compresses the plaintext.

Data compression saves transmission time and disk space and, more importantly, strengthens cryptographic security. Most cryptanalysis techniques exploit patterns found in the plaintext to crack the cipher. Compression reduces these patterns in the plaintext, thereby greatly enhancing resistance to cryptanalysis.

Because the public key encryption algorithm is much slower than conventional single-key encryption, encryption is better accomplished by using a high-quality fast conventional single-key encryption algorithm to encipher the message.

In a process invisible to the user, PGP creates a temporary random key (the session key, which is a one-time-only secret key) just for this ‘session’, which is used to conventionally encipher the plaintext file.

The session key works with a very secure, fast, conventional encryption algorithm to encrypt the plaintext; the result is ciphertext.
Once the data is encrypted, the session key is then encrypted to the recipient’s public key. This public key-encrypted session key is transmitted along with the ciphertext to the recipient.

Decryption works in the reverse. The recipient’s copy of PGP uses his or her private key to recover the session key, which PGP then uses to decrypt the conventionally encrypted ciphertext.

The combination of the two encryption methods combines the convenience of public-key encryption with the speed of conventional encryption.

Conventional encryption is about 10,000 times faster than public-key encryption. Public-key encryption in turn provides a solution to key distribution and data transmission issues. Used together, performance and key distribution are improved without any sacrifice in security.

**Digital signatures**

A major benefit of public key cryptography is that it provides a method for employing digital signatures.

Digital signatures let the recipient of information verify the authenticity of the information’s origin, and also verify that the information was not altered while in transit.

Thus, public key digital signatures provide authentication and data integrity.

The basic manner in which digital signatures are created is:

- The signature algorithm uses your private key to create the signature and the public key to verify it.
- If the information can be decrypted with your public key, then it must have originated from you.

This system has some problems. It is slow, and it produces an enormous volume of data – at least double the size of the original information.

An improvement on this scheme is the addition of a one-way hash function in the process.
A one-way hash function takes variable-length input and produces a fixed-length output. The hash function ensures that, if the information is changed in any way (even by just one bit) an entirely different output value is produced.

**How PGP works**

PGP uses a cryptographically strong hash function on the plaintext the user is signing. This generates a fixed-length data item known as a message digest.

Then PGP uses the digest and the private key to create the ‘signature’. PGP transmits the signature and the plaintext together. PGP can also encrypt the plaintext using the recipient’s public key.

Upon receipt of the message, the recipient uses PGP to recompute the digest, thus verifying the signature.

**PGP INSTALLATION ON Z/OS**

**Downloading PGP for OS/390 from the Web**

PGP for OS/390 and MVS is freeware which can be downloaded from the Alan Nichols Web site at [http://s390.nichols.de/pgp/index.html](http://s390.nichols.de/pgp/index.html).

The current version of PGP for OS/390 (in June 2003) is 2.6.3is. You have to download a PAX file (pgp263is.pax.Z – 1.16 MB) on your PC.

**Allocating and mounting an HFS file on z/OS**

At this point, you should allocate and mount an HFS file on your z/OS system to upload PGP:

```plaintext
//STEP01 EXEC PGM=IEFBR14
//*
//MKFS DD DISP=(,CATLG),DSN=SYS2.OMVS.U.PGP.HFS,
// SPACE=(CYL,(005,01,1)),UNIT=SYSALLDA,
// DSNSTYPE=HFS
```
Uploading the PAX file to z/OS

Now you have to use FTP to transfer the PAX file to your z/OS system:

D:\Migration\Dustbin>ftp 126.9.41.65
Connect to 126.9.41.65.
220 Connection will close if idle for more than 5 minutes.
Utilisateur (126.9.41.65:(none)) : SXSP001
331 Send password please.
Mot de passe :
230 SXSP001 is logged on. Working directory is "SXSP001".
ftp> cd /u/pgp
250 HFS directory /u/pgp is the current working directory
ftp> bin
200 Representation type is Image
ftp> put pgp263is.pax.Z
200 Port request OK.
125 Storing dataset /u/pgp/pgp263is.pax.Z
250 Transfer completed successfully.
ftp : 12176161 octets sent in 21,16 seconds at 57,55 Ko/sec.
ftp> quit
221 Quit command received. Goodbye.

D:\Migration\Dustbin>

Uncompressing the PAX file

You should now use a Telnet session to log on to your USS system and uncompress the PAX file:

EZYTE27I login: SXSP001
EZYTE28I SXSP001 Password:
IBM
Licensed Material - Property of IBM
5694-A01 (C) Copyright IBM Corp. 1993, 2001
Compiling PGP C/C++ source

First, you should check the value of several USS environment variables in your /etc/profile:

```
export _CC_CNAME=CBCDRVR           # The OS/390 C/C++ driver program
export _CC_PLIB_PREFIX=SYS1        # prefix for SYS1.SCEELKEX
```

© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
Then, you should compile the PGP C/C++ sources:

```bash
SXSPØØ1:/u/pgp/pgp/src: >make clean
SXSPØØ1:/u/pgp/pgp/src: >make os390
make all CPP=/lib/cpp OBJS_EXT=".c370.o" CFLAGS=" -O -DC370 -DHIGHFIRST" 
cc -O -DC370 -DHIGHFIRST -DHIGHFIRST pgp.c 
cc -O -DC370 -DHIGHFIRST -DHIGHFIRST crypto.c 
cc -O -DC370 -DHIGHFIRST keymgmt.c 
cc -O -DC370 -DHIGHFIRST fileio.c
WARNING CBC3236 ./c370.h:114 Macro name ENOENT has been redefined.
FSUM3065 The COMPIL E step ended with return code 4.
cc -O -DC370 -DHIGHFIRST mdfile.c 
cc -O -DC370 -DHIGHFIRST more.c 
cc -O -DC370 -DHIGHFIRST armor.c 
cc -O -DC370 -DHIGHFIRST mpilib.c 
cc -O -DC370 -DHIGHFIRST mpilio.c 
cc -O -DC370 -DHIGHFIRST genprime.c 
cc -O -DC370 -DHIGHFIRST rsagen.c 
cc -O -DC370 -DHIGHFIRST random.c 
cc -O -DC370 -DHIGHFIRST idea.c
cc -O -DC370 -DHIGHFIRST passwd.c 
cc -O -DC370 -DHIGHFIRST md5.c 
cc -O -DC370 -DHIGHFIRST system.c 
cc -O -DC370 -DHIGHFIRST language.c 
cc -O -DC370 -DHIGHFIRST getopt.c 
cc -O -DC370 -DHIGHFIRST keyadd.c 
cc -O -DC370 -DHIGHFIRST config.c 
cc -O -DC370 -DHIGHFIRST keymaint.c 
cc -O -DC370 -DHIGHFIRST charset.c 
cc -O -DC370 -DHIGHFIRST randpool.c 
cc -O -DC370 -DHIGHFIRST noise.c 
cc -O -DC370 -DHIGHFIRST zbits.c 
cc -O -DC370 -DHIGHFIRST deflate.c 
cc -O -DC370 -DHIGHFIRST zfile_io.c 
cc -O -DC370 -DHIGHFIRST zglobals.c 
cc -O -DC370 -DHIGHFIRST zinflate.c 
cc -O -DC370 -DHIGHFIRST zip.c 
cc -O -DC370 -DHIGHFIRST zzipup.c 
cc -O -DC370 -DHIGHFIRST ztrees.c 
cc -O -DC370 -DHIGHFIRST unzip.c 
cc -O -DC370 -DHIGHFIRST rsaglue1.c 
cc -O -DC370 -DHIGHFIRST c370.c
cc -o pgp pgp.o crypto.o keymgmt.o fileio.o mdfile.o more.o armor.o mpilib.o mp
    iio.o genprime.o rsagen.o random.o idea.o passwd.o md5.o system.o language.o g
    etopt.o keyadd.o config.o keymaint.o charset.o randpool.o noise.o
```

This step produces the PGP module:

```
SXSPØØ1:/u/pgp/pgp/src: >ls -al pgp
-rwxrwxrwx   1 SXSPØØ1  OMVSGRP   765952 Jun 13 18:06 pgp
SXSPØØ1:/u/pgp/pgp/src: >
```

You can now call PGP:

```
SXSPØØ1:/u/pgp/pgp/src: >pgp -h
No configuration file found.
```

Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-96 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-18
International version - RSA Patent Expired. Does not use RSAREF.
Current time: 2003/06/13 17:09 GMT

Usage summary:
To encrypt a plaintext file with recipient's public key, type:
```
pgp -e textfile her_userid [other userids] (produces textfile.pgp)
```
To sign a plaintext file with your secret key:
```
pgp -s textfile [-u your_userid] (produces textfile.pgp)
```
To sign a plaintext file with your secret key, and then encrypt it
with recipient's public key, producing a .pgp file:
```
pgp -es textfile her_userid [other userids] [-u your_userid]
```
To encrypt with conventional encryption only:
```
pgp -c textfile
```
To decrypt or check a signature for a ciphertext (.pgp) file:
```
pgp ciphertextfile [-o plaintextfile]
```
To produce output in ASCII for email, add the -a option to other
options.
To generate your own unique public/secret key pair: pgp -kg
For help on other key management functions, type: pgp -k

SXSPØØ1:/u/pgp/pgp/src: >

For ease of use, you should have the PGP directory to the PATH
environment variable in /etc/profile:
```
export PATH=$PATH:/u/pgp/pgp/src
```

**If do not have a C/C++ compiler**

If you have no C/C++ compiler, the executable PGP module is
included in the src directory.
So omit the `make clean` and `make os390` commands.

**PGP online documentation**

PGP documentation can be found in the directory `/u/pgp/pgp/doc/`.

**PGP KEYS MANAGEMENT**

Now that PGP is installed on your z/OS system, in order to use it, you need to:

- Create your own keypair – PGP requires a keypair: a private key and a public key.
- Exchange your public keys with other PGP users.

**Create your private/public keypair**

You have to create your own private/public keypair, which will be stored in two ‘keyrings’.

A keyring is a file that stores keys.

Each user has two keyrings – one public and one private:

- The private keyring (secring.pgp) stores your private key.
- The public keyring (pubring.pgp) stores public keys; yours, and those that you receive from other PGP users.

To generate your own unique public/secret key pair, you should use the following command from the USS prompt:

```
SXSP001:/u/SXSP001: >pgp +nomanual -kg
No configuration file found.
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses. 
(c) 1990-96 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-18
International version - RSA Patent Expired. Does not use RSAREF.
Current time: 2003/06/16 10:17 GMT
```

Pick your RSA key size:

1) 512 bits - Low commercial grade, fast but less secure
2) 768 bits - High commercial grade, medium speed, good security
3) 1024 bits - "Military" grade, slow, highest security
Choose 1, 2, or 3, or enter desired number of bits: 2048

Generating an RSA key with a 2048-bit modulus.

You need a user ID for your public key. The desired form for this user ID is your name, followed by your E-mail address enclosed in <angle brackets>, if you have an E-mail address. For example: John Q Smith 12345.6789@compuserve.com

Enter a user ID for your public key:
MyOrg Site1
MyOrg Site1

You need a pass phrase to protect your RSA secret key. Your pass phrase can be any sentence or phrase and may have many words, spaces, punctuation, or any other printable characters.

Enter pass phrase: pass phrase for MyOrg Site1

Enter same pass phrase again: pass phrase for MyOrg Site1

Note that key generation is a lengthy process.

We need to generate 1536 random bits. This is done by measuring the time intervals between your keystrokes. Please enter some random text on your keyboard until you hear the beep:

1536
1528 ff
1504 fffgeez
1448 ehptptep
1376 fptoto
1320 ftppepep
1248 rprrprp
1192 totptp
1136 tptptp
1080 tptptpp
1016 tpppp
984 evstcy
928 tptptp
872 toprp
824 rtpvm
776 tpbpbp
720 ptptp
672 fppfpp
616 rprprp
568 dptptp
512 vpvpvp
456 rprprp
400 yyoy
The ‘pass phrase’ is a kind of password to protect your secret key. Each time you need to use your secret key (to decrypt a file, to sign a message…), you will have to specify your pass phrase.

This command produces the public and the private keyrings:

SXSPØØ1:/u/SXSPØØ1: >ls -al *.pgp
-rw-rw-rw- 1 SXSPØØ1 OMVSGRP  963 Jun 16 11:22 pubring.pgp
-rw-rw-rw- 1 SXSPØØ1 OMVSGRP 1489 Jun 16 11:22 secring.pgp
SXSPØØ1:/u/SXSPØØ1: >

Exchange your public keys with other PGP users

After you have created a keypair, you can begin corresponding with other PGP users.

But first, you will need to:

• Extract and send a copy of your public key to other PGP users.
• Add other PGP users’ public keys to your public keyring.

Extracting your public key

To extract your public key from your public ring, you should enter:

SXSPØØ1:/u/SXSPØØ1: >pgp -kx MyOrg Site1
No configuration file found.
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-96 Philip Zimmermann, Phil's Pretty Good Software. 1996.01.18
International version - RSA Patent Expired. Does not use RSAREF.
Adding a public key to your public keyring

When you receive a public key from another PGP user, you have to use the following command to add it to your public keyring:

```
SXSP001:/u/SXSP001: >pgp -ka Site1.pgp
No configuration file found.
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-96 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-18
International version - RSA Patent Expired. Does not use RSAREF.
Current time: 2003/06/16 09:00 GMT

Looking for new keys...
pub  2048/E59DA965 2003/06/16  MyOrg Site1

Checking signatures...
pub  2048/E59DA965 2003/06/16  MyOrg Site1
sig!      E59DA965 2003/06/16  MyOrg Site1

Keyfile contains:
  1 new key(s)

One or more of the new keys are not fully certified.
Do you want to certify any of these keys yourself (y/N)? y

Key for user ID: MyOrg Site1
2048-bit key, key ID E59DA965, created 2003/06/16
Key fingerprint = 70 CF C1 A4 8E F6 28 01 D8 65 4A E7 90 B5 90 EC
This key/userID association is not certified.
  Questionable certification from:
  MyOrg Site1

Do you want to certify this key yourself (y/N)? y
Looking for key for user 'MyOrg Site1':

Key for user ID: MyOrg Site1
2048-bit key, key ID E59DA965, created 2003/06/16
  Key fingerprint = 70 CF C1 A4 8E F6 28 01 D8 65 4A E7 90 B5 90 EC

READ CAREFULLY: Based on your own direct first-hand knowledge, are you absolutely certain that you are prepared to solemnly certify that the above public key actually belongs to the user specified by the above user ID (y/N)? y

You need a pass phrase to unlock your RSA secret key.
Key for user ID: MyOrg Site2
512-bit key, key ID 3EFAE341, created 2003/06/11

Enter pass phrase: MyOrg Site2
Pass phrase is good. Just a moment....
Key signature certificate added.

Make a determination in your own mind whether this key actually belongs to the person whom you think it belongs to, based on available evidence. If you think it does, then based on your estimate of that person's integrity and competence in key management, answer the following question:

Would you trust "MyOrg Site1" to act as an introducer and certify other people's public keys to you? (1=I don't know. 2=No. 3=Usually. 4=Yes, always.) ? 4

PGP USAGE
You can use PGP to:
• Exchange an encrypted file from Site2 to Site1.
• Sign a message to send it from Site2 to Site1.

In order to use PGP, the users must have their PGP ring files in their home directory.

On Site2, we will use the following sample message file to send it to Site1:
Exchanging an encrypted file

Encrypting a file on Site2
To encrypt a plaintext file with the recipient’s Site1 public key, you should enter:

```
SXSP001:/u/SXSP001: >pgp -e text_msg.txt MyOrg Site1
```

Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses. (c) 1990-96 Philip Zimmermann, Phil’s Pretty Good Software. 1996-01-18 International version - RSA Patent Expired. Does not use RSAREF.

Current time: 2003/06/16 09:21 GMT

Recipients’ public key(s) will be used to encrypt.
Key for user ID: MyOrg Site1
2048-bit key, key ID E59DA965, created 2003/06/16

Key for user ID: MyOrg Site1
2048-bit key, key ID E59DA965, created 2003/06/16

Ciphertext file: text_msg.txt.pgp
SXSP001:/u/SXSP001: >

This command produces the encrypted file, which is not readable!
Decrypting a file on Site1

To decrypt the message, the recipient on Site1 should use the following command:

```
SXSPØØ1:/u/SXSPØØ1: >pgp -d text_msg.txt.pgp
```

File is encrypted. Secret key is required to read it.
Key for user ID: MyOrg Site1
2048-bit key, key ID E59DA965, created 2003/06/16

You need a pass phrase to unlock your RSA secret key.
Enter pass phrase: pass phrase for MyOrg Site1
Pass phrase is good. Just a moment......

The result is the 'clear' file:

```
BROWSE -- /u/SXSPØØ1/text_msg.txt ------------ Line 00000000 Col 001 025
```

Using PGP in batch

You can also use PGP as a batch job using BPXBATCH:
This first example shows how to specify the pass phrase using the 
\texttt{--z} parameter.

The next example will use the \texttt{PGPPASS} environment variable:

\begin{verbatim}
//INET   EXEC PGM=BPXBATCH, REGION=4096K,
// PARM='SH pgp -d '/u/SXSP001/text_msg.txt.pgp'
//SYSERR DD PATH='/tmp/pgpbatch.syserr',
// PTHOPTS=(OWRONLY, OCREAT, OTRUNC),
// PATHMODE=SIRWXU
//STDOUT DD PATH='/tmp/pgpbatch.stdout',
// PTHOPTS=(OWRONLY, OCREAT, OTRUNC),
// PATHMODE=SIRWXU
//STDERR DD PATH='/tmp/pgpbatch.stderr',
// PTHOPTS=(OWRONLY, OCREAT, OTRUNC),
// PATHMODE=SIRWXU
//SYSOUT DD PATH='/tmp/pgpbatch.sysout',
// PTHOPTS=(OWRONLY, OCREAT, OTRUNC),
// PATHMODE=SIRWXU
//STDENV DD *
PGPPASS=pass phrase for MyOrg Site1
/*

Signing a message

\textbf{Signing a ‘clear’ message on Site2}

To sign a plaintext file with your secret key and have the output 
readable to people:

\texttt{SXSP001:/u/SXSP001: >pgp -sta text_msg.txt -u MyOrg Site2}

\texttt{No configuration file found.}
Pretty Good Privacy™ 2.6.3i - Public-key encryption for the masses.  
(c) 1990-96 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-18 
International version - RSA Patent Expired. Does not use RSAREF.  
Current time: 2003/06/20 15:19 GMT

A secret key is required to make a signature.  
You need a pass phrase to unlock your RSA secret key.  
Key for user ID: MyOrg Site2  
2048-bit key, key ID E59DA965, created 2003/06/16

Enter pass phrase: pass phrase for MyOrg Site2 
Pass phrase is good. Just a moment....  
Clear signature file: text_msg.txt.asc  
SXSP001:/u/SXSP001: >

The result of this command is the following ‘signed message’:

BROWSE ... /u/SXSP001/text_msg.txt.asc ............ Line 00000000 Col 001 064
Command ===> Scroll ===> HALF
********************************************************** Top of Data *****************************************
-----BEGIN PGP SIGNED MESSAGE-----
line 01 from MyOrg Site2
line 02 from MyOrg Site2
line 03 from MyOrg Site2
line 04 from MyOrg Site2
line 05 from MyOrg Site2
line 06 from MyOrg Site2
line 07 from MyOrg Site2
line 08 from MyOrg Site2
line 09 from MyOrg Site2
line 10 from MyOrg Site2
line 11 from MyOrg Site2

-----BEGIN PGP SIGNATURE-----
Version: 2.6.3i
Charset: noconv
iQEVAwUBpvMIj+cDqrlnalAIQEd5gfjXKS2jilu3zOGFVJpvnbc7M0llfhs7XD
b1f3VurG9OB1/z3xHa7aRQAvePZzbdclyrhB0tQ3oLRZtanPDV1y9o6iEXbazhI
xla/5XxGjzqSM9sDfjhhr7TYiWC2hpyAt eoYciro9HFV0088BqHLP/QZUVbZe5FN
EG+b10kYaXaN01CVk2lOexQTtE27DahqRxrQ7qZ0r1EmvTj05I43UPXnjHO9dR2
h51c41EHLS4h7CPyYQvKpw0lOq9VF9taUHjaCvKJlrxUzuZLD18z6N00gQrjmLBVQ
3BD95ZPOMKjkw5eCEwvaZxVYXC0NkZpTGJa3VHuvFn2Rb2nSj8Ug==
=24Fu
-----END PGP SIGNATURE-----
********************************************************** Bottom of Data *****************************************

Verifying the signature of the ‘clear’ message from Site2 on Site1

To check the signature integrity of a signed file:

SXSPØØ1:/u/SXSPØØ1: >pgp text_msg.txt.pgp
No configuration file found.
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-96 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-18
International version - RSA Patent Expired. Does not use RSAREF.
Current time: 2003/06/20 12:43 GMT

File has signature. Public key is required to check signature.

Good signature from user "MyOrg Site2".
Signature made 2003/06/20 14:35 GMT using 2048-bit key, key ID E59DA965

Plaintext filename: text_msg.txt
SXSPØØ1:/u/SXSPØØ1: >

Signing an ‘encrypted’ message on Site2

To sign a plaintext file with your secret key, and then encrypt it with the recipient’s public key:

SXSPØØ1:/u/SXSPØØ1: >pgp -es text_msg.txt MyOrg Site1 -u MyOrg Site2
No configuration file found.
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-96 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-18
International version - RSA Patent Expired. Does not use RSAREF.
Current time: 2003/06/20 16:01 GMT

A secret key is required to make a signature.
You need a pass phrase to unlock your RSA secret key.
Key for user ID: MyOrg Site2
2048-bit key, key ID E59DA965, created 2003/06/16

Enter pass phrase: pass phrase for MyOrg Site2
Pass phrase is good. Just a moment....

Recipients' public key(s) will be used to encrypt.
Key for user ID: MyOrg Site1
2048-bit key, key ID 5A0B2D55, created 2003/06/20

Key for user ID: MyOrg Site1
2048-bit key, key ID 5A0B2D55, created 2003/06/20

Key for user ID: MyOrg Site2
2048-bit key, key ID E59DA965, created 2003/06/16

.
Verifying the signature of the ‘encrypted’ message from Site2 on Site1

To decrypt an encrypted file and to check the signature integrity of a signed file:

SXSPØØ1:/u/SXSPØØ1: >pgp text_msg.txt.pgp
No configuration file found.
Pretty Good Privacy(tm) 2.6.3i - Public key encryption for the masses.
(c) 1990-96 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-18
International version - RSA Patent Expired. Does not use RSAREF.
Current time: 2003/06/20 14:03 GMT

File is encrypted. Secret key is required to read it.
Key for user ID: MyOrg Site1
2048-bit key, key ID 5A0B2D55, created 2003/06/20

You need a pass phrase to unlock your RSA secret key.
Enter pass phrase: pass phrase for MyOrg Site1
Pass phrase is good. Just a moment......
File has signature. Public key is required to check signature.

Good signature from user "MyOrg Site2".
Signature made 2003/06/20 16:01 GMT using 2048-bit key, key ID E59DA965

Plaintext filename: text_msg.txt
SXSPØØ1:/u/SXSPØØ1: >

PGP LINKS

Systems Programmer (France) © Xephon 2003
Using TAR and JAR files on MVS

I recently needed to transmit a large number of TIF images (held in MVS datasets) to a server running a Windows environment. The Unix TAR command can be used under Unix System Services on MVS to create an archive file that can then be transmitted to the server and expanded using the familiar Windows ZIP utility.

In the following example the images are first copied from a PDS to an HFS directory using the OPUTX EXEC.

```bash
// /*
// /* use the OPUTX exec to copy each member of the
// /* PDS into the HFS directory
// /*
// /* OPUTX EXEC PGM=IKJEFT01
// */ SYSEXEC DD DSN=SYS1.SBPXEXEC, DISP=SHR
// */ SYSPROC DD DISP=SHR, DSN=SYS1.SISPCLI B
// */ ISPPLIB DD DISP=SHR, DSN=SYS1.ISISPENU
// */ DD DISP=SHR, DSN=SYS1.ISSFPLIB
// */ ISPMLIB DD DISP=SHR, DSN=SYS1.ISSPMENU
// */ DD DISP=SHR, DSN=SYS1.ISSFMLIB
// */ DD DISP=SHR, DSN=SYS1.SBPXMENU
// */ ISPSLIB DD DISP=SHR, DSN=SYS1.ISPSLI B
// */ DD DISP=SHR, DSN=SYS1.ISPSLI B
// */ ISPTLIB DD DISP=SHR, DSN=SYS1.ISPTENU
// */ DD DISP=SHR, DSN=SYS1.ISFTLIB
// */ ISPPROF DD DSN=&&ISPPROF, SPACE=(TRK,(5,1,2)), LIKE=&userid.ISPPROF,
// */ DD DISP=(NEW, PASS)
// */ SYSTSPRT DD SYSOUT=* 
// */ SYSTSN DD *
// OPUTX 'your-PDS-of-images' '/your-dir' ASIS BINARY CONVERT(NO) + 
// MODE(644) SUFFIX(tif)
/*

The TAR command is then used to create the archive dataset — note that in this particular case the compression flag is not used because TIF images are already in a compressed format.

```bash
// /*
// /* use Unix System Services TAR command
// /* to create an archive dataset
// /*
// /* TAR EXEC PGM=BPXBATCH,
```
The resulting MVS dataset, &userid.TIFFS.TAR, can be sent (obviously as a binary transmit) to the server and unzipped to restore the individual image files. Note that the Windows ZIP program will recognize files with an extension of .tar.

If you want to include text files in your archive, there is a slight problem in that we are using a binary transfer—somewhere along the line the data will need to be translated into ASCII. The following job will take an MVS text file (ie EBCDIC character data) and convert it to ASCII. The TEXT option in this case will result in the addition of an LF character being added to delimit the end of each record. This will be OK for a Unix system; however, the standard Windows text file expects CR LF or X'0D0A' as the delimiter.

```
// OCOPY  EXEC PGM=IKJEFT01
// INDD     DD DSN=your-text-file,DISP=OLD
// OUTHFS   DD PATH='/your-dir/myfile.txt'
//          PATHDISP=(KEEP,DELETE),
//          PATHOPTS=(OWRONLY,OCREAT),PATHMODE=SIRWXU
// SYSTSPRT DD SYSOUT=*  
// SYSTSN   DD *
// OCOPY INDD(INDD) OUTDD(OUTHFS) TEXT CONVERT((BPXFX311)) + FROM1Ø47
```

While the Windows ZIP utility is quite happy to work with Unix TAR files created on MVS, the reverse is not true. To cater for the situation where you may want to package up some files on a Windows environment and upload them to MVS as a single dataset, you can make use of the Java archive utility, which provides a similar ZIP functionality, and, being Java, provides platform-independence. Further information about Java archive files can be found on the Internet (see the Sun Java tutorial pages).
Assuming that you have made Java available on your MVS system (mounting the HFS containing the Java software and setting the environment variables), the following JCL illustrates the use of the JAR command to expand a Java archive file that was created on a Windows PC:

```bash
/* use JAR to unload the archive */

JAR EXEC PGM=BPXBATCH,
    PARM='sh cd work;jar xf /work/fixes.jar'
STDOUT DD PATH='/tmp/stdout',
    PATHOPTS=(OCREAT,OWRONLY),PATHMODE=SIRWXU,
    PATHDISP=KEEP
STDERT DD PATH='/tmp/stderr',
    PATHOPTS=(OCREAT,OWRONLY),PATHMODE=SIRWXU,
    PATHDISP=KEEP

The OGETX EXEC can then be used to move files into a standard MVS PDS:

OGETX /work '&userid.MVS.PDS' LC SUFFIX(bin) BINARY

With many vendors providing fixes over the Internet as downloads, one possible use for this kind of job would be to move a number of fixes from a directory on your PC up to an MVS dataset.

Dave Welch (New Zealand) © Xephon 2003

Finding CSECTs within LPA load modules in virtual storage

In the course of development of certain types of system application, the ability to programmatically locate CSECT addresses within LPA-resident modules can sometimes prove useful. Unfortunately, the information to facilitate such location, ie the displacement of a given CSECT within the load module, while present in the linkage editor control information for the module, is not retained in virtual storage when the module is...
actually loaded. Thus, it becomes necessary to refer back to the linkage editor CESD (Composite External Symbol Dictionary) control records, which form a portion of the load module's contents in its LPALST library on DASD. From these, the desired CSECT can be located, and its load module displacement extracted. This value, when added to the output of a system service call (which identifies the module's virtual storage load address), precisely identifies the virtual storage address of the aforesaid CSECT.

FNDCSCT is a statically or dynamically called routine which performs the above function. It is passed the load module and CSECT names as parameters, and, upon completion of a successful location operation, returns the associated virtual storage address in register 0 and a return code of 0 in register 15. WTO-type error messages and a non-zero return code in register 15 are generated upon recognition of any conditions which preclude successful completion. Note that the location technique employed will result in the appearance of one or more IEC141I 013-18 exception messages if the desired module is not found in the first of two or more LPALST libraries. These do not signal error conditions if the module is found in a subsequent LPALST library. If the module is not found in any LPALST library, then IEC141I 013-18 messages will appear for all such libraries, and a further ‘FNDCSCT007E Load module loadmodulename was not found’ message will signal the error condition.

FNDCSCT should be linkedited as a stand-alone load module into any desired load library. No specific linkage editor attributes need be assigned.

The calling sequences for FNDCSCT are as follows:

**Static call:**

```
call  FNDCSCT,(loadmod,csect)
ltr   r15,r15
bnz   errorrtn
loadmod dc  cl8'loadmodulename'
csect dc  cl8'Csectname'
```
Dynamic call:

```assembly
link    ep=FNDCSCT,(loadmod,csect)
ltr    r15,r15
bnz    errortrn
loadmod dc c18'loadmodulename'
csect   dc c18'CSectname'

FNDCSCT
fn dcsct   amode 31
fn dcsct   rmode 24
fn dcsct   csect
r0       equ  0
r1       equ  1
r2       equ  2
r3       equ  3
r4       equ  4
r5       equ  5
r6       equ  6
r7       equ  7
r8       equ  8
r9       equ  9
r10      equ 10
r11      equ 11
r12      equ 12
r13      equ 13
r14      equ 14
r15      equ 15
stm   r14,r12,12(r13) entry linkage
lr    r12,r15
using fn dcsct,r12
st    r13,savearea+4
la    r15,savearea
st    r15,8(r13)
lr    r13,r15
b    fn c0020 branch to start
return ds Øh
xr    r15,r15 exit linkage
l    r13,4(r13)
l    r14,12(r13)
lm    r1,r12,24(r13)
br    r14 return
fn c0020 ds Øh
lr    r9,r1 save pal pointer
l    r3,cvtptr CVT pointer
icm   r3,15,cvtsmext-cvt(r3) CVT extension
bnz    fn c0050 it's there
```

© 2003. Xerphon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
wto 'FNDCSCT001E No CVTX address found'
l a r15,8
b return+2

fsc0050 ds 0h
icm r3,15, cvtep|ps* cvtstg xr3) LPAT address
bnz fsc0100 it's there
wto 'FNDCSCT002E No LPAT address found'
l a r15,8
b return+2

fsc0100 ds 0h
cle =cl 4' LPAT', 0(r3) really the
LPAT?
00490000
be fsc0150 YES
wto 'FNDCSCT003E LPAT ID check failed'
l a r15,8
b return+2

fsc0150 ds 0h
icm r4,15,4(r3) number of LPAT entries
bnz fsc0200 more than none
wto 'FNDCSCT004E No LPAT entries present'
l a r15,8
b return+2

fsc0200 ds 0h
l r2,0(,r9) get member name address
mve membrtu+6(8),0(r2) move member name to txt unit
l a r3,9(r3) point to first LPAT entry

fsc0250 ds 0h
cle 0(r3),0 end of entries?
be fsc0900 yes, module not found
mve dsnamtu+6(44),0(r3) no, move dsname to text unit
mvi alcverb,s99vrbal prime dynalloc fields
l a r0, dsnamtu
st r0, alctua1
l a r0, membrtu
st r0, alctua2
l a r0, statstu
st r0, alctua3
l a r0, rtddntu
st r0, alctua4
oi alctua4, x'80'
l a r1, alcrbptr

dynalloc
l tr r15,r15 allocation ok?
bz fsc0400 yes
st r15, allocrc
mve wto00250+37(44), dsnamtu+6 move dataset name
trt wto00250+37(44), trtable search for terminating blank
mvi 0(r1),c'(' member name preparation
lr r5,r1 retain address of blank
mvc 1(8,r5), memr tu+6  move member name
trt 1(8,r5), trtable  search for terminating blank
mvi Ø(r1), c')'  end of member name
cnop 0, 4
wto 00250 wto 'FNDCSCT005E Error allocating'

mvc dfdaplp, =a(alcrb)  initialize DAIRFAIL parms
mvc dfrcp, =a(allocrc)
l a r1, =a(Ø)
st r1, dfjeff02
la r1, =x'4032'
sto r1, dfidp
xc dfpcllp, dfcpcllp
mvi dfbufp, =a(dfbufs)
l a r1, dfpars
link ep=IKJEFF18  invoke DAIRFAIL
ltr r2, r15  ok?
bz fcsc0300  yes
wto 'FNDCSCT006E DAIRFAIL error - return code set to DAIRFAIL'
l return code'
l r r15, r2
b return+2

fcsc0300 ds Øh
lh r5, dfbuf11  extract the DAIRFAIL message
sh r5, =h'5'
ch r5, =h'112'
bnh fcsc0320
lh r5, =h'112'

fcsc0320 ds Øh
ex r5, exmvc1  move it to the wto
cnop 0, 4
wto 00300
wto 'FNDCSCT006E

clic dfbuf12, =a(Ø)  any second level message?
be fcsc0340  no
lh r5, dfbuf12  yes, extract as well
sh r5, =h'5'
ch r5, =h'112'
bnh fcsc0330
lh r5, =h'112'

fcsc0330 ds Øh
ex r5, exmvc2  move it to wto
cnop 0, 4
wto 00330
wto 'FNDCSCT006E
fcsc0340 ds 0h
l r15, allocrc
b return+2

fcsc0400 ds 0h
mvc library+dcbddnam-ihadcb(8), rtdntu+6 move ddname
open (library, (INPUT)), mode=31 open the library
tm library+(dcbflags-ihadcb), dcbflags ok?
bo fcsc1000 yes, module has been located
la r3, 45(r3), (r3), (r3) no, next LPAT library
b fcsc0250 recycle

fcsc0900 ds 0h
l r3, 0(r9) get load module name addr
mvc wto00900+32(8), 0(r3) move load module name

cnop 0, 4
wto00900 wto 'FNDCSCT007E Load module xxxxxxxx was not found'
la r15, 8
b return+2

fcsc1000 ds 0h
get library get a load module record
lr r3, r1 retain its address
cli 0(r3), x'20' CESD record?
bo fcsc1000 no, get another one
la r4, 0
icm r4, 3, 6(r3) get record length
la r3, 8(r3)
la r4, 0(r3), r4) point past last byte
sh r4, =h'8' back off sufficiently
l r5, 4, (r9) get passed CSECT name addr

fcsc1150 ds 0h
clr r3, r4 past upper search limit?
bln fcsc1000 yes, go get another record
clc 0(8, r3), 0(r5) no, CSECT names match?
bne fcsc1500 no

fcsc1500 ds 0h
la r3, 1, (r3) next byte
b fcsc1150 reiterate

fcsc2000 ds 0h
la r4, 0
icm r4, b'0111', 9(r3) load the displacement
l r8, 0, (r9) point to load module name
csvquery inepname=(r8), search=LP, outloadpt=loadpt look for it
iltr r2, r15 find it?
blz fcsc2050 yes
ch r2, =h'8' module not found?
bz fcsc2020 yes
wto 'FNDCSCT008E CSVQUERY error - return code set to CSVQUIER*
Y return code'
lr    r15,r2
b     return+2
fcs2020 ds 0h
wto   'FNDCSCT009E CSVQUERY could not find the requested load*
       module'
la    r15,8
b     return+2
fcs2050 ds 0h
l     r0,loadpt
alr   r0,r4
       add CSECT
displacement 01960013
b     return

***************
* DCB ABEND exit *
***************
fcs8000 ds 0h
lr    r3,r1
lr    r4,r14
mvi   3(r3),4
lr    r14,R4
br    r14

***************
* DCB EODAD routine *
***************
fcs9000 ds 0h
l     r2,4(,r9)
mvc   wto09000+51(8),0(r2)
cnop  0,4
wto09000 wto   'FNDCSCT010E No CESD record found for CSECT xxxxxxxx  '
la    r15,8
b     return+2

***************
* Executed instructions *
***************
exmvc1 mvc   wto00300+20(0),dfbuft1
exmvc2 mvc   wto00330+20(0),dfbuft2

***************
* Data Area *
***************
savearea dc 18f'0'
dsnmtu dc al 2(daldsnam),al 2(1),al 2(44),cl 44' '
membrtu dc al 2(dalmembr),al 2(1),al 2(08),cl 08' '
statstu dc al 2(dalstats),al 2(1),al 2(01),xl 01'8'
rtddntu dc al 2(dalrtddn),al 2(1),al 2(08),cl 08' '
library dcb ddname=dummy,macrf=GL,dso=PS,recfm=U,lrecl=0,
           blksize=32760,devd=DA,eodad=fcs9000,exlst=exlst
exlst dc 0f'0',x'11',al 3(fcs8000)
loadpt dc a(0)
allocrc dc f'0'

module load point
allocation return code
Reorganization of datasets

KSDS datasets with a lot of activity tend to get split. In practice, CI splits are more frequent than CA splits, and when a CA split happens the number of CI splits is normally already large.
Datasets that tend to get split should be periodically reorganized, in order to avoid performance degradation.

The program presented here reads a LISTCAT output, searching for KSDS files whose number of CI splits is greater than a specified limit. When it finds one, it writes out a five-step JCL to reorganize that file.

The steps are as follows:

- Create VSAM temp file (similar to the original)
- Repro original to the temp
- Delete and define the original file
- Repro from temp to the new file
- Delete the temp.

The volume where the new file is allocated is the same as the one where the original file was. I made no provision for multi-volume files because I felt files with such dimensions should be treated separately. This program should be used only for single volume datasets.

All the JCL is written to a single file that can later be submitted, or modified if necessary before being submitted. The program also writes to SYSTSPRT a list of the files considered for reorganization, showing how many there are.

This EXEC should be run in a two-step batch job, where the first step issues a LISTCAT ALL for a given catalog, writing the output to a file, and the second step runs the EXEC, reading the output and writing out the JCL to another file.

**SAMPLE JOB**

```plaintext
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD DISP=(NEW,PASS),DSN=&&TEMP1,UNIT=VIO,
//       DCB=(LRECL=121,RECFM=FB,BLKSIZE=2420),
//       SPACE=(TRK,(30,30))
//SYSTSPRT DD sysout=*,
//SYSTSPRT DD sysout=*,
//SYSIN DD *
```
LISTCAT ALL CATALOG(my.catalog)

/*
//STEP2    EXEC PGM=IRXJCL,PARM='VSREORG'
//SYSEXEC  DD DISP=SHR,DSN=where.is.the.exec
//SYSTSPRT DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
//FILEIN   DD DISP=(OLD,DELETE),DSN=&&TEMP1 
//FILEOUT  DD DISP=SHR,DSN=jcl.file

VSREORG REXX

/*= REXX ==============================================================*/
/* VSREORG - Reads a "LISTCAT ALL" listing allocated to FILEIN and */
/* generates JCL for the reorganization of KSDS datasets */
/* whose CI split number is greater than a certain */
/* threshold. The JCL is written to DDname FILEOUT. */
/* Threshold value is indicated by variable splitsci. */
/*================================================================*/

splitsci = 10             /* minimum number of CI splits for reorg */
suf = "TP"                 /* suffix for temporary datasets */
execio 1 diskr filein
if rc <>Ø then do
  say "Error reading filein"
  exit
end
pull linha
cc = left(linha,1)
data_flag = Ø
c = Ø
do alpha = Ø
  execio 1 diskr filein
  if rc <>Ø then leave
  pull linha
  if cc = 1 then linha = substr(linha,2)
call check_line_type
  if data_flag = 1 then do
    call extract_values
  end
  call write_fileout
end
say total "datasets were considered for reorganization"
exit
/*================================================================*/
/* check_line_type: */
select
when substr(linha,1,7)="CLUSTER" then do
    c = c + 1
    cluster.c = word(linha,3)
    data_flag = Ø
    aix = Ø
end
when substr(linha,4,4)="DATA" & aix = Ø then do
    data.c = word(linha,3)
    data_flag = 1
end
when substr(linha,1,3)="AIX" then do
    data_flag = Ø
    aix = 1
end
when substr(linha,4,5)="INDEX" & data_flag = 1 & aix = Ø then do
    index.c = word(linha,3)
    data_flag = Ø
end
when substr(linha,1,7)="NONVSAM" then do
    data_flag = Ø
end
otherwise nop
end
return
/*==================================================================*/
/*                   extract values from line                       */
/*==================================================================*/
extract_values:
select
when substr(linha,8,7)="REC-TOT" & data_flag = 1 then do
    linha = translate(linha," ", "-");
    rectot.c = right(word(linha,3),11)
    splici.c = right(word(linha,6),5)
    if splici.c < splitsci then do
        c = c - 1
        data_flag = Ø
    end
end
when substr(linha,8,8)="SHROPTNS" & data_flag = 1 then do
    shropt.c = substr(linha,17,3)
    if word(linha,5) <> "INDEXED" then do
        c = c - 1
        data_flag = Ø
    end
end
when substr(linha,8,7)="REC-INS" then do
    linha = translate(linha," ", "-");
    freeci.c = right(word(linha,6),2)
end
when substr(linha, 8, 7)="REC-UPD" then do
  linha = translate(linha, " ", ", -")
  freeca.c = right(word(linha, 6), 2)
end

when substr(linha, 8, 7)="REC-DEL" then do
  linha = translate(linha, " ", ", -")
  splica.c = right(word(linha, 6), 3)
end

when substr(linha, 8, 6)="KEYLEN" then do
  linha = translate(linha, " ", ", -")
  keylen.c = right(word(linha, 2), 2)
  alrecl.c = right(word(linha, 4), 5)
  alrecl.c = space(alrecl.c, Ø)
  cisize.c = right(word(linha, 8), 5)
  cisize.c = space(cisize.c, Ø)
end

when substr(linha, 8, 3)="RKP" then do
  linha = translate(linha, " ", ", -")
  keypos.c = right(word(linha, 2), 2)
  mlrecl.c = right(word(linha, 4), 5)
  mlrecl.c = space(mlrecl.c, Ø)
end

when substr(linha, 8, 7)="SPACE-T" then do
  linha = translate(linha, " ", ", -")
  sptype.c = left(word(linha, 3), 3)
  if sptype.c = "CYL" then sptype.c = "CYLINDERS"
  if sptype.c = "TRA" then sptype.c = "TRACKS"
  if sptype.c = "REC" then sptype.c = "RECORDS"
end

when substr(linha, 8, 7)="SPACE-P" then do
  linha = translate(linha, " ", ", -")
  spprim.c = right(word(linha, 3), 5)
  spprim.c = space(spprim.c, Ø)
end

when substr(linha, 8, 7)="SPACE-S" then do
  linha = translate(linha, " ", ", -")
  spseco.c = right(word(linha, 3), 4)
  spseco.c = space(spseco.c, Ø)
end

when substr(linha, 8, 6)="VOLSER" then do
  linha = translate(linha, " ", ", -")
  volume.c = word(linha, 2)
end
otherwise nop
end
return

/*==================================================================*/
/*                       Write fileout                              */
/*==================================================================*/
write_fileout:
  total = Ø
  do k = 1 to c
    dropbuf
    if length(cluster.k."suf") > 42 then do
      say ">>> Temporary name exceeds 42 characters"
      say ">>> for dataset " cluster.k."suf"
      say ">>> Dataset skipped"
    iterate k
  end
  queue "/STEP"k"1 EXEC PGM=IDCAMS,COND=(Ø,LT)"
  queue "/SYSPRINT DD SYSOUT=*"
  queue "/SYSIN DD *
  queue " DEFINE CLUSTER( -"
  queue "   NAME("cluster.k"."suf") -"
  queue "   "sptype.k"("spprim.k spseco.k") -"
  queue "   RECSZ("alrecl.k mlrecl.k") -"
  queue "   KEYS("keylen.k keypos.k") -"
  queue "   DATA( -"
  queue "   "NAME("cluster.k"."suf".D) -"
  queue "   INDEX( -"
  queue "   "NAME("cluster.k"."suf".I))"
  queue "/*
  queue "/STEP"k"2 EXEC PGM=IDCAMS,COND=(Ø,LT)"
  queue "/INF DD DISP=SHR,"n
  queue "/DSN="cluster.k"
  queue "/OUTF DD DISP=SHR,"n
  queue "/DSN="cluster.k"."suf"
  queue "/SYSPRINT DD SYSOUT=*"
  queue "/SYSIN DD *
  queue " REPRO INFILE(INF) OUTFILE(OUTF)"
  queue "/*
  queue "/STEP"k"3 EXEC PGM=IDCAMS,COND=(Ø,LT)"
  queue "/SYSPRINT DD SYSOUT=*"
  queue "/SYSIN DD *
  queue " DELETE ("cluster.k") CL PURGE"
  queue " DEFINE CLUSTER( -"
  queue "   NAME("cluster.k") -"
  queue "   "sptype.k"("spprim.k spseco.k") -"
  queue "   VOLUME("volume.k") -"
  queue "   RECSZ("alrecl.k mlrecl.k") -"
  queue "   FSPC("freeci.k freeca.k") -"
  queue "   KEYS("keylen.k keypos.k") -"
  queue "   DATA( -"
  queue "   NAME("data.k") -"
  queue "   CISZ("cisize.k") -"
  queue "   INDEX( -"
  queue "   NAME("index.k")"
  queue "/*"
Why not share your expertise and earn money at the same time? *MVS Update* is looking for macros, program code, JCL, REXX EXECs, etc, that experienced MVS users have written to make their life, or the lives of their users, easier.

We will publish it (after vetting by our expert panel) and send you a cheque when the article is published. Articles can be of any length and can be sent to Trevor Eddolls at any of the addresses shown on page 2. Alternatively, they can be e-mailed to trevore@xephon.com.

A free copy of our *Notes for contributors* is available from our Web site at www.xephon.com/nfc.
## October 1999 – September 2003 index

Items below are references to articles that have appeared in *MVS Update* since October 1999. References show the issue number followed by the page number(s). Subscribers can download copies of all issues in Acrobat PDF format from Xephon’s Web site.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>64-bit real storage</td>
<td>198.13-26</td>
</tr>
<tr>
<td>AES algorithm</td>
<td>176.3-9</td>
</tr>
<tr>
<td>Alias</td>
<td>184.3-8</td>
</tr>
<tr>
<td>Allocating cartridges</td>
<td>174.15-23</td>
</tr>
<tr>
<td>ANTRQST</td>
<td>204.22-33</td>
</tr>
<tr>
<td>APF</td>
<td>193.3-5</td>
</tr>
<tr>
<td>Archiving</td>
<td>187.8-14</td>
</tr>
<tr>
<td>ASCII</td>
<td>192.64-69</td>
</tr>
<tr>
<td>ASIDs</td>
<td>195.8-17</td>
</tr>
<tr>
<td>Assembler</td>
<td>190.17-31, 193.50-57</td>
</tr>
<tr>
<td>ASVT</td>
<td>182.49-68</td>
</tr>
<tr>
<td>Automation</td>
<td>182.26-31</td>
</tr>
<tr>
<td>Auxiliary Storage Monitor</td>
<td>182.13-24</td>
</tr>
<tr>
<td>Back-up</td>
<td>193.27-49, 194.30-60, 194.64-71, 195.31-60</td>
</tr>
<tr>
<td>Bar code</td>
<td>201.56-71</td>
</tr>
<tr>
<td>Batch</td>
<td>183.15-19</td>
</tr>
<tr>
<td>Bimodal Migration Accommodation</td>
<td>201.3-5</td>
</tr>
<tr>
<td>BIND</td>
<td>197.23-32</td>
</tr>
<tr>
<td>BPXMTEXT</td>
<td>202.70-71</td>
</tr>
<tr>
<td>Browse</td>
<td>188.39-71, 189.56-69</td>
</tr>
<tr>
<td>C functions</td>
<td>192.22-32</td>
</tr>
<tr>
<td>C</td>
<td>183.29-32, 190.17-31, 193.50-57</td>
</tr>
<tr>
<td>CA 1 TMC</td>
<td>157.3-5</td>
</tr>
<tr>
<td>CA 1</td>
<td>168.9-52</td>
</tr>
<tr>
<td>Cache status management</td>
<td>160.11-17</td>
</tr>
<tr>
<td>Calculating length</td>
<td>196.8-13</td>
</tr>
<tr>
<td>Cartridge drives</td>
<td>193.5-14, 195.19-30</td>
</tr>
<tr>
<td>Change</td>
<td>184.14-23</td>
</tr>
<tr>
<td>Channel information</td>
<td>165.12-27</td>
</tr>
<tr>
<td>Checkpoint</td>
<td>185.16-23</td>
</tr>
<tr>
<td>CI size</td>
<td>200.9-15</td>
</tr>
<tr>
<td>Cleaning volumes</td>
<td>172.27-42</td>
</tr>
<tr>
<td>COBOL</td>
<td>186.62, 194.10-26, 204.56-63</td>
</tr>
<tr>
<td>COBOL coding efficiency</td>
<td>161.3-15</td>
</tr>
<tr>
<td>COBOL II</td>
<td>157.5-7</td>
</tr>
<tr>
<td>COBOL Unit Tester</td>
<td>181.69-70</td>
</tr>
<tr>
<td>COBOL variables</td>
<td>161.64-65</td>
</tr>
<tr>
<td>COBOL Version 2 Release 2</td>
<td>171.8-13</td>
</tr>
<tr>
<td>Comparing files</td>
<td>203.25-34</td>
</tr>
<tr>
<td>Comparison</td>
<td>190.31-37</td>
</tr>
<tr>
<td>Compression</td>
<td>184.24-51</td>
</tr>
<tr>
<td>Concurrent copy</td>
<td>166.3-8</td>
</tr>
<tr>
<td>COPY</td>
<td>187.3-8</td>
</tr>
<tr>
<td>CPP</td>
<td>194.3-7</td>
</tr>
<tr>
<td>CPSM/SPOC</td>
<td>202.27-40</td>
</tr>
<tr>
<td>Cross memory mapping</td>
<td>159.49-57</td>
</tr>
<tr>
<td>CSECTs</td>
<td>200.64-71, 204.56-63</td>
</tr>
<tr>
<td>CSI</td>
<td>203.69-71</td>
</tr>
<tr>
<td>Cursor-sensitive ISPF</td>
<td>158.56-71, 160.50-71, 161.65-71</td>
</tr>
<tr>
<td>DASD</td>
<td>145.40-43, 146.3-9, 151.3-11, 170.44-63, 180.48-64, 187.38-71, 189.37-40</td>
</tr>
<tr>
<td>DASD information</td>
<td>189.6-20</td>
</tr>
<tr>
<td>DASD initialization</td>
<td>182.32-47, 183.33-48</td>
</tr>
<tr>
<td>DASD space monitor</td>
<td>177.3-12</td>
</tr>
<tr>
<td>DASD tuning</td>
<td>160.17-27</td>
</tr>
<tr>
<td>DASD volume display</td>
<td>170.12-17</td>
</tr>
<tr>
<td>Data warehousing</td>
<td>204.7-9</td>
</tr>
<tr>
<td>Dataset creation date</td>
<td>175.55-58</td>
</tr>
<tr>
<td>Dataset information</td>
<td>185.9-16, 189.41-55</td>
</tr>
<tr>
<td>Dataset names</td>
<td>202.50-55</td>
</tr>
<tr>
<td>Dataset ownership</td>
<td>194.7-10</td>
</tr>
<tr>
<td>Date and time</td>
<td>202.8-9</td>
</tr>
<tr>
<td>Debug</td>
<td>194.10-26</td>
</tr>
<tr>
<td>Defrag</td>
<td>180.3-9, 196.62-71, 197.32-47</td>
</tr>
<tr>
<td>Delete</td>
<td>200.7-9</td>
</tr>
<tr>
<td>DES algorithm</td>
<td>162.12-31</td>
</tr>
<tr>
<td>DFHSM</td>
<td>191.33-57, 193.27-49, 194.64-71, 204.3-6, 202.19-27</td>
</tr>
<tr>
<td>Dialog Manager</td>
<td>189.21-37</td>
</tr>
<tr>
<td>Topic</td>
<td>Page Numbers</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>PF keys</td>
<td>162.70</td>
</tr>
<tr>
<td>PF Keys</td>
<td>199.21-23</td>
</tr>
<tr>
<td>PGP</td>
<td>204.33-53</td>
</tr>
<tr>
<td>PL/I</td>
<td>184.70-71</td>
</tr>
<tr>
<td>POST macro</td>
<td>177.54-70</td>
</tr>
<tr>
<td>Prefixes</td>
<td>203.8-9</td>
</tr>
<tr>
<td>PROCLIB</td>
<td>189.69-71</td>
</tr>
<tr>
<td>PROFILE</td>
<td>163.64-71</td>
</tr>
<tr>
<td>PROGxx</td>
<td>179.3-5</td>
</tr>
<tr>
<td>PUTFINES</td>
<td>161.27-31</td>
</tr>
<tr>
<td>Queries</td>
<td>201.9-12</td>
</tr>
<tr>
<td>Query</td>
<td>199.17-20</td>
</tr>
<tr>
<td>Reconstructing source code</td>
<td>177.19-54</td>
</tr>
<tr>
<td>Record tailoring</td>
<td>175.28-55</td>
</tr>
<tr>
<td>Recovery</td>
<td>194.30-60, 195.31-60</td>
</tr>
<tr>
<td>Redbooks</td>
<td>181.67-69</td>
</tr>
<tr>
<td>Re-entrant programming</td>
<td>172.63-68</td>
</tr>
<tr>
<td>Register contents</td>
<td>194.27-29</td>
</tr>
<tr>
<td>REORG</td>
<td>203.35-45</td>
</tr>
<tr>
<td>REPLACE</td>
<td>187.3-8</td>
</tr>
<tr>
<td>Return code special register</td>
<td>166.8-10</td>
</tr>
<tr>
<td>REXX</td>
<td>163.3-6, 167.49-52, 168.7-9, 173.3-9, 188.3-4</td>
</tr>
<tr>
<td>REXX over IP</td>
<td>168.52-65, 169.52-65</td>
</tr>
<tr>
<td>RMF Spreadsheet Reporter</td>
<td>168.3-7</td>
</tr>
<tr>
<td>Rounding errors</td>
<td>203.10-15</td>
</tr>
<tr>
<td>SCRT</td>
<td>202.41-49</td>
</tr>
<tr>
<td>Search</td>
<td>188.18-27, 188.28-39</td>
</tr>
<tr>
<td>Search engine</td>
<td>190.7-17</td>
</tr>
<tr>
<td>Searching with COBOL</td>
<td>165.42-45</td>
</tr>
<tr>
<td>SELCOPY</td>
<td>169.3-4</td>
</tr>
<tr>
<td>SELCOPY and BASE 64</td>
<td>176.41-53</td>
</tr>
<tr>
<td>SMP/E</td>
<td>160.15-27, 182.25, 193.15-26</td>
</tr>
<tr>
<td>SMP/E SMPPTS</td>
<td>181.3-18</td>
</tr>
<tr>
<td>SMP/E SYMSMODS</td>
<td>173.9-15</td>
</tr>
<tr>
<td>SMS</td>
<td>162.32-48, 185.23-53, 186.25-61, 187.15-24, 189.6-20, 201.47-56</td>
</tr>
<tr>
<td>SnapShot copy</td>
<td>183.15-19, 184.71</td>
</tr>
<tr>
<td>Software changes</td>
<td>196.3-7</td>
</tr>
<tr>
<td>Sorting hexadecimal data</td>
<td>157.3-8</td>
</tr>
<tr>
<td>Sorting stem variables</td>
<td>163.51-71</td>
</tr>
<tr>
<td>SORT SYMNAMES</td>
<td>199.9-16</td>
</tr>
<tr>
<td>Spool offload facility</td>
<td>175.6-28</td>
</tr>
<tr>
<td>Spreadsheet</td>
<td>190.37-71</td>
</tr>
<tr>
<td>STC</td>
<td>186.10-22</td>
</tr>
<tr>
<td>Storage tables</td>
<td>198.29-44</td>
</tr>
<tr>
<td>Strings</td>
<td>160.47-49,</td>
</tr>
</tbody>
</table>
Information Builders has announced availability of WebFOCUS for Mainframe Linux BI tool, allowing secure data access and integration over Linux, across 35 other platforms, including MVS and OS/390, VM/CMS, Unisys, Unix, and NT, and more than 85 applications.

Additionally, mainframe Linux customers can also exploit WebFOCUS’s data access capabilities and securely access any data set under MVS or OS390, including DB2, IMS, and VSAM.

Features include reporting, query, and analysis facilities with support for standard reports to ad hoc queries and in-depth OLAP analysis.

Data visualization, such as GIS mapping and advanced graphical components, allows users at all levels to perform graphical analysis to help discover relationships and spot trends.

Information delivery and management features let users schedule, distribute, share, and archive corporate information, while data access, integration, and ETL allow companies to access virtually any type of information source or build data marts and data warehouses to exploit their information.

For further information contact:
Information Builders, Two Penn Plaza, New York, NY 10121-2898, USA.
Tel: (212) 736 4433.

* * *

IIBM has announced WebSphere Business Integration Adapters V2.3.0 and V2.3.1, which introduce 11 new adapters and provide connectivity to WebSphere Application Server.

The WebSphere Business Integration Adapter portfolio has been expanded to support industry applications. This release contains new adapters to provide connectivity to IndusConnect Framework, Maximo MEA, and QAD MFG/PRO.

New technology adapters in this release provide support for healthcare industry standards, such as Health Level Seven (HL7), as well as providing connectivity to iSeries, Lotus Domino, Exchange, COM, and CORBA.

Two new mainframe adapters are also being introduced. The first provides connectivity to a Natural application by executing the Natural program and retrieving the data returned from the execution. The adapter uses JDBC connectivity to the OS/390 platform. Standard operations on the applications are executed using stored procedure calls. The second is the adapter for IDMS, which provides access to IDMS data on mainframe systems using JDBC connectivity and host RPC. Standard operations on the applications are executed using stored procedure calls.

Both mainframe adapters use the NEON Shadow client and server to achieve connectivity. These adapters are available on AIX, HP-UX, Solaris, and Windows.

For further information contact your local IBM representative.

* * *