In this issue

3  Implementing the new AES encryption algorithm
18  Using SVC screening to rename or delete a dataset without SYSDSN ENQ
39  Utilities for FTP
41  SELCOPY and BASE64
53  Formatting internal system trace table entries – part 1
72  MVS news

© Xephon plc 2001
Implementing the new AES encryption algorithm

The US Department of Commerce recently picked a Belgian algorithm called Rijndael (pronounced ‘rain doll’), to be its Advanced Encryption Standard (AES). The AES was developed by two Belgian (more precisely Flemish) researchers: Dr Joan Daemen, and Dr Vincent Rijmen. It is intended to be issued as a FIPS standard and will replace DES. DES was approved by the US Commerce Department as a standard in 1977, but it no longer provides the level of security needed by many of today’s applications. Obviously, Triple DES does provide much stronger security, but not in an efficient way. AES will be used by the US government to protect sensitive information. If we looked at what happened with DES, there is no doubt that many organizations and companies throughout the world will also adopt it.

The AES supports key sizes of 128 bits, 192 bits and 256 bits (while the DES has a key size of only 56 bits). It is a symmetric encryption algorithm that encrypts blocks with a length of 128, 192 or 256 bits; 128 bits is the most common (and the value recommended in the FIPS standard), it is the one that I hard-coded in the REXX EXEC provided in this article.

The algorithm is fast, simple, secure, versatile, and it has low memory requirements. It may be implemented very efficiently on a wide range of processors and in hardware. It has undergone close scrutiny and was chosen after three years of thorough examination.

I used the FIPS draft published by NIST in February 2001 (‘Announcing the Advanced Encryption Standard’) to write a REXX EXEC that may be used to encipher or decipher datasets. This FIPS documentation is much more readable than the original Rijndael specification; programmers with a minimal mathematical background should find it easy to understand.

The AES is a cipher with a simple and elegant structure, although it is unconventional in that its blocks of data are considered as arrays of bytes, and operations like addition and multiplication are ‘redefined’: addition is the XOR logical operation, that is the bitxor function in REXX, and multiplication is our ‘mult’ function.
Unlike the DES, which operated on bits, the AES operates on bytes, which makes it easy to program even in high-level languages. Although the exec here should work fine, it is given mainly for educational purposes. You may find it slow. Some ways to make it faster would include:

- Rewrite it from scratch in Assembler.
- Compile the EXEC with the REXX compiler (I noticed it made it six times faster).
- Do not use my ‘mult’ function (because it makes decryption slower), but replace it with four different functions, mult09, mult0b, mult0d, mult0e, that would be called in our Mix4, Mix5, Mix6, Mix7 functions.
- Recode the decryption process. The decryption may be accomplished according to two different methods: the ‘inverse cipher’ (the one I used here), and the ‘equivalent inverse cipher’, which should be more efficient.

The mode of operation that is used here is CBC (Cipher Block Chaining), the more secure since identical blocks of data result in different blocks after encryption.

Each record is processed in CBC mode, with an init vector that is hard-coded. For the AES, blocks must be 128-bit (or 16-byte); for the last partial block we use CFB (cipher feedback) in order not to change the size of the data.

JCL TO USE THE AES REXX EXEC

```hll
// SET KEY='0123456789AABBCCDDEEFF' encryption key in hex
// SET INFILE='MY.DATASET' file to be encrypted
// SET LIB='MY.LIB.CLIST' clist library
///<
//TEST EXEC PGM=IKJEFT01,PARM=AES ** NO PARAMETER : RUN EXAMPLES **
//SYSPROC DD DISP=SHR,DSN=&LIB
//SYSTSPRT DD SYSOUT=* 
//SYSTSIN DD DUMMY
///<
//ENCRYPT EXEC PGM=IKJEFT01,PARM='%AES &KEY ENCRYPT'
//SYSPROC DD DISP=SHR,DSN=&LIB
//SYSTSPRT DD SYSOUT=* 
```
THE AES REXX EXEC

/* REXX */
parse arg key option /* parameters : */
/* 1) encryption key in hex format */
/* 2) processing option (encrypt, decrypt) */
/* Check option */
option = translate(option)
if option <> 'ENCRIPT' & option <> 'DECRIPT' then do
  if option = '' then signal example
  say 'Option' option 'in error, must be encrypt or decrypt'
  exit(4)
end

trace = 'n'
ddin = 'INFILE' : ddout = 'OUTFILE'
init_vector = 'ØØ'x

/* Process initialization */
call init(128) /* we use AES-128 */
key = x2c(key) /* key must be in hex format */
say 'Key is :' c2x(key)
call Key_Expansion(key) /* Key expansion */

/* Main loop to read a record, process it, and rewrite it */
N = Ø /* record count */
B = Ø /* byte count */
DO FOREVER
  'EXECIO 1 DISKR ' ddin /* reading input file */
IF RC > Ø THEN LEAVE
N = N + 1                               /* increment record count */
parse pull record                      /* get record from stack */
if option = 'ENCRYPT' then say record
B = B + length(record)                 /* increment byte count */
if option = 'ENCRYPT' then ,
  record = Zone_Cipher_CBC(record)     /* encipher record */
else ,
  record = Zone_Decipher_CBC(record)   /* decipher record */
if option = 'DECRYPT' then say record
push record                             /* put record to stack */
'EXECIO 1 DISKW ' ddout                /* writing output file */
END
say N 'records copied,' B 'bytes processed.'
'EXECIO Ø DISKR' ddin '(FINIS)'         /* close file */
'EXECIO Ø DISKW' ddout '(FINIS)'        /* close file */
exit

/*------------------------------------------------------------------*/
/* Example : 128-bit key (from Appendix D of AES FIPS publication)*/
/*------------------------------------------------------------------*/
Example:
trace = 'N'
call init(128)
input    = '3243f6a8885a308d313198a2e0370734'x  /* encryption input */
expected = '392581D02DC09FBDC118597196A0B32'x  /* encrypted input */
key      = '2b7e15162aed2a6abf7158809cf4c3'x
call Key_Expansion(key)                 /* key expansion */
say
say 'Testing AES-'||32*Nk 'with key=' c2x(key)
say '                input=' c2x(input)
output   = AES_cipher(input)            /* encrypting the input */
if output = expected then say 'encryption OK, output=' c2x(output)
else say 'error, output=' c2x(output)
output = AES_Inv_cipher(output)         /* Decrypting */
if output = input then say 'decryption OK, output=' c2x(output)
else say 'error, output=' c2x(output)

/*------------------------------------------------------------------*/
/* Example : 192-bit key (from Appendix D of AES FIPS publication)*/
/*------------------------------------------------------------------*/
call init(192)
input    = '00123456789aabbccddeeff'x  /* encryption input */
expected = 'dda97ca4864cdef06aef70a0ec0d7191'x  /* encrypted input */
key      = '001203405060708090a0b0c0d0e0f101121314151617'x
call Key_Expansion(key)                 /* key expansion */
say
say 'Testing AES-'||32*Nk 'with key=' c2x(key)
say '                input=' c2x(input)
output   = AES_cipher(input)            /* encrypting the input */
if output = expected then say 'encryption OK, output=' c2x(output)
else say 'error, output=' c2x(output)
output = AES_Inv_cipher(output) /* decrypting */
if output = input then say 'decryption OK, output=' c2x(output)
else say 'error, output=' c2x(output)
/*-----------------------------------*/
/* Example : 256-bit key (from Appendix D of AES FIPS publication) */
/*-----------------------------------*/
call init(256)
input = '00112233445566778899aabbcddeeff'x /* encryption input */
expected = '8ea2b7ca516745bfeafc49904b496009'x /* encrypted input */
key = '000102030405060708090a0b0c0d0e0f1011121314151617'x || '18191a1b1c1d1e1f'x
call Key_Expansion(key) /* key expansion */
say
say 'Testing AES-'''32*Nk 'with key=' c2x(key)
say 'input=' c2x(input)
output = AES_cipher(input) /* kncrypting the input */
if output = expected then say 'encryption OK, output=' c2x(output)
else say 'error, output=' c2x(output)
output = AES_Inv_cipher(output) /* Decrypting */
if output = input then say 'decryption OK, output=' c2x(output)
else say 'error, output=' c2x(output)
exit
/*-----------------------------------*/
/* Encrypting a variable-length zone of data in CBC mode */
/*-----------------------------------*/
Zone_Cipher_CBC: procedure expose init_vector ,
                   Nk Nb Nr Rcon. w. trace
parse arg zone
chain = left(init_vector,16,'00'x) /* initialize CBC chaining */
output_zone = '' /* initialize output zone */
/* Main loop to process a 16-byte block - CBC encryption */
do i = 1 to length(zone)%16
   block = substr(zone,1+16*(i-1),16) /* take a block in the zone */
   block = AES_cipher( bitxor(block,chain) ) /* CBC enciphering */
   chain = block /* reinit chaining value */
   output_zone = output_zone || block /* concat resulting block */
   end
/* Process last block with length < 16, if any */
/* The last block is enciphered using a CFB encryption mode. */
/* in order to let the length of the output zone inchanged */
lastblock_length = length(zone) - 16*(length(zone)%16)
if lastblock_length = ø then return output_zone
lastblock = substr(zone,length(zone)-lastblock_length+1)
/* isolate last block of data */
block = bitxor(AES_cipher(chain), lastblock, '00'x) /* CFB mode*/
Zone_Decipher_CBC: procedure expose init_vector, Nk, Nb, Nr, Rcon, w, trace

parse arg zone

chain = left(init_vector,16,'ØØ'x) /* initialize CBC chaining */
output_zone = '' /* initialize output zone */
/* Main loop to process a 16-byte block - CBC decryption */
do i = 1 to length(zone)%16
   block = substr(zone,1+16*(i-1),16) /* take a block in the zone */
   block = bitxor(AES_Inv_cipher(block),chain) /* CBC deciphering*/
   chain = substr(zone,1+16*(i-1),16) /* reinit chaining value */
   output_zone = output_zone || block /* concat resulting block */
end
/* Process last block with length < 16, if any */
/* The last block is deciphered using a CFB encryption mode. */
/* in order to let the length of the output zone inchanged */
lastblock_length = length(zone) - 16*(length(zone)%16)
if lastblock_length = Ø then return output_zone
lastblock = substr(zone,length(zone)-lastblock_length+1) /* isolate last block of data */
block = bitxor(AES_cipher(chain), lastblock, 'ØØ'x) /* CFB mode*/
return output_zone || left(block, lastblock_length)

AddRoundKey: procedure expose w.

parse arg state, round /* argument must be char, 16 bytes */
if length(state) <> 16 then exit(55)
j = round*4 ; word = w.j
j = j+1 ; word = word || w.j
j = j+1 ; word = word || w.j
j = j+1 ; word = word || w.j
return bitxor(state,word)

MixColumns: procedure
parse arg state /* argument must be char, 16 bytes */
if length(state) <> 16 then exit(8)
col.Ø = substr(state,1,4) ; col.1 = substr(state,5,4)
col.2 = substr(state,9,4) ; col.3 = substr(state,13,4)
col.Ø = Mixcol(col.Ø) ; col.1 = Mixcol(col.1)
col.2 = Mixcol(col.2) ; col.3 = Mixcol(col.3)
return col.Ø||col.1||col.2||col.3

/*------------------------------------------------------------------*/
/* InvMixColumns() is the inverse of the MixColumns() transformation*/
/*------------------------------------------------------------------*/
InvMixColumns: procedure

parse arg state     /* argument must be char, 16 bytes */
if length(state) <> 16 then exit(8)
col.Ø = substr(state,1,4) ; col.1 = substr(state,5,4)
col.2 = substr(state,9,4) ; col.3 = substr(state,13,4)
col.Ø = InvMixcol(col.Ø) ; col.1 = InvMixcol(col.1)
col.2 = InvMixcol(col.2) ; col.3 = InvMixcol(col.3)
return col.Ø||col.1||col.2||col.3

/*------------------------------------------------------------------*/
/* Mixing a column according to the MixColumns function (encryption)*/
 /*------------------------------------------------------------------*/
Mixcol: procedure

parse arg col
if length(col) <> 4 then exit(19)
return MixØ(col) || Mix1(col) || Mix2(col) || Mix3(col)

 /*------------------------------------------------------------------*/
/* Mixing a column according to InvMixColumns function (decryption)*/
 /*------------------------------------------------------------------*/
InvMixcol: procedure

parse arg col
if length(col) <> 4 then exit(19)
return Mix4(col) || Mix5(col) || Mix6(col) || Mix7(col)

 /*------------------------------------------------------------------*/
/* Mix a column ; used by MixColumns for encryption */
 /*------------------------------------------------------------------*/
MixØ: procedure

parse arg word
if length(word) <> 4 then exit(20)
s.Ø = substr(word,1,1) ; s.1 = substr(word,2,1)
s.2 = substr(word,3,1) ; s.3 = substr(word,4,1)
r1 = d2c( xtime(c2d(s.Ø)) )     /* multiply by 'Ø2'x */
r2 = bitxor(s.1,d2c( xtime( c2d(s.1) ) ) ) /* multiply by 'Ø3'x */
return bitxor(bitxor(r1,r2),s.2),s.3)
Mix1: procedure
parse arg word
if length(word) <> 4 then exit(21)
s.Ø = substr(word,1,1) ; s.1 = substr(word,2,1)
s.2 = substr(word,3,1) ; s.3 = substr(word,4,1)
r1 = d2c( xtime(c2d(s.1)) )      /* multiply by 'Ø2'x */
r2 = bitxor(s.2,d2c( xtime( c2d(s.2) ) ) ) /* multiply by 'Ø3'x */
return bitxor(bitxor(bitxor(r1,r2),s.Ø),s.3)

Mix2: procedure
parse arg word
if length(word) <> 4 then exit(22)
s.Ø = substr(word,1,1) ; s.1 = substr(word,2,1)
s.2 = substr(word,3,1) ; s.3 = substr(word,4,1)
r1 = d2c( xtime(c2d(s.2)) )      /* multiply by 'Ø2'x */
r2 = bitxor(s.3,d2c( xtime( c2d(s.3) ) ) ) /* multiply by 'Ø3'x */
return bitxor(bitxor(bitxor(r1,r2),s.Ø),s.1)

Mix3: procedure
parse arg word
if length(word) <> 4 then exit(23)
s.Ø = substr(word,1,1) ; s.1 = substr(word,2,1)
s.2 = substr(word,3,1) ; s.3 = substr(word,4,1)
r1 = d2c( xtime(c2d(s.3)) )      /* multiply by 'Ø2'x */
r2 = bitxor(s.Ø,d2c( xtime( c2d(s.Ø) ) ) ) /* multiply by 'Ø3'x */
return bitxor(bitxor(bitxor(r1,r2),s.1),s.2)

Mix4: procedure
parse arg word
if length(word) <> 4 then exit(21)
s.Ø = substr(word,1,1) ; s.1 = substr(word,2,1)
s.2 = substr(word,3,1) ; s.3 = substr(word,4,1)
Mix5: procedure
parse arg word
if length(word) <> 4 then exit(22)
s.Ø = substr(word,1,1) ; s.1 = substr(word,2,1)
s.2 = substr(word,3,1) ; s.3 = substr(word,4,1)
r1 = mult(c2d(s.Ø),Ø9) /* multiply by 'Ø9'x */
r2 = mult(c2d(s.1),14) /* multiply by 'Øe'x */
r3 = mult(c2d(s.2),11) /* multiply by 'Øb'x */
r4 = mult(c2d(s.3),13) /* multiply by 'Ød'x */
return bitxor(bitxor(bitxor(r1,r2),r3),r4)

Mix6: procedure
parse arg word
if length(word) <> 4 then exit(23)
s.Ø = substr(word,1,1) ; s.1 = substr(word,2,1)
s.2 = substr(word,3,1) ; s.3 = substr(word,4,1)
r1 = mult(c2d(s.Ø),13) /* multiply by 'Ød'x */
r2 = mult(c2d(s.1),Ø9) /* multiply by 'Ø9'x */
r3 = mult(c2d(s.2),14) /* multiply by 'Øe'x */
r4 = mult(c2d(s.3),11) /* multiply by 'Øb'x */
return bitxor(bitxor(bitxor(r1,r2),r3),r4)

Mix7: procedure
parse arg word
if length(word) <> 4 then exit(23)
s.Ø = substr(word,1,1) ; s.1 = substr(word,2,1)
s.2 = substr(word,3,1) ; s.3 = substr(word,4,1)
r1 = mult(c2d(s.Ø),11) /* multiply by 'Øb'x */
r2 = mult(c2d(s.1),13) /* multiply by 'Ød'x */
r3 = mult(c2d(s.2),Ø9) /* multiply by 'Ø9'x */
r4 = mult(c2d(s.3),14) /* multiply by 'Øe'x */
return bitxor(bitxor(bitxor(r1,r2),r3),r4)
/* In the ShiftRows() transformation, the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets). The first row, Row Ø, is not shifted. */

ShiftRows: procedure
   parse arg state /* argument must be char, 16 bytes */
   if length(state) <> 16 then exit(8)
   s2 = substr(row(state,2),2,3) || substr(row(state,2),1,1)
   s3 = substr(row(state,3),3,2) || substr(row(state,3),1,2)
   s4 = substr(row(state,4),4,1) || substr(row(state,4),1,3)
   result = row(state,1) || s2 || s3 || s4 /* new rows */
   return row(result,1)||row(result,2)||row(result,3)||row(result,4)

InvShiftRows: procedure
   parse arg state /* argument must be char, 16 bytes */
   if length(state) <> 16 then exit(8)
   s2 = substr(row(state,2),4,1) || substr(row(state,2),1,3)
   s3 = substr(row(state,3),3,2) || substr(row(state,3),1,2)
   s4 = substr(row(state,4),2,3) || substr(row(state,4),1,1)
   result = row(state,1) || s2 || s3 || s4 /* new rows */
   return row(result,1)||row(result,2)||row(result,3)||row(result,4)

Row: procedure
   parse arg state,i /* argument must be char, 16 bytes */
   if length(state) <> 16 then exit(8)
   return substr(state,i,1)||substr(state,i+4,1)||, substr(state,i+8,1)||substr(state,i+12,1)

RotWord: procedure
   parse arg x /* argument must be char, 4 bytes */
   if length(x) <> 4 then exit(9)
   return right(x,3)||left(x,1)

/* Binary polynomial multiplication defined in the AES. */
/* Used only for decryption (InvMixcolumns function) */
mult: procedure
arg a,b              /* arguments must be decimal, result is char */
if a > 255 then say 'a=' a '(or' d2x(a) 'in hex) is in error'
if b > 255 then say 'b=' b '(or' d2x(b) 'in hex) is in error'
res = 'ØØ'x
if bitand('Ø1'x,d2c(b)) = 'Ø1'x then res = d2c(a)
a = xtime(a)
if bitand('Ø2'x,d2c(b)) = 'Ø2'x then res = bitxor(d2c(a),res)
a = xtime(a)
if bitand('Ø4'x,d2c(b)) = 'Ø4'x then res = bitxor(d2c(a),res)
a = xtime(a)
if bitand('Ø8'x,d2c(b)) = 'Ø8'x then res = bitxor(d2c(a),res)
a = xtime(a)
if bitand('1Ø'x,d2c(b)) = '1Ø'x then res = bitxor(d2c(a),res)
a = xtime(a)
if bitand('2Ø'x,d2c(b)) = '2Ø'x then res = bitxor(d2c(a),res)
a = xtime(a)
if bitand('4Ø'x,d2c(b)) = '4Ø'x then res = bitxor(d2c(a),res)
a = xtime(a)
if bitand('8Ø'x,d2c(b)) = '8Ø'x then res = bitxor(d2c(a),res)
return res

/*------------------------------------------------------------------*/
/* Function xtime                                                   */
/* Multiplication by x (ie,'ØØØØØØ1Ø' or 'Ø2') can be implemented */
/* at the byte level as a left shift and a subsequent conditional */
/* bitwise XOR with '1b'.                                           */
/*------------------------------------------------------------------*/
xtime: procedure
arg d                                   /* argument must be decimal */
if d > 255 then do
say 'Error, xtime called with argument=' d
exit(2ØØ)
end
if d < 128 then return d+d                            /* left shift */
else return c2d(bitxor(d2c(d+d-256),'1b'x))

/*------------------------------------------------------------------*/
/* The SubBytes() transformation is a non-linear byte substitution */
/* that operates independently on each byte of the state           */
/* using a substitution table (S-box).                            */
/*------------------------------------------------------------------*/
SubBytes: procedure
parse arg x                           /* argument must be character */
Sbox =
'63777bf26b6fc53001672bfed7ab76'x || ,
'ca82c97dafa5947f0add4a2af9c472c0'x || ,
'b7f99326363ff77cc34a5e5f171d83115'x || ,
'c0723c31896059a0712802eb27b275'x || ,
'09832c1ab6e5aa0523db6329e32f84'x || ,
'53d10ed2fcb156acb0e94a4c58cfx'x || ,
'd0efaaaab434d338545f9072f503c9fa8'x || ,
'51a3408f929d38f5bcb6d2a110fff3d2'x || ,
'cd0c13ec5f974417c4a77e3d645d1973'x || ,
return translate(x,Sbox)

InvSubBytes: procedure

/* InvSubBytes() is the inverse of the byte substitution transform. */
/* ation, in which the inverse S-box is applied to each byte */
/* of the state. */
/* *----------------------------------------------------------------*/
InvSubBytes: procedure

parse arg x                           /* argument must be character */

Sbox_inv =    '52096ad53036a538bf40a39e81f3d7fb'x || ,
'7ce339829b2ff87348e434acdeee9cb'x || ,
'547b94326a233dee4c950b42fac34e'x || ,
'082ea16628d924b2765ba2496d8bd125'x || ,
'72f6648668886164a455cc5d65b692'x || ,
'6c70480fddbd9da5e154657a78d984'x || ,
'90d88b008c6b30af7e45805b8b34506'x || ,
'd02c8ecfa3f0f02c1a9bd0301138a6b'x || ,
'a9111414f674dce97f2ccef04e673'x || ,
'96ac74227ad3585e2f93781c75d6f6e'x || ,
'47f11a711295c5896fb760ea18be11b'x || ,
'fc563e4bc6d27929adbc0fe7e8cd5af4'x || ,
'1fdda8338b07c731bl2l0592700ec5f'x || ,
'60517fa919b54a0d2e57a9f93c99cef'x || ,
'a0e03b4dace2af5b08ebbb3c83539961'x || ,
'172b047eba77d626e1691463552107cd'x

return translate(x,Sbox_inv)

/* Initial parameters ; we implement AES-128 here */
/* *----------------------------------------------------------------*/
init: procedure expose Nk  Nb  Nr  Rcon. trace

arg type

if type <> 128 & type <> 192 & type <> 256 then do
    say 'type=' type 'in error, must be 128, 192 or 256'
    exit(8)
end

/* Initialize values for AES-128 */
Nk = 4  /* Number of 32-bit words comprising the Cipher Key. For this */
     /* standard, Nk = 4, 6, or 8. (AES-128, AES-192, AES-256) */
Nb = 4  /* Number of columns (32-bit words) comprising the State. */
     /* For this standard, Nb = 4. */
Nr = 10  /* Number of rounds, which is a function of Nk and Nb (which */
        /* is fixed). For this standard, Nr = 10, 12, or 14. */
if type = 192 then do
   Nk = 6 ; Nr = 12
   /* AES-192 */
end

if type = 256 then do
   Nk = 8 ; Nr = 14
   /* AES-256 */
end

The round constant word array, Rcon[i], contains the values given by
[x**i-1,(Ø0),(Ø0),(Ø0)], with x**i-1 being powers of x (x is denoted
as (Ø2)) in the field GF(2**8))
*/
Rcon.1 = 'Ø1ØØØØØØ'x  ; Rcon.2 = 'Ø2ØØØØØØ'x

id = 2                                                    /* x = Ø2 */
do i = 3 to Nr
   id = xtime(id)                   /* compute all powers of x = Ø2 */
   Rcon.i = d2c(id) || 'ØØØØØØ'x
end

if trace = 'Y' then say 'Initialized for' type'-bit keys'
return

/*------------------------------------------------------------------*/
/* Key Expansion                                                    */
/*                                                                  */
/* The AES algorithm takes the Cipher Key, and performs a Key        */
/* Expansion routine to generate a key schedule. The Key Expansion  */
/* generates a total of Nb (Nr + 1) words: the algorithm requires    */
/* an initial set of Nb words, and each of the Nr rounds requires    */
/* Nb words of key data.                                            */
/*                                                                  */
/* Input = key            Output = "w." array (the key schedule)    */
/*------------------------------------------------------------------*/
Key_Expansion: procedure expose Nk Nb Nr Rcon. w. trace
parse arg key

key = left(key,4*Nk,'ØØ'x)   /* right padding to get max key length */
if trace = 'Y' then say 'Key =' c2x(key)
i = Ø
    /* create word array first entries */
do while i < Nk
      w.i = substr(key,4*i+1,4)
      i = i + 1
end

    /* populate other word array entries */
    i = Nk
    do while i < Nb*(Nr+1)
      j = i-1 ; temp = w.j
      if i // Nk = Ø then do
        j = i%Nk
        temp = bitxor(SubBytes(RotWord(temp)),Rcon.j)
      end
      else do
        if Nk = 8 & i // Nk = 4 then ,
temp = SubBytes(temp)
end
j = i - Nk  ;  w.i = bitxor(temp,w.j)
i = i + 1
end

/* list the key schedule */
i = Ø
do while i < Nb*(Nr+1)
    if trace = 'Y' then say 'w.'i '=' c2x(w.i)
i = i + 1
end
return

/*------------------------------------------------------------------
/* AES-enciphering a block of 16 bytes                             */
/*------------------------------------------------------------------*/
AES_cipher: procedure expose Nk  Nb  Nr  Rcon.  w. trace
parse arg input
if length(input) <> 16 then exit(1ØØ)
state = AddRoundKey(input,Ø)
do i = 1 to Nr-1
    state = SubBytes(state)
    if trace = 'Y' then say 'Round' i 'after subbytes   ' c2x(state)
    state = ShiftRows(state)
    if trace = 'Y' then say 'Round' i 'after shiftrows  ' c2x(state)
    state = MixColumns(state)
    if trace = 'Y' then say 'Round' i 'after Mixcolumns ' c2x(state)
    state = AddRoundKey(state,i)
    if trace = 'Y' then say 'Round' i 'after AddRoundkey' c2x(state)
end
i = Nr
state = SubBytes(state)
    if trace = 'Y' then say 'Round' i 'after subbytes   ' c2x(state)
    state = ShiftRows(state)
    if trace = 'Y' then say 'Round' i 'after shiftrows  ' c2x(state)
    state = AddRoundKey(state,i)
    if trace = 'Y' then say 'Round' i 'after AddRoundkey' c2x(state)
return state

/*------------------------------------------------------------------
/* AES-deciphering a block of 16 bytes                             */
/*------------------------------------------------------------------*/
AES_Inv_cipher: procedure expose Nk  Nb  Nr  Rcon.  w. trace
parse arg input
if length(input) <> 16 then exit(1ØØ)
state = AddRoundKey(input,Nr)
do i = Nr-1 to 1 by -1
    state = InvShiftRows(state)
    if trace = 'Y' then say 'Round' i 'after Invshiftrows ' c2x(state)
    state = InvSubBytes(state)
end

if trace = 'Y' then say 'Round' i 'after Invsubbytes' c2x(state)
state = AddRoundKey(state,i)
if trace = 'Y' then say 'Round' i 'after AddRoundkey' c2x(state)
state = InvMixColumns(state)
if trace = 'Y' then say 'Round' i 'after InvMixcolumns' c2x(state)
end

i = Ø
state = InvShiftRows(state)
if trace = 'Y' then say 'Round' i 'after Invshiftrows' c2x(state)
state = InvSubBytes(state)
if trace = 'Y' then say 'Round' i 'after Invsubbytes' c2x(state)
state = AddRoundKey(state,i)
if trace = 'Y' then say 'Round' i 'after AddRoundkey' c2x(state)
return state

REFERENCES
The references below may be helpful:

- The Rijndael Page: www.esat.kuleuven.ac.be/~rijmen/rijndael/
- NIST’s AES Home Page: csrc.nist.gov/encryption/aes/
- The Advanced Encryption Standard (Rijndael), by John Savard: home.ecn.ab.ca/~jsavard/crypto/co040801.htm

Thierry Falissard
MVS Consultant (France) © Xephon 20001

MVS Update on the Web

Code from individual articles of MVS Update, and complete issues in Acrobat PDF format, can be accessed on our Web site, at:

http://www.xephon.com/aixupdate.html

You will be asked to enter a word from the printed issue.

Using SVC screening to rename or delete a dataset without SYSDSN ENQ

INTRODUCTION
As a system programmer sometimes you need to copy members into a PDS (or a PDSE) which is already allocated by TSO users or STCs. But sometimes this PDS is too small, or its directory is not large enough to contain these new members. Then you get a X37 abend. So, you have to wait until all TSO users and STCs have deallocated the PDS, to be able to delete and recreate it in order to increase its size.

But, it is very well known that system programmers are not always so patient. In some situations (eg a test system, read-only dataset, etc), it is possible to replace the dataset ‘on-the-fly’. So, I wrote the following Assembler routine (NOSYSDSN) to get help in this situation.

This routine uses ‘SVC screening’, which is an MVS system facility described in the *MVS Programming: Authorized Assembler Services Guide*, to bypass SVC X'38' (ENQ) and SVC X'30' (DEQ).

SVC SCREENING CONCEPTS
Subsystem SVC screening allows a system routine to define those SVCs that a specific task can validly issue. When SVC screening is active for a task, the system determines, for each SVC issued by that task, whether the task can request that SVC function. If the SVC request is invalid, control is given to a special error subroutine supplied by the routine that activated the screening function.

SVC SCREENING IMPLEMENTATION
The task, executing under PSW key zero, activates SVC screening by setting two fields in the TCB for which screening is desired. The two fields consist of a screen flag bit and a one-word field containing the address of the SVC screen table, which provides the list of SVCs that the task cannot issue. The important SVC screening fields in the TCB are:
- TCBSVCS – a flag bit. When set to 1, it indicates that screening is in effect for this task.

- TCBSVCA2 – address of the subsystem screen table.

The task that needs SVC screening should obtain storage via GETMAIN for a 264 byte area called the subsystem screen table. To prevent a page fault, this area must come from the LSQA (subpool 253-255), the SQA (subpool 245), or must be in fixed storage. The subsystem screen table contains two areas as follows:

- SSTSVCN — subsystem SVC entry (8 bytes):
  - For bytes 0-3, bit zero is one of the following: O – indicates 24-bit addressing mode, and 1 indicates 31-bit addressing mode.
  - For bytes 0-3, bits 1-31 are the entry point address of the subsystem subroutine that will get control whenever a task has issued an SVC against which there is a screening restriction.
  - Byte 4 – X'00' is the subroutine is to run as a Type 1 SVC. X'08' means the subroutine may be used only by a program that is APF authorized, X'80' means the subroutine will execute as a Type 2 SVC, X'C0' means the subroutine will execute as a Type 3 or 4 SVC. And X'20' means the subroutine will execute as a Type 6 SVC.
  - Byte 5 is one of the following: X'00' – indicates that the SVC cannot be issued in AR mode, and X'80' – indicates that the SVC may be issued in AR mode.
  - With bytes 6-7 locks will be held on entry to the subroutine. If the appropriate lock bit is 1, the lock will be acquired by the SVC FLIH. The lock bits are: bit lock, 0 LOCAL, and 1 CMS. Bits 5-15 are always zero (off).

- SSTMASK — SVC screening mask (256 bytes):
  - With bytes 8-263 each byte corresponds to an SVC number in ascending order in the range 0-255. When the high order bit in a byte is 1, the task may validly issue the respective SVC; when the bit is zero, there is a screening restriction that prohibits the task from issuing the SVC.
USE OF SVC SCREENING BY NOSYSDSN

NOSYSDSN uses SVC screening to suppress SVC X'30' / X'38' calls. In order to protect volume VTOC, NOSYSDSN issues a RESERVE macro against the target volume before manipulating the dataset without ENQ. To allocate a dataset in an SMS pool, the corresponding storage class should be defined (temporarily) ‘GARANTED SPACE’ to be sure to RESERVE the right volume!

NOSYSDSN SOURCE

NOSYSDSN CSECT
NOSYSDSN AMODE 24
NOSYSDSN RMODE 24
* NAME: NOSYSDSN
* FUNCTION: SCRATCH, RENAME OR ALLOCATE A DATASET WITHOUT USING
* USING SYSDSN ENQ
* * THE DATASET CAN BE SMS OR NOT SMS MANAGED.
* * THE DATASET CAN BE CATALOGUED OR NOT CATALOGUED.
* * IF THE DATASET IS SMS MANAGED, IT WILL BE AUTOMATICALLY
* UNCATALOG.
* EXAMPLE:
* * //STEPØ1 EXEC PGM=NOSYSDSN
* //STEPLIB DD DISP=SHR,DSN=SYS2.LINKLIB
* //SYSPRINT DD SYSOUT=*"*/
* */ DCB MODEL FOR ALLOCATION REQUEST
* */
* ///MOD1 DD DISP=(,PASS),DSN=&amp;MOD1,
* // DCB=(LRECL=80,RECFM=FB,BLKSIZE=27920),
* // SPACE=(TRK,(1,1,20))
* */
* ///SYSIN DD *
* * DELETE DSN=TMPS50.TEST.OLD VOL=TMPS50 Y
* RENAME DSN=TMPS50.TEST VOL=TMPS50 Y
* NEW=TMPS50.TEST.OLD
* ALLOC DSN=TMPS50.TEST VOL=TMPS50 Y
* SPACE=(CYL,(0020,0005,0000))
* DCBMOD=MOD1
* */
* * REGISTER USAGE:
* *
* R2 WORK REGISTER
* R3 WORK REGISTER
* R4  WORK REGISTER
* R5  WORK REGISTER
* R8  TCB
* R9  UCB
* R10 FOR BAL
* R11 BASE REGISTER
* R12 BASE REGISTER
  SAVE (14,12)
  BASR R12,Ø
BR12 EQU *
  USING *,R12
LA  R11,Ø95(R12)
LA  R11,1(R11)
USING BR12+Ø96,R11
GETMAIN R,LV=WORKL
ST  R1,8(R13)
ST  R13,4(R1)
LR  R13,R1
USING WORK,R13

* —————————————————————————————————————————————————————————————————*
* SET RC
* —————————————————————————————————————————————————————————————————*
SR  RØ,RØ  DEFAULT RC = Ø
ST  RØ,RET_CODE

* —————————————————————————————————————————————————————————————————*
* OPEN FILES
* —————————————————————————————————————————————————————————————————*
OPEN  (SYSIN,(INPUT))
OPEN  (SYSPRINT,(OUTPUT))

* —————————————————————————————————————————————————————————————————*
* POINT TO SYSTEM AREA
* —————————————————————————————————————————————————————————————————*
SR  R8,R8
USING PSA,R8
L  R7,FLCCVT
USING CVTMAP,R7
L  R6,CVTTCBP
L  R8,4(R6)  GET CURRENT TCB
USING TCB,R8

* —————————————————————————————————————————————————————————————————*
* GETMAIN WORKAREA IN SQA SUBPOOL 245 FOR SVC SCREENING
* —————————————————————————————————————————————————————————————————*
AUTHON GET AUTHORIZED VIA AUTH SVC
MODESET KEY=ZERO,MODE=SUP
GETMAIN R,SP=245,LV=SSTLEN
ST  R1,SVCADDR  STORE ADDRESS OF WORKAREA
LR  R2,R1
L  R3,=A(SSTLEN)
LA  R4,SSTSVCN  POINT TO SVC TABLE MODEL
LR  R5,R3
MVCL R2,R4  COPY SVC TABLE
L  R2,SVCADDR

LA R15,SSTPGM-SSTSVCN(R2)      POINT TO RECOVERY PROGRAM
STCM R15,B'1111',Ø(R2)         STORE IT IN SVC TABLE
STCM R2,B'1111',TCBSVCA2       STORE SVC TABLE IN TCB

READREC EQU *

* READ SYSIN RECORD

GET SYSIN,IRECORD
CLC IRECORD(Ø1),=C'**'       COMMENT ?
BE READREC                   YES, READ NEXT RECORD
BAL R1Ø,PARSEIN              PARSE SYSIN RECORD
BAL R1Ø,WRITEIN              COPY TO SYSPRINT

LA R9,UCBAREA
USING UCBCMSEG,R9

XC UCBWORK,UCBWORK           CLEAR WORK AREA
XC UCBAREA,UCBAREA           CLEAR WORK AREA

UCBSCAN COPY,                 X
VOLSER=OVOL,                 X
WORKAREA=UCBWORK,            X
UCBAREA=UCBAREA,             X
DEVCLASS=DASD,               X
DYNAMIC=YES,                 X
RANGE=ALL

LTR R15,R15
BZ UCBOK

MVC ORECORD,BLANKS
MVC ORECORD,MSGØ1ØE
MVC ORECORD+18(Ø6),OVOL
PUT SYSPRINT,ORECORD
B READREC

UCBOK EQU *

* GET UCB ADDRESS VOR RESERVE

MVC VOLCUA,UCBCHAN
XC UCBWORK,UCBWORK           CLEAR WORK AREA

UCBSCAN ADDRESS,              X
WORKAREA=UCBWORK,            X
UCBPTR=UADDR,                X
DEVN=VOLCUA,                 X
DEVCLASS=DASD,               X
DYNAMIC=YES,                 X
NOPIN,                       X
LOC=ANY,                     X
RANGE=ALL

* SMS VOLUME?

TM UCBFL5,UCBSMS
BO SMSOK
MVC SMSF,=C'N'
MVC ORECORD,BLANKS
MVC ORECORD,MSGØ3II
PUT SYSPRINT,ORECORD
B PVOLLI

SMSOK EQU *
MVC SMSF,=C'Y'
MVC ORECORD,BLANKS
MVC ORECORD,MSGØ30I
PUT SYSPRINT,ORECORD

PVOLLI EQU *

* PREPARE VOLUME LIST *
* *
MVC VOLENT,=H'1' 1 VOLUME
MVC VOLSTAT,=H'Ø' RESET STATUS BYTE
MVC VOLSER,OVOL COPY VOLSER
MVC VOLTYPE,UCBTyp COPY VOLUME TYPE

* WHICH FUNCTION IS CALLED? *
* *
CLC FUNCTION,=CL6'RENAME'
BE RENAME
CLC FUNCTION,=CL6'DELETE'
BE DELETE
CLC FUNCTION,=CL6'ALLOC'
BE ALLOC

NEXTREC EQU *
BAL R1Ø,SKIPLINE
B READREC

SETRC8 EQU *
LA R15,8 SET RC = 8
ST R15,RET_CODE
B END

* RENAME FUNCTION *
* *
RENAME EQU *
BAL R1Ø,SVCON
SR RØ,RØ SET RØ TO Ø
RENAME CAMLSTR RENAME DATASET
LR R2,R15
BAL R1Ø,SVCOFF
LTR R2,R2
BZ RENAOK
BAL R1Ø,RC1
MVC ORECORD,BLANKS
MVC ORECORD,MSGØ2ØE
PUT SYSPRINT,ORECORD
MVC ORECORD,BLANKS
MVC ORECORD,MSGØ22E
MVC ORECORD+28(Ø2),WRC_4+1 COPY RC
MVC ORECORD+42(Ø2),WRSN_4+1 COPY RSN
PUT SYSPRINT,ORECORD
```
B     SETRC8
RENAOK EQU *
MVC ORECORD,BLANKS
MVC ORECORD,MSG0201
PUT SYSPRINT,ORECORD
BAL R10,CATALOG
B   NEXTREC

* DELETE FUNCTION
*   __________________________________________________________
DELETE EQU *
BAL R10,SVCON            TURN ON SVC SCREENING
SR R0,R0                SET R0 TO Ø
SCRATCH CAMLSTD          DELETE DATASET
LR R2,R15
BAL R10,SVCOFF
LTR R2,R2
BZ DELOK
BAL R10,RC1
BAL R10,SVCOFF          TURN ON SVC SCREENING
MVC ORECORD,BLANKS
MVC ORECORD,MSG021E
PUT SYSPRINT,ORECORD
MVC ORECORD,BLANKS
MVC ORECORD,MSG023E
MVC ORECORD+28(Ø2),WRC_4+1 COPY RC
MVC ORECORD+42(Ø2),WRSN_4+1 COPY RSN
PUT SYSPRINT,ORECORD
B   SETRC8
DELOK EQU *
MVC ORECORD,BLANKS
MVC ORECORD,MSG0211
PUT SYSPRINT,ORECORD
BAL R10,CATALOG
B   NEXTREC

* ALLOC FUNCTION
*   __________________________________________________________
ALLOC EQU *
LA R2,RBLOCK
ST R2,RBLOCKP
OI RBLOCKP,S99RBPN
LA R3,RBLOCK
USING S99RB,R3
LA R4,RBLOCKX          REQUEST BLOCK EXTENSION
USING S99RBX,R4
XC S99RB(RBLEN),S99RB
XC S99RBX(RBXLEN),S99RBX
MVI S99RBLN,RBLEN
MVI S99VERB,S99VRBAL
ST R4,S99999X
MVC S99EID,='CL6'S99RBX'
```
OI S99EVER,S99RBXVR
OI S99EOPTS,S99EIMSG
OI S99EOPTS,S99EWTP
OI S99EMGSV,S99XINFO

ISSUE ALL SMS MESSAGES

LA R4,TUP_1
ST R4,S99TXTTPP
LA R5,TU_1
ST R5,TUP_1
LA R5,TU_2
ST R5,TUP_2
LA R5,TU_3
ST R5,TUP_3
LA R5,TU_4
ST R5,TUP_4
LA R5,TU_5
ST R5,TUP_5
LA R5,TU_6
ST R5,TUP_6
LA R5,TU_7
ST R5,TUP_7
LA R5,TU_8
ST R5,TUP_8
LA R5,TU_9
ST R5,TUP_9
LA R5,TU_A
ST R5,TUP_A
OI TUP_A,S99TUPLN

MVC TU_1(TUM_1_L),TUM_1
MVC TU_2(TUM_2_L),TUM_2
MVC TU_3(TUM_3_L),TUM_3
MVC TU_4(TUM_4_L),TUM_4
MVC TU_5(TUM_5_L),TUM_5
MVC TU_6(TUM_6_L),TUM_6
MVC TU_7(TUM_7_L),TUM_7
MVC TU_8(TUM_8_L),TUM_8
MVC TU_9(TUM_9_L),TUM_9
MVC TU_A(TUM_A_L),TUM_A

CLC NCAT,=C'Y' DISP = ( ,CATLG )
BE DISPCAT
MVC TU_2(TUM_2K_L),TUM_2K
B DISPGO

DISPCAT EQU *
MVC TU_2(TUM_2_L),TUM_2

DISPGO EQU *
PCK PW4,NPRIM
ZAP PW8,PW4
CVB R5,PWB
ST R5,XPRIM
PCK PW4,NSEC
ZAP PW8,PW4
CVB R5,PWB
ST R5,XSEC
PACK  PW4,NDIR
ZAP  PW8,PW4
CVB  R5,PW8
ST   R5,XDIR
MVC  TU_3+6(44),NDSN
MVC  TU_4+6(Ø8),NDCBMOD
MVC  TU_6+6(Ø3),XPRIM+1
MVC  TU_7+6(Ø3),XSEC+1
MVC  TU_8+6(Ø3),XDIR+1
MVC  TU_A+6(Ø6),OVOL
BAL  R1Ø,SVCON            TURN ON SVC SCREENING
LA   R1,RBLOCKP
DYNALLOC
LR   R2,R15
BAL  R1Ø,SVCOFF
LTR  R2,R2
BZ   ALLOCOK
MVC  ORECORD,BLANKS
MVC  ORECORD,MSGØ24E
PUT   SYSPRINT,ORECORD
B     SETRC8
ALLOCOK EQU *
MVC  ORECORD,BLANKS
MVC  ORECORD,MSGØ24I
PUT   SYSPRINT,ORECORD
B     NEXTREC

* —————————————————————————————————————————————————————————————————*
*                      ENDYSIN EQU *
* —————————————————————————————————————————————————————————————————*
L     R1,SVCADDR            RELEASE SQA STORAGE
FREEMAIN R,LV=SSTLEN,A=(R1),SP=245

* —————————————————————————————————————————————————————————————————*
* SCRATCH MACRO SETS THE TCBFJMC BIT (STEP-MUST-COMPLETE).  *
* THE STATUS MACRO IS USED TO RESET IT.                       *
* —————————————————————————————————————————————————————————————————*
*    STATUS RESET,MC,STEP
*    MODESET KEY=NZERO,MODE=PROB
*    AUTHOFF
* —————————————————————————————————————————————————————————————————*
* CLOSE FILES                                                   *
* —————————————————————————————————————————————————————————————————*
CLOSE SYSIN
CLOSE SYSPRINT
RETURN  EQU *
L     R2,RET_CODE                 GET RC
L     R13,4(R13)                 RESTORE R13
L     R1,8(R13)
FREEMAIN R,LV=WORKL,A=(R1)
LR   R15,R2                     SET RC
L    R14,12(R13)
LM   R0,R12,20(R13)
*   SR   R15,R15           SET UP RC
    BSM  0,R14              RETURN TO MVS AND USE RC=R15

*=================================================================
* TURN ON SVC SCREENING
*=================================================================
SVCON  EQU  *

* FIRST, PROTECT THE VTOC USING RESERVE
*

    LA  R3,OVOL
    LA  R4,UADDR
RESERVE (SYSVTOC,(R3),E,6,SYSTEMS), LOC=ANY,UCB=(R4)

*=================================================================
* NOW, WE CAN TURN ON SVC SCREENING
*

    OI   TCBFLGS7,TCBSVCS SVC SCREENING IS ON
    BR  R10

*=================================================================
* TURN OFF SVC SCREENING
*

SVCOFF  EQU  *

* FIRST, WE TURN OFF SVC SCREENING
*

    NI   TCBFLGS7,255-TCBSVCS SVC SCREENING IS OFF
    LA  R3,OVOL

*=================================================================
* THEN, WE CAN RELEASE THE VTOC RESERVE
*

    DEQ (SYSVTOC,(R3),6,SYSTEMS),RMC=STEP
    BR  R10

*=================================================================
* PARSE SYSIN RECORD
*

PARSEIN  EQU  *
    CLC   IRECORD(Ø6),=CL6'RENAME'
    BNE   PEØ1
    MVC   FUNCTION,IRECORD
    MVC   ODSN,IRECORD+11
    MVC   OVOL,IRECORD+6Ø
    MVC   NCAT,IRECORD+67
    CLC   NCAT,=CL1'N'
    BE    FNCAT1
    MVC   NCAT,=CL1'Y'
FNCAT1   EQU  *
    GET   SYSIN,IRECORD
    MVC   NDSN,IRECORD+11
    BR   R1Ø
    PEØ1  EQU  *
CLC IRECORD(06),=CL6'DELETE'
BNE PE02
MVC FUNCTION,IRECORD
MVC ODSN,IRECORD+11
MVC OVOL,IRECORD+60
MVC NCAT,IRECORD+67
CLC NCAT,=CL1'N'
BE FNCAT2
MVC NCAT,=CL1'Y'

FNCAT2 EQU *
BR R10

PE02 EQU *
CLC IRECORD(06),=CL6'ALLOC'
BNE PE99
MVC FUNCTION,IRECORD
MVC NDSN,IRECORD+11
MVC OVOL,IRECORD+60
MVC NCAT,IRECORD+67
CLC NCAT,=CL1'N'
BE FNCAT3
MVC NCAT,=CL1'Y'

FNCAT3 EQU *
GET SYSIN,IRECORD
MVC NUNIT,IRECORD+14
MVC NPRIM,IRECORD+19
MVC NSEC,IRECORD+24
MVC NDIR,IRECORD+29
GET SYSIN,IRECORD
MVC NDCBMOD,IRECORD+14
BR R10

PE99 EQU *
MVC ORECORD,BLANKS
MVC ORECORD,MSG005E
MVC ORECORD+29(06),IRECORD
PUT SYSPRINT,ORECORD
LA R15,8
ST R15,RET_CODE
B END

WRITEIN EQU *
MVC ORECORD,BLANKS
MVC ORECORD,MSG009I
MVC ORECORD+23(06),FUNCTION
PUT SYSPRINT,ORECORD
CLC FUNCTION,=CL6'DELETE'
BE WDELETE
CLC FUNCTION,=CL6'ALLOC'
BE WALLOC
MVC ORECORD,BLANKS
MVC ORECORD,MSG010I
MVC ORECORD+23(44),ODSN
PUT SYSPRINT,ORECORD
MVC ORECORD,BLANKS
MVC ORECORD,MSGØ11I
MVC ORECORD+23(44),NDSN
PUT SYSPRINT,ORECORD
MVC ORECORD,BLANKS
MVC ORECORD,MSGØ12I
MVC ORECORD+23(Ø6),OVOL
PUT SYSPRINT,ORECORD
MVC ORECORD,BLANKS
MVC ORECORD,MSGØ14I
MVC ORECORD+23(Ø1),NCAT
PUT SYSPRINT,ORECORD
B WRETURN

WDELETE EQU *
MVC ORECORD,BLANKS
MVC ORECORD,MSGØ13I
MVC ORECORD+23(44),ODSN
PUT SYSPRINT,ORECORD
MVC ORECORD,BLANKS
MVC ORECORD,MSGØ12I
MVC ORECORD+23(Ø6),OVOL
PUT SYSPRINT,ORECORD
MVC ORECORD,BLANKS
MVC ORECORD,MSGØ14I
MVC ORECORD+23(Ø1),NCAT
PUT SYSPRINT,ORECORD
B WRETURN

WALLOC EQU *
MVC ORECORD,BLANKS
MVC ORECORD,MSGØ11I
MVC ORECORD+23(44),NDSN
PUT SYSPRINT,ORECORD
MVC ORECORD,BLANKS
MVC ORECORD,MSGØ12I
MVC ORECORD+23(Ø6),OVOL
PUT SYSPRINT,ORECORD
MVC ORECORD,BLANKS
MVC ORECORD,MSGØ14I
MVC ORECORD+29(Ø3),NUNIT
MVC ORECORD+34(Ø4),NPRIM
MVC ORECORD+39(Ø4),NSEC
MVC ORECORD+44(Ø4),NDIR
PUT SYSPRINT,ORECORD
MVC ORECORD,BLANKS
MVC ORECORD,MSGØ15I
MVC ORECORD+29(Ø3),NUNIT
MVC ORECORD+34(Ø4),NPRIM
MVC ORECORD+39(Ø4),NSEC
MVC ORECORD+44(Ø4),NDIR
PUT SYSPRINT,ORECORD
MVC ORECORD,BLANKS
MVC ORECORD,MSGØ16I
MVC ORECORD+23(Ø8),NDCBMOD
PUT SYSPRINT,ORECORD
MVC ORECORD,BLANKS
MVC ORECORD,MSGØ14I
MVC ORECORD+23(Ø1),NCAT
PUT SYSPRINT,ORECORD
B WRETURN
WRETURN EQU *
BR R10
SKIPLINE EQU *
MVC ORECORD,BLANKS
PUT SYSPRINT,ORECORD
BR R10
CATALOG EQU *
CLC SMSF,=C'Y'
BE CRETURN
CLC NCAT,=CL1'N'
BE CRETURN
CATALOG CAMLSTU
LTR R15,R15
BNZ UNNOK
CLC FUNCTION,=CL6'RENAME'
BNE CRETURN
CATALOG CAMLSTC
LTR R15,R15
BNZ CANOK
CRETURN EQU *
BR R10
CANOK EQU *
UNNOK EQU *
* MVC ORECORD,BLANKS
* MVC ORECORD,MSG013I
* MVC ORECORD+23(44),ODSN
* PUT SYSPRINT,ORECORD
B CRETURN

*————————————————————————————————————————————————————————————————*
* PROCESS SCRATCH AND RENAME RC AND RSN                          *
*————————————————————————————————————————————————————————————————*
RC1 EQU *
CVD R15.DOUBLE
MVC WRC_4,MASK4
ED WRC_4.DOUBLE+6
SR R15,R15
LH R15,VOLSTAT
CVD R15.DOUBLE
MVC WRSN_4,MASK4
ED WRSN_4.DOUBLE+6
BR R10
LTORG

*————————————————————————————————————————————————————————————————*
* FILE DEFINITIONS                                               *
*————————————————————————————————————————————————————————————————*
SYSIN DCB DDNAME=SYSIN,
            DSORG=PS,
            MACRF=GM,
            RECFM=FB,
            LRECL=80,
            SEQUENTIAL X
            INPUT X
            X
EODAD=ENDSYSIN
END OF DATA BRANCH

SYSPRINT DCB
DDNAME=SYSPRINT,
DD NAME X
DSORG=PS,
SEQUENTIAL X
MACRF=PM,
OUTPUT X
RECFM=FBA,
LRECL=133

*—* MESSAGES —* *
* 0 0 0 0 0 0 *
* MSGØ05E DC CL133' MSGØ05E - FUNCTION XXXXXX NOT SUPPORTED' *
* MSGØ10E DC CL133' MSGØ10E - VOLUME XXXXXX NOT FOUND' *
* MSGØ20E DC CL133' MSGØ20E - DATASET NOT RENAMED' *
* MSGØ21E DC CL133' MSGØ21E - DATASET NOT DELETED' *
* MSGØ22E DC CL133' MSGØ22E - RENAME - RC = X''XX'' / RSN = X''XX'' *
* MSGØ23E DC CL133' MSGØ23E - SCRATCH - RC = X''XX'' / RSN = X''XX'' *
* MSGØ24E DC CL133' MSGØ24E - DATASET NOT ALLOCATED' *
* MSGØ30E DC CL133' MSGØ30E - ERROR DURING CATALOG' *
* MSGØ09I DC CL133' MSGØ09I - FUNCTION : ' *
* MSGØ10I DC CL133' MSGØ10I - OLD DSN : ' *
* MSGØ11I DC CL133' MSGØ11I - NEW DSN : ' *
* MSGØ12I DC CL133' MSGØ12I - VOLSER : ' *
* MSGØ13I DC CL133' MSGØ13I - DSN : ' *
* MSGØ14I DC CL133' MSGØ14I - CATALOG : ' *
* MSGØ15I DC CL133' MSGØ15I - SPACE : SPACE(XXX,(XXXX,XXXX,XXXX))X ' *
* MSGØ16I DC CL133' MSGØ16I - DCBMOD : ' *
* MSGØ20I DC CL133' MSGØ20I - DATASET RENAMED SUCCESSFULLY' *
* MSGØ21I DC CL133' MSGØ21I - DATASET DELETED SUCCESSFULLY' *
* MSGØ24I DC CL133' MSGØ24I - DATASET ALLOCATED SUCCESSFULLY' *
* MSGØ30I DC CL133' MSGØ30I - DATASET IS SMS MANAGED' *
* MSGØ31I DC CL133' MSGØ31I - DATASET IS NOT SMS MANAGED' *

SYSVTOC DC CL8'SYSVTOC'
BLANKS DC CL133''

*—* MASK —* *
* —* *
* MASK4 DC X'21202020' *
* —* *
* —* *
* CAMLST MACRO INSTRUCTIONS —* *
* —* *
* CAMLSTR CAMLST RENAME,ODSN,NDSN,VOLIST CAMLST FOR RENAME
**SVC SCREENING TABLE**

<table>
<thead>
<tr>
<th>SSTSVCN</th>
<th>DC</th>
<th>A(SSTPGM) ADDRESS OF ERROR ROUTINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC</td>
<td>X'ØØ' EMULATE TYPE-2 SVC</td>
<td></td>
</tr>
<tr>
<td>DC</td>
<td>X'ØØ' AR MODE NOT ALLOWED</td>
<td></td>
</tr>
<tr>
<td>DC</td>
<td>X'ØØØØ' NO LOCKS</td>
<td></td>
</tr>
</tbody>
</table>

**SVC SCREEN MASK TABLE**

Entries marked X'ØØ' allow the SVC to be executed, those marked X'8Ø' cause control to be passed to SSTPGM.

**SVC 3Ø AND 3Ø ARE NOT EXECUTED**

*SVC 3Ø = DEQ
*SVC 3Ø = ENQ*

**SVC SCREEN RECOVERY ROUTINE**

*SSTPGM DC X'17FF' ASM CODE FOR 'XR R15,R15'
DC X'Ø7FE' ASM CODE FOR 'BR R14'
SSTLEN EQU *-SSTSVCN LENGTH OF STORAGE REQUIRED*

**TEXT UNIT MODELS FOR SVC 99**

*TUM_1 EQU DISP=(NEW, )
DC AL2(DALSTATS)*
DC X'0001'
DC X'0001'
DC X'04'

TUM_1_L EQU *-TUM_1
*

TUM_2 EQU *

TC     DISP=(   ,CATLG)
DC     AL2(DALNDISP)
DC     X'0001'
DC     X'0001'
DC     X'02'
CATLG

TUM_2_L EQU *-TUM_2
*

TUM_2K EQU *

TC     DISP=(   ,KEEP)
DC     AL2(DALNDISP)
DC     X'0001'
DC     X'0001'
DC     X'08'
KEEP

TUM_2K_L EQU *-TUM_2K
*

TUM_3 EQU *

TC     DSN=XXXXXXXXXXXXXXXX
DC     AL2(DALDSNAM)
DC     X'0001'
DC     X'002C'
X'2C' = 44
DC     CL44''
XXXXXXX

TUM_3_L EQU *-TUM_3
*

TUM_4 EQU *

TC     DCB=*..XXX
DC     AL2(DALDCBDD)
DC     X'0001'
DC     X'0008'
DC     CL8''
XXXXXXX

TUM_4_L EQU *-TUM_4
*

TUM_5 EQU *

TC     SPACE=(CYL,(   ,   ))
DC     AL2(DALCYL)
DC     X'0000'

TUM_5_L EQU *-TUM_5
*

TUM_6 EQU *

TC     SPACE=(   , (X,   ))
DC     AL2(DALPRIME)
DC     X'0001'
DC     X'0003'
DC     X'000000'
X

TUM_6_L EQU *-TUM_6
*

TUM_7 EQU *

TC     SPACE=(   , (   ,X,   ))
DC     AL2(DALSECND)
DC     X'0001'
DC     X'0003'
DC     X'000000'
X

TUM_7_L EQU *-TUM_7
*
TUM_8  EQU *  SPACE=( , ( , X))
  DC AL2(DALDIR)
  DC X'0001'
  DC X'0003'
  DC X'000000'
X
TUM_8_L EQU *-TUM_8
*
TUM_9  EQU *  UNIT=SYSALLDA
  DC AL2(DALUNIT)
  DC X'0001'
  DC X'0008'
  DC CL8'SYSSLDA'
TUM_9_L EQU *-TUM_9
*
TUM_A  EQU *  VOL=SER=XXXXXX
  DC AL2(DALVLSER)
  DC X'0001'
  DC X'0006'
  DC CL6''
TUM_A_L EQU *-TUM_A
*
*
FUNCTION DS  CL6
  ODSN  DS CL44          OLD DSN
  OVOL  DS CL6           VOLSER
  NDSN  DS CL44          NEW DSN
  NCAT  DS CL1           NOT CATALOGUED
*
  NUNIT  DS CL3
  NPRIM  DS CL4
  NSEC  DS CL4
  NDIR  DS CL4
  NDCBMOD DS CL8
*
  XPRIM  DS F
  XSEC  DS F
  XDIR  DS F
*
  PW4  DS PL4
  PW8  DS PL8
*
  VOLIST  DS ØF
  VOLENT  DS H
  VOLTYPE  DS CL4
  VOLSER  DS CL6
  VOLSTAT  DS H
*
* WTO TO DEBUG
*
WTOC  WTO  '        X
WTOL EQU *-WTOC  LENGTH OF MACRO EXPANSION
*
WTO DS CL(WTOL)
*
* WORKAREA AND SAVEAREA
* —————————————————————————————————————————————————————————————————*
WORK DSECT
SAVEAREA DS 18F
IRECORD DS CL80  SYSIN RECORD
ORECORD DS CL133  SYSPRINT RECORD
UCBWORK DS CL100  FOR UCBSCAN
UCBAREA DS CL48  FOR UCBSCAN
SMSF DS C  SMS FLAG
RET_CODE DS F  RETURN CODE
DOUBLE DS D
WRC_4 DS CL4
WRSN_4 DS CL4
SVADDR DS F  ADDRESS OF SQA AREA
UADDR DS F  UCB ADDRESS
VOLCUA DS H
RBLOCKP DS F  REQUEST BLOCK POINTER
RBLOCK DS CL(RBLEN)  REQUEST BLOCK
   DS F
RBLOCKX DS CL(RBXLEN)  REQUEST BLOCK EXTENSION
*
TUP_1 DS F  TEXT UNIT POINTER
TUP_2 DS F  TEXT UNIT POINTER
TUP_3 DS F  TEXT UNIT POINTER
TUP_4 DS F  TEXT UNIT POINTER
TUP_5 DS F  TEXT UNIT POINTER
TUP_6 DS F  TEXT UNIT POINTER
TUP_7 DS F  TEXT UNIT POINTER
TUP_8 DS F  TEXT UNIT POINTER
TUP_9 DS F  TEXT UNIT POINTER
TUP_A DS F  TEXT UNIT POINTER
TU_1 DS CL(TUM_1_L)
TU_2 DS CL(TUM_2_L)
TU_3 DS CL(TUM_3_L)
TU_4 DS CL(TUM_4_L)
TU_5 DS CL(TUM_5_L)
TU_6 DS CL(TUM_6_L)
TU_7 DS CL(TUM_7_L)
TU_8 DS CL(TUM_8_L)
TU_9 DS CL(TUM_9_L)
TU_A DS CL(TUM_A_L)
WORKL EQU *-WORK
RBLEN EQU S99RBEND-S99RB
RBXLEN EQU 36
* —————————————————————————————————————————————————————————————————*

DF/SMS MACRO RETURN AND STATUS CODES

* THESE INSTRUCTIONS ARE DOCUMENTED IN:
  * DFSMS/MVS DFSMSDFP ADVANCED SERVICES

* RENAME

* RETURN CODE:

* X'00' THE DATASET HAS BEEN SUCCESSFULLY RENAMED
* X'04' NO VOLUME CONTAINING ANY PART OF THE DATASET WAS
  MOUNTED
* X'08' AN UNUSUAL CONDITION WAS ENCOUNTERED ON ONE OR MORE
  VOLUMES
* X'0C' ONE OF THE FOLLOWING CONDITIONS OCCURRED:
  * THE DADSM RENAME PARAMETER LIST IS NOT VALID.
  * THE VOLUME LIST IS NOT VALID.
  * AT ENTRY TO RENAME, REGISTER 0 WAS NOT ZERO AND
    DID NOT POINT TO A VALID UCB.

* STATUS CODE:

* X'00' THE FORMAT-1 DSCB FOR THE DATASET HAS BEEN RENAMED
  IN THE VTOC ON THIS VOLUME.
* X'01' THE VTOC OF THIS VOLUME DOES NOT CONTAIN THE
  FORMAT-1 DSCB OF THE DATASET TO BE RENAMED.
* X'02' ONE OF THE FOLLOWING CONDITIONS OCCURRED:
  * THE DATASET COULD NOT BE RENAMED BECAUSE THE
    DATASET WAS PASSWORD PROTECTED AND THE PASSWORD
    WAS NOT SUPPLIED IN THE TWO ATTEMPTS ALLOWED.
  * AN ATTEMPT WAS MADE TO RENAME A VSAM DATA SPACE
    OR AN INTEGRATED CATALOG FACILITY VSAM DATASET.
  * AN ATTEMPT WAS MADE TO RENAME A VTOC INDEX DATA
    SET.
  * AN SMS-VALIDATION FAILURE OCCURRED.
* X'03' A FORMAT-1 DSCB CONTAINING THE NEW DATASET NAME
  ALREADY EXISTS IN THE VTOC OF THIS VOLUME, OR AN
  ATTEMPT WAS MADE TO RENAME A DATASET TO A NAME
  STARTING WITH SYS1.VTOCIX.
* X'04' ONE OF THE FOLLOWING CONDITIONS OCCURRED:
  * A PERMANENT I/O ERROR OCCURRED WHILE TRYING TO
    RENAME THE DATASET ON THIS VOLUME.
  * AN INVALID FORMAT-1 DSCB WAS ENCOUNTERED WHILE
    PROCESSING THIS VOLUME.
  * NO SPACE IS AVAILABLE IN THE INDEX FOR THE
* NEW NAME, AND NO ADDITIONAL VIERS ARE AVAILABLE.
* X'05' IT COULD NOT BE VERIFIED THAT THIS VOLUME WAS
  MOUNTED NOR WAS A UNIT AVAILABLE FOR MOUNTING THE
  VOLUME.
* X'06' THE OPERATOR WAS UNABLE TO MOUNT THIS VOLUME.
* X'07' THE DATASET WAS NOT RENAMED, BECAUSE IT WAS
  CURRENTLY OPEN FOR PROCESSING.
* X'08' THE DATASET IS DEFINED TO RACF, BUT EITHER YOU ARE
  NOT AUTHORIZED TO THE DATASET OR THE DATASET IS
  DEFINED TO RACF ON MULTIPLE VOLUMES.

* SCRATCH
*    -------------------
*    RETURN CODE:
*    
*    X'00' THE DATASET HAS BEEN SUCCESSFULLY DELETED.
*    X'04' NO VOLUME CONTAINING ANY PART OF THE DATASET WAS
  MOUNTED.
*    X'08' AN UNUSUAL CONDITION WAS ENCOUNTERED ON ONE OR MORE
  VOLUMES.
*    X'0C' ONE OF THE FOLLOWING CONDITIONS OCCURRED:
*        * THE SCRATCH PARAMETER LIST IS NOT VALID.
*        * THE VOLUME LIST IS NOT VALID.
*        * AT ENTRY TO SCRATCH, REGISTER 0 WAS NOT ZERO AND
          DID NOT POINT TO A VALID UCB.

* STATUS CODE:
*    
*    X'00' THE FORMAT-1 DSCB FOR THE DATASET HAS BEEN DELETED
  IN THE VTOC ON THIS VOLUME.
*    X'01' THE VTOC OF THIS VOLUME DOES NOT CONTAIN THE
  FORMAT-1 DSCB OF THE DATASET TO BE RENAMED.
*    X'02' ONE OF THE FOLLOWING CONDITIONS OCCURRED:
*        * THE DATASET COULD NOT BE DELETED BECAUSE THE
          DATASET WAS PASSWORD PROTECTED AND THE PASSWORD
          WAS NOT SUPPLIED IN THE TWO ATTEMPTS ALLOWED.
*        * AN ATTEMPT WAS MADE TO DELETE A VSAM DATA SPACE
          OR AN INTEGRATED CATALOG FACILITY VSAM DATASET.
*        * AN ATTEMPT WAS MADE TO DELETE A VTOC INDEX DATASET.
*        * AN SMS-VALIDATION FAILURE OCCURRED.
*    X'03' THE DATASET WAS NOT DELETED BECAUSE EITHER THE OVRD
  OPTION WAS NOT SPECIFIED OR THE RETENTION CYCLE HAD
  NOT EXPIRED.
*    X'04' ONE OF THE FOLLOWING CONDITIONS OCCURRED:
*        * AN INVALID FORMAT-1 DSCB WAS ENCOUNTERED WHILE
          PROCESSING THIS VOLUME.
*        * AN UNEXPECTED CVAF ERROR RETURN CODE WAS ENCOUNTERED.
*        * AN INSTALLATION EXIT REJECTED THE REQUEST.
*        * A PERMANENT I/O ERROR OCCURRED WHILE TRYING TO
          DELETE THE DATASET ON THIS VOLUME.
*    X'05' IT COULD NOT BE VERIFIED THAT THIS VOLUME WAS
* MOUNTED NOR WAS A UNIT AVAILABLE FOR MOUNTING THE
* VOLUME.
* X'06' THE OPERATOR WAS UNABLE TO MOUNT THIS VOLUME.
* X'07' THE DATASET WAS NOT DELETED, BECAUSE IT WAS
* CURRENTLY OPEN FOR PROCESSING.
* X'08' THE DATASET IS DEFINED TO RACF, BUT EITHER YOU ARE NOT
* AUTHORIZED TO THE DATASET OR THE DATASET IS A VSAM DATA
* SPACE.

END

JCL TO CALL NOSYSDSN

/STEP02 EXEC PGM=NOSYSDSN
/STELIB DD DISP=SHR,DSN=SYS2.LINKLIB
/SYSPRINT DD SYSOUT=*  
/*
/* DCB MODEL FOR ALLOCATION REQUEST
/*
/MOD1 DD DISP=(,PASS),DSN=&&MOD1,
/ DCB=(LRECL=80,RECFM=FB,BLKSIZ=27920),
/ SPACE=(TRK,(1,1))
/*
/SYSIN DD *
* THE FORMAT OF THE SYSIN DATASET IS IMPOSED:
* ____________________________________________
* LINE 1: (REQUIRED FOR DELETE / RENAME / ALLOC)
* COLUMN 01-06 = DELETE / RENAME / ALLOC (REQUEST TYPE)
* COLUMN 12-55 = DATASET NAME (DSN=......)
* COLUMN 61-66 = VOLSER (WHERE THE DATASET IS LOCATED) (VOL=....)
* COLUMN 68-68 = Y / N (IF THE DATASET CATALOGUED ?)
* LINE 2: (REQUIRED FOR RENAME)
* COLUMN 12-55 = NEW DATASET NAME (DSN=.....)
* LINE 2: (REQUIRED FOR ALLOC)
* COLUMN 14-35 = SPACE ATTRIBUTES (SPACE=...)
* LINE 3: (REQUIRED FOR ALLOC)
* COLUMN 15-22 = DCB MODEL DDNAME (DCBMOD=...)
* 1  2   3   4   5   6   7
* 0   0   0   0   0   0   0
DELETE DSN=TMP50.TEST.OLD        VOL=TMP50 Y
RENAME DSN=TMP50.TEST
        VOL=TMP50 Y
        NEW=TMP50.TEST.OLD
ALLOC  DSN=TMP50.TEST
        VOL=TMP50 Y
        SPACE=(CYL,(0020,0005,0000))
        DCBMOD=MOD1
/*
Utilities for FTP

INTRODUCTION
In recent weeks I have become involved in a number of projects involving e-mailing files and using FTP transfers. The following article describes some of the tools that I have created and which may be of use to others:

- An EDIT macro for converting data to ASCII (this requires a small Assembler program).
- An EDIT macro for converting ASCII data to EBCDIC (this also requires an Assembler program).

EDIT MACRO TO CONVERT TO ASCII

/* REXX */
/* */
/* This edit macro is designed to convert EBCDIC data to ASCII */
/* */
ADDRESS ISEDIT
'MACRO'
'(start) = LINENUM .ZF'
'(endit) = LINENUM .ZL'
DO point=start UNTIL point>=endit
  '(line) = LINE' point
  address linkmvs "C2ASCII line"
  'LINE' point '= (line)'
END
"LOCATE 1"
EXIT 1

SUPPORTING ASSEMBLER ROUTINE C2ASCII
The following routine requires no special linkage but it will need to be available in the TSO STEPLIB concatenation.

******************************************************************
* C2ASCII: CONVERT DATA TO ASCII
******************************************************************
C2ASCII AMODE 31
C2ASCII RMODE ANY

EDIT MACRO TO CONVERT TO EBCDIC

This is very similar to the ASCII routines as can be seen:

/* REXX */
/* */
/* This edit macro is designed to convert ASCII data to EBCDIC */
ADDRESS ISREDIT
'MACRO'
'(start) = LINENUM .ZF'
'(endit) = LINENUM .ZL'
DO point=start UNTIL point=endit
   '(line) = LINE' point
   address linkmvs "C2EBCDIC line"
   'LINE' point '= (line)'
END
"LOCATE 1"
EXIT 1

SUPPORTING ASSEMBLER ROUTINE C2EBCDIC

******************************************************************
* C2EBCDIC: CONVERT DATA TO EBCDIC
******************************************************************
C2EBCDIC AMODE 31
C2EBCDIC RMODE ANY
C2EBCDIC CSECT
BAKR 14,Ø
LR 12,15
USING C2EBCDIC,12
L 1,Ø(1)
LH 5,Ø(1)  * GET THE LENGTH OF THE PARAMETER
LA 4,2(1)
XLATE (4),(5),TO=E
PR
END

Systems Programmer (UK) © Xephon 2001
INTRODUCTION
This article contains the following code:

- A SELCOPY for creating BASE64 encoded data that can be used to convert files into data that can be MIME attached for SMTP transfer.
- A SELCOPY for converting data into HTML and then BASE64 encoding that information for easy e-mailing and browsing on a PC.
- A SELCOPY for decoding BASE64 data into mainframe files.
- An edit macro to convert BASE64 data to EBCDIC.
- A NETRexx program for decoding BASE64 data.

SELCOPY BASE64 ENCODER
Before providing the code, you may be wondering why I have written a BASE64 encoder. Essentially it stems (primarily) from working with mainframe SMTP and e-mail attachments. Frequently when you receive e-mail there are attached files which (when selected) automatically invoke a tool appropriate to the file that has been sent. Although there are a number of methods for achieving this, MIME (Multipart Internet Mail Extensions) is one of the more common and this tends to use BASE64 as its method for sending attachments. BASE64 is a technique of encoding whereby data is processed 3 bytes at a time and each set of 6 bits within the selected 24 is assigned a character from a special BASE64 alphabet. If the data does not contain a multiple of 3 bytes, the data is padded with low values before encoding. The data may also be padded with ‘=”s if this situation occurs to indicate the end of the encoding sequence. The supplied encoder in this article does not worry about the use of ‘=”s because it ensures that the data is always padded out with blanks to a multiple of 3 bytes.
BASE64 data has a maximum LRECL of 76 and this encoder always exploits the maximum for ease of coding. MIME however is not just the encoder. It is also necessary to provide MIME information to define the data incorporated in an e-mail. The following is the base information required to define a MIME e-mail:

- **MIME-Version: 1.0** – this is the start of the MIME e-mail information.
- **Content-Type: text/html; name="a.htm"** – this defines the data for an application to interpret.
- **Content-Transfer-Encoding: Base64** – specifies how the data is encoded.
- **Content-Disposition: attachment; filename="a.htm"** – the filename for the attachment.

If it is intended to incorporate more than one file attachment, or specify text in the e-mail then it is necessary to divide the data into its appropriate parts. To do this the Content-Type should be of the following form:

**Content-Type: multipart/mixed; boundary="=garbage=cribbage="** – where the boundary is a suitably unique collection of characters to separate the data parts. To see how this is used, the following is an example of a multipart e-mail as supplied to a mainframe SMTP job. Note the use of the boundary and how it is identified with the leading ‘—’.

```
MIME-Version: 1.0
Content-Type: multipart/mixed;
  boundary="=garbage=cribbage="
—=garbage=cribbage=
  Content-Type: text/plain
  Content-Transfer-Encoding: quoted-printable
Here we go with a couple of attachments
—=garbage=cribbage=
  Content-Type: text/txt; name="a.txt"
  Content-Transfer-Encoding: Base64
  Content-Disposition: attachment; filename="a.txt"
// DD DSN=base64.file1,DISP=SHR - Base64 encoded data from using the
// SELCOPY below
// DD *
—=garbage=cribbage=
  Content-Type: application/x-rtf; name="a.rtf"
```

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
THE SELCOPY CODE

//your job card
//A EXEC PGM=SELCOPY
//SYSPRINT DD SYSOUT=*  
//INFILE DD DSN=file.to.encode,DISP=SHR
//OFILE DD DSN=encoded.file.name,DISP=(,CATLG),LRECL=76
//SYSIN DD *
READ INFILE W 300000 NORDW - Note if you wish to send VSAM, use the VSAM keyword here *
* IF EOF INFILE
  THEN POS @FLG='Y' * SET TO YES
  THEN GOTO BUILD-RECORD *
* IF INCOUNT GT 1! THEN MOVE 4 FROM UXLRECL TO @RECL * reset L *
* PERFORM PRE-SETS S 1 *
  POS L+1 MOD X'0D25'             * END OF LINE MARKER
  ADD 2 TO 4 AT @RECL TYPE B      * INCREMENT LRECL
  LRECL 4 AT @RECL TYPE B *
  MOVE 1,L TO @END       * PUT DATA INTO CONVERSION WORK AREA *
  ADD 4 AT @RECL TO 4 AT @SIZE TYPE B * READY FOR NEXT RECORD
@END=4 AT @SIZE TYPE B                         * PICK UP WHERE TO PUT DATA
  ADD 4 AT @RECL TO 4 AT @REM TYPE B  * AND INCREMENT TOTAL DATA COUNT *
  IF POS @REM LT X'00000039' * IF THERE IS LESS THAN 1 RECORD THERE
    THEN GG                        * GET MORE DATA *
  BUILD-RECORD *
  IF POS @REM LT X'00000039' * HEADING FOR END OF CONVERSION?
    AND POS @FLG = 'N'
      THEN GG *
  IF POS @REM EQ X'00000000' * ALL DATA DONE?
    AND POS @FLG = 'Y'
      THEN EOJ *
  IF POS @FLG = 'Y'
    AND POS @REM LT X'00000039'
THEN POS @REM MOD X'00000039' " JUST ONE RECORD TO CLEAR
THEN POS @END MOD X'40'
THEN MOVE 200 FROM @END TO @END+1 * PAD
THEN @END=@STRT+200
*
PERFORM BUILD-DATA
*
NOW FILE OFILE FROM @DATA LRECL 76
*
MOVE @STRT+57,@END TO @STRT * SHIFT THE DATA ACROSS
SUB 4 AT @DECR FROM 4 AT @REM TYPE B * DROP THE REMAINDER DOWN
SUB 4 AT @DECR FROM 4 AT @SIZE TYPE B * DROP THE REMAINDER DOWN
@END=4 AT @SIZE TYPE B * PICK UP WHERE TO PUT DATA
GOTO BUILD-RECORD
*
BUILD-DATA
*
* ON INITIAL ENTRY WE NEED TO SET UP OUTPUT RECORD AND INPUT RECORD
* START POINTERS.
*
@BEGN=150000
@OUT=100000 * OUTPUT RECORD BUILT AT POS 100000
*
TRAN @REC AT @BEGN ASCII * CONVERT TO ASCII FIRST
*
* PROCESS 57 CHARACTERS TO CREATE 76 BYTE ENCODED DATA
*
DO DIVIDES-ROUTINE TIMES=19
*
* NOW DO CHARACTER CONVERSIONS
*
TRAN 76 AT 100000 TAB 256 AT 250000 * final base64 encode bit
*
RETURN
*
DIVIDES-ROUTINE
*
MOVE 3 FROM @BEGN TO 213001
POS 213000 MOD X'00'
DIV 4 AT 213000 BY 262144 REM 4 AT 214000 TYPE B
MOVE 1 FROM 213003 TO @OUT
DIV 4 AT 214000 BY 4096 REM 4 AT 215000 TYPE B
MOVE 1 FROM 214003 TO @OUT+1
DIV 4 AT 215000 BY 64 REM 4 AT 216000 TYPE B
MOVE 1 FROM 215003 TO @OUT+2
MOVE 1 FROM 216003 TO @OUT+3
*
* NOW CONTINUE LOOP
*
@BEGN=@BEGN+3
SELCOPY HTML BASE64 GENERATOR

This code is similar to the basic BASE64 encoder, but it has the advantage of creating the following content type.

Content-Type: text/html; name="a.htm" – this allows for automatic invocation of your browser.

```plaintext
//your job card
//A EXEC PGM=SELCOPY
//SYSPRINT DD SYSOUT=*  
//INFILE DD DSN=file.to.encode,DISP=SHR
//OFILE DD DSN=encoded.file,DISP=(,CATLG),LRECL=76  
//SYSIN DD *
READ INFILE W 30000 NORDW *

IF EOF INFILE
    THEN POS 1 MOD '</PRE></BODY></HTML>'
    THEN LRECL 20
    THEN POS @FLG='Y' * SET TO YES *
```
IF INCOUNT GT 1! THEN MOVE 4 FROM UXLRECL TO @RECL * reset L
* PERFORM PRE-SETS S 1
* POS L+1 MOD X'ØD25' * END OF LINE MARKER
ADD 2 TO 4 AT @RECL TYPE B * INCREMENT LRECL
LRECL 4 AT @RECL TYPE B
* MOVE 1,L TO @END * PUT DATA INTO CONVERSION WORK AREA
* ADD 4 AT @RECL TO 4 AT @SIZE TYPE B * READY FOR NEXT RECORD
@END=4 AT @SIZE TYPE B * PICK UP WHERE TO PUT DATA
ADD 4 AT @RECL TO 4 AT @REM TYPE B * AND INCREMENT TOTAL DATA COUNT
* IF POS @REM LT X'ØØØØØØ39' * IF THERE IS LESS THAN 1 RECORD THERE
    THEN GG * GET MORE DATA
* BUILD-RECORD
* IF POS @REM LT X'ØØØØØØ39' * HEADING FOR END OF CONVERSION?
    AND POS @FLG = 'N'
    THEN GG
* IF POS @REM EQ X'ØØØØØØØØ' * ALL DATA DONE?
    AND POS @FLG = 'Y'
    THEN EOJ
* IF POS @FLG = 'Y'
    AND POS @REM LT X'ØØØØØØ39'
    THEN POS @REM MOD X'ØØØØØØ39' * JUST ONE RECORD TO CLEAR
    THEN POS @END MOD X'4Ø'
    THEN MOVE 200 FROM @END TO @END+1 * PAD
    THEN @END=@STRT+200
* PERFORM BUILD-DATA
* NOW FILE OFILE FROM @DATA LRECL 76
* MOVE @STRT+57,@END TO @STRT * SHIFT THE DATA ACROSS
SUB 4 AT @DECR FROM 4 AT @REM TYPE B * DROP THE REMAINDER DOWN
SUB 4 AT @DECR FROM 4 AT @SIZE TYPE B * DROP THE REMAINDER DOWN
@END=4 AT @SIZE TYPE B * PICK UP WHERE TO PUT DATA
GOTO BUILD-RECORD
* BUILD-DATA
* * ON INITIAL ENTRY WE NEED TO SET UP OUTPUT RECORD AND INPUT RECORD
* START POINTERS.
* @BEGN=150000
@OUT=100000 * OUTPUT RECORD BUILT AT POS 100000
* TRAN @REC AT @BEGN ASCII * CONVERT TO ASCII FIRST
* * PROCESS 57 CHARACTERS TO CREATE 76 BYTE ENCODED DATA *
* DO DIVIDES-Routine TIMES=19
* * NOW DO CHARACTER CONVERSIONS *
* TRAN 76 AT 100000 TAB 256 AT 250000 * final base64 encode bit
* RETURN *
* DIVIDES-Routine
*
* MOVE 3 FROM @BEGN TO 213001
* POS 213000 MOD X'00'
* DIV 4 AT 213000 BY 262144 REM 4 AT 214000 TYPE B
* MOVE 1 FROM 213003 TO @OUT
* DIV 4 AT 214000 BY 4096 REM 4 AT 215000 TYPE B
* MOVE 1 FROM 214003 TO @OUT+1
* DIV 4 AT 215000 BY 64 REM 4 AT 216000 TYPE B
* MOVE 1 FROM 215003 TO @OUT+2
* MOVE 1 FROM 216003 TO @OUT+3
* * NOW CONTINUE LOOP *
* @BEGN=@BEGN+3
* @OUT=@OUT+4
* RETURN *
* PRE-SETS *
* POS 290000 MOD '<HTML><HEAD><TITLE>SELCOPY GENERATED</TITLE>
* POS 290044 MOD '<BODY BGCOLOR=LIGHTCYAN TEXT=BLACK><PRE>'
* MOVE 1,L TO 1ØØØØØ * TEMPORARY DATA SHIFT
* MOVE 84 FROM 29ØØØØ TO 1 * OF 84 BYTES TO INSERT HTML CODE
* MOVE L FROM 1ØØØØØ TO 85
* @RECL = 280000 * MY LOCATION FOR RECORD LENGTH MANIPULATION
* MOVE 4 FROM UXLRECL TO @RECL
* ADD 84 TO 4 AT @RECL TYPE B * ADD IN THE HTML BIT
* LRECL 4 AT @RECL TYPE B * AND MODIFY THE L POS
* @SIZE=200000 * WHERE TO PLACE DATA
* @REM=200100 * AMOUNT OF DATA TO CONVERT
* POS @SIZE MOD=X'000249F0' * STORE AT 150000
* POS @REM MOD=X'00000000' * SET TO ZERO
* @DECR=140000 * BASE 64 MAX LRECL
* POS @DECR MOD X'00000039' * WHICH IS 57
* @REC=57 * KEEP VALUE IN CLEAR DECIMAL
* @DATA=100000 * OUTPUT RECORD LOCATION
**SELCOPY BASE64 DECODER**

The following code will decode a BASE64 (as generated by the SELCOPY) file back to its original format. It involves two distinct steps. The first converts the data from LRECL 76 to LRECL 57 EBCDIC data (note that BASE64 is 33% larger than the original data), while the second turns the file back into its original file structure. This particular bit of code can have uses for the store and forward of Visual Basic-type files. Normally when data is stored on a PC and then FTPed back to a mainframe, the original data record lengths are lost. By using a BASE64 encoded file, this problem can be avoided because the second step in this job uses the CR/LF characters to calculate the record length and thereby rebuild the variable length records.

```plaintext
@STRT=150000 * OUTPUT RECORD LOCATION
@FROM=150057 * LEFT SHIFT START POS
@FLG=130000 * END OF FILE FLAG
POS @FLG='N' * SET TO NO
@END=4 AT @SIZE TYPE B * PICK UP WHERE TO PUT DATA
* * BUILD THE BASE64 TRANSLATION TABLE
* POS 250000 MOD='ABCDEFGHIJKLMNOPQRSTUVWXYZ'
POS 250026 MOD='abcdefghijklmnopqrstuvwxyz0123456789+/'
RETURN

SELCOPY BASE64 DECODER

//tour job card
//DECODE EXEC PGM=SELCOPY
//SYSPRINT DD SYSOUT=*  /
//INFILE DD DSN=base64.encoded.file,DISP=SHR
//OFILE  DD DSN=decoded.file,DISP=(,CATLG),LRECL=57
//SYSIN DD *
* DECODE A BASE64 ENTRY
READ INFILE W 300000 NORDW *
* MOVE 100 FROM 290000 TO L+1 * BLANK PAD THE DATA *
PERFORM TRAN-BUILD S 1
TRAN L AT 1 TAB 256 AT 250000 @OUT=200000 * PLACE OUTPUT DATA HERE
POS @OUT MOD X'00'
MOVE 100 FROM @OUT TO @OUT+1 * ZEROISE DATA
@BEGIN=1     * START CONVERSION POINT
DO MULTIPLIES TIMES=19
TRAN 57 AT 200000 EBCDIC  FILE OFILE FROM 200000 LRECL 57
```

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
MULTIPLIES

POS 200 MOD X'000000'
MULT 1 AT @BEGIN BY 262144 INTO 3 AT 200 TYPE B
ADD 3 AT 200 TO 3 AT @OUT TYPE B
MULT 1 AT @BEGIN+1 BY 4096 INTO 3 AT 200 TYPE B
ADD 3 AT 200 TO 3 AT @OUT TYPE B
MULT 1 AT @BEGIN+2 BY 64 INTO 3 AT 200 TYPE B
ADD 3 AT 200 TO 3 AT @OUT TYPE B
ADD 1 AT @BEGIN+3 TO 3 AT @OUT TYPE B
*
* NOW CONTINUE LOOP
*
@BEGIN=@BEGIN+4
@OUT=@OUT+3
RETURN
*
TRAN-BUILD
*
* BUILD THE BASE64 UN-TRANSLATION TABLE
*
pos 250000 mod x'00'
move 255 from 250000 to 250001
POS 250078 MOD=x'3E' * +
POS 250097 MOD=x'3F' * /
pos 250129 MOD=x'a1b1c1d1e1f202122' * a to i
POS 250145 MOD=x'232425262728292a2b' * j to r
POS 250162 MOD=x'2c2d2e2f30313233' * s to z
POS 250193 MOD=x'00102030405060708' * A to I
POS 250209 MOD=x'090a0b0c0d0e0f1011' * J to R
POS 250226 MOD=x'1213141516171819' * S to Z
POS 250240 MOD=x'3435363738393a3b3c3d' * 0 to 9
POS 250258 MOD=x'40' * the pad equals character
RETURN

//REBUILD EXEC PGM=SELCOPY
//SYSPRINT DD SYSOUT=* //INFILE DD DSN=decoded.file,DISP=SHR
//OFILE DD DSN=rebuild.file,DISP=(,CATLG),LRECL=32756,
//     BLKSIZE=32760,SPACE=(CYL,(1,1)) - note that this DCB information
//     may require modifying depending upon the file being re-created.
//SYSIN DD *
READ INFILE W 400000 NORDW
*
IF EOF INFILE
    THENIF @END GT 150000 * CHECK FOR DATA STILL TO OUTPUT
        THEN LRECL=@END-150000
        THEN FILE OFILE FROM 150000
IF EOF INFILE ! THEN EOJ
POS 140000 MOD X'000249F0' S 1 * PREPARE THE END COUNT (POS 150000)
REXX BASE64 DECODER

The following is a simple EDIT macro to convert small BASE64 encoded data back into clear EBCDIC.

```rexx
/* REXX */
/* */
/* This edit macro is designed to convert a base64 file back to EBCDIC display. */
/* */
ADDRESS ISREDIT
'MACRO'
TOVAR='ØØØ1Ø2Ø3Ø4Ø5Ø6Ø7Ø8Ø9ØAØBØCØDØEØF'X
tovar=tovar||'1Ø1112131415161718191A1B1C1D1E1F'X
tovar=tovar||'2Ø2122232425262728292A2B2C2D2E2F'X
tovar=tovar||'3Ø3132333435363738393A3B3C3D3E3FØØ'X
fromvar='ABCDEFGHIJKLMNOPQRSTUVWXYZ'
fromvar=fromvar||'abcdefghijklmnopqrstuvwxyz'
fromvar=fromvar||'Ø123456789+/='
'(start) = LINENUM .ZF'
'(endit) = LINENUM .ZL'
DO point=start UNTIL point>=endit
'(line) = LINE' point
line=TRANSLATE(line,tovar,fromvar)
newline=''
DO cnt=1 TO 76 BY 4
    decoded=Ø
    DO i=1 TO 4
        decoded=decoded||line|i
    END
    newline=newline||decoded
END
```

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
NETREXX BASE64 DECODER

The following program will decode a file that has been BASE64 encoded using the above Selcopy code, and which is located on a PC. To use this, you will need to have downloaded NETRexx and JDK 1.1.8 (at least) from the Web. First compile this code as (say) R64DEC and then you can run this from the MSDOS command prompt with the Java command Java R64DEC. It will then come up asking for the name of the file to be converted into a readable entity. It will create this as the same file name as the original file but with a file extension of R64. The advantage of this is that through the use of the mainframe encoder it is possible to create a file in BASE64 format which can be read on PC and mainframe, and which when stored on a PC does not lose record length information.

```rexx
/* REXX */
/* prepare the translation information */
fromvar='ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyzØ123456789+/

tovar='

loop i=Ø to 63
  a=i.D2C()
  tovar=tovar||a
end

/* */
/* Now ask for the files to process */
/* */
say 'Please specify input file name. An output file will be created with the
  suffix .R64'
splodge=ask
splodgeout=splodge||'.R64'
/* */
/* Open the files */
```
/* */
 outfile=FileWriter(splodgeout)
dest=BufferedWriter(outfile)
 infile=FileReader(splodge)
 source=BufferedReader(infile)

loop label mainloop forever
  text=Rexx source.readline()
  if text = null then leave
  tranchar = text.translate(tovar,fromvar)
  newchar=''
  loop label charloop i=1 to 76 by 4
    var1 = tranchar.substr(i,1)
    var2 = tranchar.substr(i+1,1)
    var3 = tranchar.substr(i+2,1)
    var4 = tranchar.substr(i+3,1)
    var1 = var1.C2D()*262144
    var2 = var2.C2D()*4096
    var3 = var3.C2D()*64
    var4 = var4.C2D()
    numvar = var1+var2+var3+var4
    char1=numvar%65536
    char2=(numvar-(65536*char1))%256
    char3=numvar-(65536*char1)-(256*char2)
    char1=char1.D2C()
    char2=char2.D2C()
    char3=char3.D2C()
    newchar=newchar||char1||char2||char3
  end charloop
  dest.write(newchar,0,newchar.length())
end mainloop
source.close()
dest.close()

---

**E-mail alerts**

If you would like to be notified when new issues of *MVS Update* have been placed on our Web site, you can sign up for our e-mail alert service, which notifies you when new issues (including new free issues) have been placed on our Web site. To sign up, go to [http://www.xephon.com/mvsupdate.html](http://www.xephon.com/mvsupdate.html) and click the ‘Receive an e-mail alert’ link.
From the moment that an OS/390 system is initialized, hardware and software events are continuously recorded in fixed storage buffers, designated ‘system trace tables’. Entries in system trace tables are rarely needed; however, they can be an invaluable aid when used to resolve problems with the operating system itself or with application programs. The inchoate size of a system trace table (STT) is sixty-four kilobytes (sixteen 4K buffers) on each processor. Its size may be enlarged with an operator command, TRACE ST. In order to double the size of the STT from 64K to 128K, one would issue the following OS/390 operator command: TRACE ST,128K.

A functional description of tracing events occurring within an operating system is available in Chapter 8 of the IBM publication: *MVS Diagnosis: Tools and Service Aids*. GTF trace is the principal tool that systems programmers can use to diagnose problems. PPGFISTE is meant to be an adjunct to it, not a replacement. It is so simple to use that anyone can use it to obtain a snapshot of his program as it executes. PPGFISTE makes a single pass through the STTs. Someone could improve upon it by allowing multiple passes through the STTs on a user-specified time interval, and number of passes. Also, it could be modified to format a specific type, or multiple specific types of TTEs.

PPGFISTE provides a hexadecimal dump of an STT entry along with prose that describes most of the fields contained there rather than the Unique-1 and Unique-2 meaningless drivel produced by GTF trace. For selected STT entries, PPGFISTE attempts to provide the name of the module as well as the offset into it where an interrupt occurred. I learned where some of the anchors of system modules were by using an on-line monitor to search for the head of the module chain. While I did not manage to locate the head of all chains, I did locate enough of them to make it worth the time that I spent endeavouring to locate them.

There are a couple of little white lies that will be in any output that is generated by PPGFISTE. One is due to the fact that there is a decided
lack of documentation for a trace table entry (TTE) for a subchannel type major ID. Since empirical evidence seemed to indicate that its contents closely matched that of an entry for a clear subchannel TTE, the same code is used to format both entries. Also, common code is used to format the common section of subchannel TTEs in order to reduce the programming effort used to create PPGFISTE. The result was that a driver-id is produced for a MODIFY Subchannel TTE even though it does not contain one. This is probably one of many fields within a TTE referenced so rarely that there should be minimal confusion caused by the misnomer assigned to that reserved field. And finally, a definite sign that PPGFISTE has gone straightaway into never-never-land is the appearance of BR TTEs in its output. The STT is an extremely volatile area.

PPGFISTE latches onto buffers as they are being modified, so there is always the distinct possibility that one gets altered as PPGFISTE processes it. Normally this condition is followed by a hexadecimal dump of the entire trace table buffer since PPGFISTE may encounter new types of TTEs as OS/390 is enhanced. If this occurs, verify that the last half word in the dump contains zeros. This is where PPGFISTE stows a pointer to the last viable entry residing within it. If the last half word is not zero, then the entry may be a new one and appropriate programming must be done to accommodate it.

There is one more thing to note. An SVCR entry for a GETMAIN TTE may state that it is a FREEMAIN. The documentation that I used stated that the registers of an SVCR TTE contain the same values as were present in the SVC entry. Wrong! Since I used those values to ascertain whether an SVC entry was a GETMAIN or a FREEMAIN, my logic was no longer viable. I did notice, however, that, at least in my shop, the condition code present when there is an SVCR entry for a FREEMAIN is 2 so I used both conditions to decide if an SVCR is a FREEMAIN, or not. It seems to work most of the time.

Many other fields, such as IOQ and IOSB addresses, are also seldom relevant in diagnosing system problems. However, they exist in the TTEs and are formatted so systems programmers will expend less of their valuable time searching for documentation of a TTE’s contents. PPGFISTE creates only a hexadecimal dump for several types of TTE, because they occur so infrequently, if ever, that the programming effort to format them was not worth it. Some examples of those types
of entries are alternate CPU recovery, machine check interruption, and SLIP program event recording.

PPGFISTE has two options. All TTEs may be formatted or only the ones for a specific task whose name is passed to PPGFISTE via the PARM keyword on the EXEC statement used to invoke it. Because TTEs containing PT, PC, PR, and BSG events cannot be associated with a specific task, they are always formatted. There are lots of these types of entries!

Remember that the STT is a highly volatile area so sometimes there are lots of entries for a task being traced, sometimes just a few, and sometimes there are none at all. I could have suspended tracing for each processor as I processed it by modifying control register twelve thereby obtaining somewhat better results, but elected not to do so.

Here is a list of the four-character names that I assigned to TTEs for which I could find no documentation. They differ from their counterparts in length only.

- SRB7 – SRB dispatch
- TSCH – SSCH: start subchannel
- GEXI – general external interrupt
- SSBD – suspended SRB dispatch
- PGMI – program interrupt
- SULS – SUSPlock suspension
- XCXI – external call external interrupt
- CCXI – clock comparator external interrupt
- I-O – I/O interrupt.

PPGFISTE must be authorized and reside in an authorized library. There must be enough virtual memory available to it to contain all STTs. For a four-CPU mainframe with defaults of 16 buffers, 260K is required. With some of the ‘wacky’ region sizes that I have seen of late, that is a mere pittance. The following JCL can be used to invoke PPGFISTE:

```plaintext
//FORMATTE EXEC PGM=PPGFISTE,REGION=1M
//SYSPRINT DD   SYSOUT=*`
```
PPGFISTE is currently used on OS/390 2.6 and has been successfully tested on OS/390 Version 2 Release 9.

PPGFISTE

**Title** 'PPGFISTE - Format Internal System Trace Table Entries'

*------------------------------*
* PPGFISTE is a utility program that formats entries in a          *
* system's internal trace table. It can be used to format         *
* all entries (default), or entries for a single task.           *
* pass the name of a single task to PPGFISTE via the 'parm'       *
* keyword on the exec statement used to invoke PPGFISTE.         *
*------------------------------*

Space 2

PPGFISTE CSECT
PPGFISTE AMODE 31
PPGFISTE RMODE 24

Print Nogen

Bakr R14,R0

Preserve environment at entry

Lr R13,R15

Prime routine's base register

Using PSA,R0

Establish PSA addressability

Using PPGFISTE,R13,R12

Establish routine addressability

Space

La R12,2048(R13)

Prime secondary

La R12,2048(R12)

Base register

Space

L R2,0(R1)

Exec statement parameter

Sr R3,R3

Clear a volatile register

Icm R3,3,0(R2)

Parameter length

Bz PPGNOTNM

Branch if none

Bct R3,0

Decrement by one for execute inst.

Ex R3,PPGMVTNM

Save name of task to be monitored

Space

PPGNOTNM

Ds 0H

Provide target for branch inst.

Space

L R7,CVTPTR

In the beginning...

Using CVT,R7

Establish CVT addressability

L R7,CVTLPDIA

Address of link pack directory

Sr R2,R2

Clear counter

PPGETLPA

Clc 0(40,R7),=40X'00'

Test if at end

Be PPGETSTO

Branch if so

Clc 0(8,R7),=8X'FF'

Test if at end of entries

Be PPGETSTO

Branch if so

Space

La R2,1(R2)

Count entry

La R7,40(R7)

Point to next entry

B PPGETLPA

Loop power! Find end of LPA entries

Space

Drop R7

Forget CVT
MOVE THE LPA ENTRIES INTO STORAGE IN THIS ADDRESS SPACE

ST R2,PPGHOLDC  STOW NUMBER OF LPA MEMBERS
LA R2,1(R2)     OBTAIN AN EXTRA SLOT
MH R2,=H'4Ø'    COMPUTE SIZE OF VIRTUAL AREA REQUIRED
STORAGE OBTAIN,LENGTH=(R2) ACQUIRE IT
LR R3,R1        REPEAT ITS ADDRESS
ST R3,PATH      STOW ITS ADDRESS

SPACE

L R7,CVTPTR     IN THE BEGINNING...
USING CVT,R7    ESTABLISH CVT ADDRESSABILITY
L R7,CVTLPDIA   ADDRESS OF LINK PACK DIRECTORY

SPACE
LR RØ,R3       POINT TO TARGET AREA
LR R1,R2       SET LENGTH OF MOVE FOR TARGET
LR R5,R2       SET LENGTH OF MOVE FOR SOURCE
LR R4,R7       SET SOURCE ADDRESS

SPACE
MVCL RØ,R4     TRANSFER LPA DIRECTORY TO THIS A.S.

SORT LPA ENTRIES BY RELOCATED ENTRY POINT

SPACE
LR R4,R3       FIRST ENTRY
USING LPDE,R4   ESTABLISH LPDE ADDRESSABILITY

SPACE
L R2,PPGHOLDC  PRIME COUNTER
PPGLCLAI L R1,LPDELPNT FETCH ADDRESS OF ENTRY POINT
N R1,PAT7FFF   CLEAR HI-ORDER BIT
ST R1,LPDELPNT STOW ADDRESS OF ENTRY POINT
LA R4,4Ø(R4)   POINT TO NEXT LPDE ENTRY
BCT R2,PPGLCLAI CLEAR HI-ORDER BIT OF NEXT ENTRY PT

SPACE
PPGSORTL L R2,PPGHOLDC PRIME COUNTER
BCTR R2,RØ      REDUCE BY ONE
LR R4,R3       POINT TO BEGINNING OF ENTRIES
LA R5,4Ø(R4)   POINT TO NEXT ENTRY
MVI PPGSW,Ø     RESET SWITCH DONE

SPACE
PPGSORT CLC LPDELPNT,LPDELPNT-LPDE(R5) TEST IF 1ST IS LESS
BE PPGNEXTE    BRANCH IF NOT
BL PPGNEXTE    BRANCH IF NOT
OI PPGSW,1     INDICATE THAT A SWITCH WAS DONE
MVC PPGHOLDL,Ø(R5) HOLD NEXT ENTRY
MVC Ø(4Ø,R5),Ø(R4) PREVIOUS ENTRY TO NEXT ONE
MVC Ø(4Ø,R4),PPGHOLDL NEXT ENTRY TO CURRENT ONE

PPGNEXTE LA R4,4Ø(R4)              POINT TO NEXT ENTRY
LA R5,4Ø(R5)              POINT TO NEXT SUBSEQUENT ENTRY
BCT R2,PPGSORT            PROCESS ALL ENTRIES
SPACE
CLI PPGSW,Ø               TEST IF ALL ENTRIES WERE IN PLACE
BNE PPGSORTL              BRANCH IF NOT
SPACE
DROP R4,R7                FORGET A FEW ADDRESSABILITIES
SPACE 3
OPEN (PATOUT,OUTPUT)      PREPARE DATASET FOR TRANSCRIPTIONS
SPACE
MODESET MODE=SUP,KEY=ZERO WALK ANYWHERE
SPACE
ESAR R1                   GET SECONDARY ASID OF THIS TASK
ST R1,PPGSASID            SAVE IT
EJECT

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* OBTAIN ADDRESS SPACE IDENTIFIER OF THE TASK WHOSE NAME WAS *
* PASSED TO THIS ROUTINE VIA A PARAMETER ON AN EXEC STATEMENT. *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
SPACE 1
CLI PPGTNAME,C' '         TEST IF TASK SELECTED
BE PPGNONAM                BRANCH IF NOT
SPACE 1
LA R3,PPGTNAME            POINT TO TASK'S NAME
BAS R2,PPGFIND            LOCATE IT
STH R5,PPGTASID           STOW IT
ST R6,PPGTASCB            AND STOW ADDRESS OF ITS ASCB
SPACE
MVC PPGWTOI+8+11(8),PPGTNAME STOW TASK'S NAME IN WTO
UNPK PPGWTOI+8+11+8+10(5),PPGTASID(3) STOW TASK'S ASID IN WTO
TR PPGWTOI+8+11+8+10(4),PATRANS-24Ø BEAUTIFY ASID
MVI PPGWTOI+8+11+8+10+4,C' ' REMOVE TRASH
SPACE
PPGWTOI WTO 'JOBNAME IS 12345678 ASID IS 12345'
EJECT

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* LOCATE AND ESTABLISH ADDRESSABILITY TO TRACE'S ADDRESS SPACE *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
SPACE 1
PPGNONAM L R7,CVTPTR       LET THERE BE LIGHT...
USING CVT,R7                ESTABLISH CVT ADDRESSABILITY
SPACE
L R3,PSATRVT               ADDRESS OF SYSTEM TRACE VECTOR TABLE
USING TRVT,R3               ESTABLISH TRVT ADDRESSABILITY
L R2,TRVTTOB               FETCH ADDRESS OF TRACE OPTION BLOCK
DROP R3                    FORGET TRVT
USING TOB,R2               ESTABLISH TOB ADDRESSABILITY
LH R5,TOBASID              FETCH TRACE'S ADDRESS SPACE ID.
SPACE
L R3,CVTCSD                FETCH ADDRESS OF COMM SYS DATA AREA
USING CSD,R3 ESTABLISH CSD ADDRESSABILITY

SPAC E 1
CLC CSDCPUOL,TOBTRPOL ENSURE THAT ALL ACTIVE CPUS R TRACED
BE PPGLSACT BRANCH IF SO

SPACE
WTO 'NOT ALL ACTIVE CPUS ARE BEING TRACED'

SPACE
LA R15,8 SET AN UNSUCCESSFUL RETURN CODE
PR R14 DEPART

SPACE 2
PPGLSACT LAM R11,R11,PPHONE INITIALIZE ACCESS REGISTER
LAM R7,R7,PPHONE INITIALIZE ACCESS REGISTER
LAM R4,R4,PPHONE INITIALIZE ACCESS REGISTER

SPACE 1
LA R1,1 SET AUTHORIZATION
AXSET AX=(R1) INDEX TO ONE
SSAR R5 USE DATA IN TRACE'S ADDRESS SPACE

SPACE 1
SAC 512 SET UNIVERSAL ACCESS MODE

EJECT

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* DETERMINE THE AMOUNT OF VIRTUAL STORAGE THAT IS REQUIRED TO *
* CONTAIN ALL TRACE ENTRIES, THEN ACQUIRE IT. *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SPACE
LH R5,CSDCPUOL FETCH NUMBER OF ACTIVE CPUS
MH R5,TOBTRBUF COMPUTE NO. BUFFERS REQ FOR ALL CPUS
ST R5,PPGBUFNO STOW NUMBER OF BUFFERS ACQUIRED
MH R5,PPG496 COMPUTE SIZE OF AREA REQUIRED
MVC PPGTRBUF,TOBTRBUF SAVE NUMBER OF BUFFERS PER CPU

SPACE
STORAGE OBTAIN,LENGTH=(5) ACQUIRE AN AREA OF THE SAME SIZE

SPACE
LR R10,R1 SAVE ITS ADDRESS

SPACE
SR R9,R9 CLEAR A WORK REGISTER!
ICM R9,12,CSDCPUAL MASK OF ONLINE CPUS TO HI-ORDER BITS
LH R14,CSDCPUOL SAVE NUMBER OF CPUS THAT ARE ONLINE
L R2,TOBPEAD POINT TO START OF PROCESSOR ARRAY

SPACE
DROP R2,R3 FORGET TOB AND CSD
USING TOBPE,R2 ESTABLISH TOBPE ADDRESSABILITY

SPACE
SR R15,R15 CLEAR A WORK REG FOR COUNTING CPUS
PPGDADRS SR R8,R8 CLEAR A WORK REG FOR REG SHIFTING
LH R3,PPGTRBUF # TRACE BUFFERS ON EACH PROCESSOR
SLDL R8,1 SHIFT CPU MASK ONE BIT TO THE LEFT
LTR R8,R8 TEST IF CPU IS ON-LINE
BNE PPGDOCPU BRANCH IF SO
LA R15,1(R15) COUNT CPU
LA R2,TOBPLEN(R2) POINT TO NXT ENTRY IN PROCESSOR ARRAY
LTR  R9,R9   TEST IF ALL CPUS HAVE BEEN PROCESSED
BNE  PPGDADDRS  BRANCH IF NOT
B   PPGDOLCS  ELSE PROCESS TRACE TABLE ENTRIES

SPACE

PPGDCPU ST  R15,PATDOUBL+4  STOW NUMBER OF CPU
UNPK  PATDOUBL(3),PATDOUBL+7(2)
TR  PATDOUBL(2),PATTRANS-240 BEAUTIFY IT
L  R7,TOBPTBVT  POINT TO CURRENT ENTRY IN TRACE TBL?
B   PPGDCAR  ENTER COMMON CODE

SPACE

USING  TBVT,R7  ESTABLISH TBVT ADDRESSABILITY

SPACE

PPGNJT L  R7,TBVTBWRD  ADR OF PREVIOUS TBVT
PPGDCAR CLC  TBVTID,=CL4'TBVT'  ENSURE THE PRESENCE OF A TRACE VECTR
BNE  PPGNTBVT  BRANCH IF NOT
L  R11,TBVTBUFV  ADDRESS OF ASSOCIATED 4K BUFFER
USING  TBUF,R11  ESTABLISH TBUF ADDRESSABILITY

SPACE

CLC  TBUFID,=CL4'TBUF'  ENSURE THE PRESENCE OF A TRACE BUFFR
BNE  PPGNTBUF  BRANCH IF CANNOT

EJECT

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

*        MOVE THE SYSTEM TRACE TABLE TO STORAGE IN THIS ADDRESS SPACE *
*        THEN RECOMPUTE ENTRIES WITHIN IT.                            *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SPACE

LR  R0,R10  POINT TO TARGET AREA
LH  R1,PPG4Ø96  SET LENGTH OF MOVE FOR TARGET
LR  R5,R1  SET LENGTH OF MOVE FOR SOURCE
LR  R4,R11  SET SOURCE ADDRESS
SPACE

MVCL  R0,R4  TRANSFER MTT TO THIS ADDRESS SPACE

SPACE

***********************************************************************
*        COMPUTE OFFSET OF NEXT ENTRY IN CURRENT BUFFER               *
***********************************************************************

SPACE

LRA  R0,Ø(Ø,R11)  REAL ADDRESS OF CURRENT BUFFER ADDR
L  R1,TBVTENTY  REAL ADDRESS OF NEXT ENTRY IN CUR BF
N  R1,PATCR12  REMOVE TRASH FROM REGISTER
SR  R1,R0  COMPUTE NEXT ENTRY'S OFFSET
ST  R1,TBUFID-TBUF(R1Ø)  STOW OFFSET INTO CURRENT BUFFER
MVC  TBUFID-TBUF(2,R1Ø),PATDOUBL STOW PROCESSOR ID. IN BUFFER
SPACE

AH  R1Ø,PPG4Ø96  POINT TO NEXT BUFFER
SPACE

BCT  R3,PPGNJT  PRIME ALL BUFFERS USED BY TRACE
SPACE

LA  R2,TOBPLEN(R2)  POINT TO NEXT ENTRY
LA  R15,1(R15)  NUMBER OF NEXT CPU
DROP  R2  FORGET TOBPE
SPACE
BCT R14,PPGDADRS TRANSFER CONTENTS OF EACH CPUS TBVTS
SPACE
PPGDOLCS
DS 0H
SH R10,PPG4096 POINT TO LAST BUFFER TRANSFERRED
ST R10,PCAR STOW ADDRESS OF CURRENT ENTRY
ST R10,PPGBUFS STOW ADDRESS OF CURRENT BUFFER
ST R10,PPGHOLDA POINT TO OLDEST BUFFER OF LAST CPU
SPACE
L R1,PPGSASID OBTAIN ACTUAL SECONDARY ASID
SSAR R1 SET SECONDARY TO CURRENT
SPACE 1
SAC 0 ACCESS DATA ONLY WITHIN THIS ASID
SPACE 1
SR R1,R1 SET AUTHORIZATION
AXSET AX=(R1) INDEX TO ZERO
SPACE 1
MODESET MODE=PROB,KEY=NZERO BECOME MORTAL ONCE AGAIN
EJECT
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* PROCESS ENTRIES FOUND IN THE TRACE TABLE THAT WAS SAVED. *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
SPACE
PPGLOOP
L R1,PPGBUFS POINT TO CURRENT BUFFER
CLC TBUFID-TBUF(2,R1),PPGNTBVT TEST IF PROCESSING SAME CPU
BE PPGCPVAL BRANCH IF SO
SPACE
MVC PPGNTBVT,TBUFID-TBUF(R1) STOW NUMBER OF CURRENT CPU
MVC PRINTOUT,CLEAR CLEAR BUFFER
MVC PRINTOUT(14),=CL14'PROCESSING CPU' CONSTANT TO OUT AREA
MVC PRINTOUT+15(2),PPGNTBVT CURRENT CPUS TRACE NTRIES FOLLOW
BAS R5,PPGPUTLS PRINT INFORMATION
MVC PRINTOUT,CLEAR CLEAR BUFFER AGAIN
SPACE
PPGCPVAL
L R7,PPGHOLDA POINT TO SYSTEM TRACE ENTRIES
 USING TTE,R7 ESTABLISH TTE ADDRESSABILITY
SPACE
LA R6,PPGBREN POINT TO BRANCH ENTRIES
 USING PPGDSECT,R6 ESTABLISH PPGDSECT ADDRESSABILITY
SPACE
CLC PPGNTBVT,TTETBR24 TEST IF END OF BUFFER
BE PPGDUST IF SO, PROCESS NEXT BUFFER
SPACE 3
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* PROCESS BRANCH ENTRIES *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
SPACE
CLI TTETRTPY,TTETBR24 TEST IF 24-BIT AMODE BR ENTRY
BE PPG4MAT BRANCH IF SO
TM TTETRTPY,TTETBROMD TEST IF 31-BIT AMODE BR ENTRY
BNO PPGNOT31 BRANCH IF NOT
EJECT
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* FORMAT TRACE ENTRIES                                            *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SPACE

PPG4MAT DS ØH PROVIDE TARGET FOR BRANCH

SPACE

SR R5,R5 CLEAR VOLATILE REGISTER

IC R5,PPGLEN LENGTH OF AREA TO BE PRINTED

PPGNANC LR R9,R5 SAVE LENGTH OF AREA PROCESSED

SPACE

BAS R11,PATPRINT PRINT CONTROL BLOCKS

PPGSKIP L R11,PPGHOLDA FETCH POINTER TO CURRENT ENTRY

LA R11,Ø(R9,R11) POINT TO NEXT ENTRY

ST R11,PPGHOLDA REVISE POINTER TO CURRENT ENTRY

SPACE

********************************************************************************
* ENSURE THAT ACCESS TO THE 'CURRENT' TTE DOES NOT OVERFLOW *
* INTO THE NEXT BUFFER.                                       *
********************************************************************************

SPACE

L RØ,PPCURSOR FETCH CURRENT ENTRY'S OFFSET

AR RØ,R9 COMPUTE OFFSET TO NEXT TTE

ST RØ,PPCURSOR REVISE OFFSET TO CURRENT TTE

C RØ,PPCEND TEST FOR OVERFLOW INTO LIMBO

BH PPGDUST BRANCH IF AN OVERFLOW WOULD OCCUR

SPACE 2

L R1,PPGBUFS ADDRESS OF CURRENT BUFFER

LR RØ,R11 ADDRESS OF CURRENT ENTRY

SR RØ,R1 COMPUTE OFFSET OF CURRENT ENTRY

CH RØ,TBUFID-TBUF+2(R1) TEST IF END OF ENTRIES IN THIS BFR

BNE PPGLOOP BRANCH IF NOT

B PPGDUST ELSE TERMINATE SCAN

SPACE 3

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* PROCESS ADDRESS-SPACE-CENTRIC ENTRIES                          *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SPACE

PPGNOT31 SR R9,R9 CLEAR REGISTER USED TO CONTAIN SIZE

LA R2,PPGADRNO NUMBER OF ADDR SPACE-CENTRIC ENTRIES

LA R6,PPGADREN BEGINNING OF THOSE ENTRIES

SPACE

PPGFNDAD CLC TTETYPE,PPGTYPE TEST IF THIS IS THE CORRECT ENTRY

L R15,PPGAPGM ADDRESS OF CODE TO PROCESS TTE TYPE

BE Ø(R15) BRANCH IF SO

LA R6,PPGSIZE(R6) ON TO THE NEXT ENTRY

BCT R2,PPGFNDAD TO ASCERTAIN IF IT IS

EJECT

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* PROCESS EXPLICIT-CENTRIC ENTRIES                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SPACE
TM   TTYTYPE,X'70'          TEST IF VALID EXPCT ENTRY
BNO  PPGUNOWN              BRANCH IF NOT
CLI  TTEXPID,X'7F'         TEST IF ENTRY FOR A USER EVENT
BE   PPGUSR               BRANCH IF SO
SPACE
LA   R2,PPGEXPNO          NUMBER OF EXPCT-CENTRIC ENTRIES
LA   R6,PPGEXPEN          BEGINNING OF THOSE ENTRIES
PPGFNDXP CLC   TTYTYPE,PPGTYPE  TEST IF THIS IS THE CORRECT ENTRY
BE   PPGOTEXP             BRANCH IF SO
SPACE
PPGLOCXP LA   R6,PPGSIZE(R6) ON TO THE NEXT ENTRY
BCT  R2,PPGFNDXP          TO ASCERTAIN IF IT IS
SPACE 2
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* FOR AN UNKNOWN TYPE OF TTE, FORMAT ENTIRE TRACE-BUFFER. *
* IF SUCH A DUMP IS UNDESIRABLE, CHANGE ONE, OR BOTH, NOP *
* INSTRUCTIONS TO AN UNCONDITIONAL BRANCH. *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
SPACE
PPGUNOWN NOP   PPGDUST           ===> ZAP TO UNCONDITIONAL BRANCH <===
SPACE
LA   R6,PPGBREN          POINT TO BRANCH ENTRIES
OI  PPGSW,1              SHOW UNKNOWN TYPE OF TRACE ENTRY
ST   R7,PATDOUBL         STOW CURRENT ADDRESS OF DATA
MVC  PRINTOUT,CLEAR      USE A CLEAN SLATE
SPACE
UNPK  PRINTOUT(9).PATDOUBL(5) BUFFER ADDRESS OF UNKNOWN TTE
MVC  PRINTOUT+8(30),=CL30' IS AN UNKNOWN TYPE OF TTE -'
TR   PRINTOUT(8),PATTRANS-240 MAKE DATA ADDRESS READABLE
SPACE
UNPK  PRINTOUT+8+30(3),TTYTYPE(2) FORMAT UNKNOWN TYPE OF TTE
MVI  PRINTOUT+8+30+2,C' ' ERASE EXTRANEOUS CHARACTER
TR   PRINTOUT+8+30(2),PATTRANS-240 MAKE NEW TTY TYPE READABLE
SPACE
PUT   PATOUT,CLEAR        TRANSCRIBE DATA
SPACE
NOP   PPGDUST             ===> ZAP TO UNCONDITIONAL BRANCH <===
SPACE
L    R7,PCAR              RETRIEVE ADDRESS OF CURRENT ENTRY
LH   R5,PPG496            SET SIZE OF A TBUF
BAS  R11,PATPRINT         FORMAT ALL OF IT
NI   PPGSW,255-1          RESET SWITCH - NO ADDRESSES REQUIRED
B    PPGDUST              PROCESS NEXT BUFFER - IF IT EXISTS
EJECT
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* ENTER EACH EXPCT-ENTRY'S PROCESSING CODE FROM HERE        *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
SPACE
PPGOTEXP CLC   PPGEPL,TTEXPTYP TEST IF CORRECT EXPCT TYPE
BNE  PPGLOCXP           BRANCH IF NOT
SPACE
IC R9,PPGLEN PRIME SIZE  
L R15,PPGAPGM ADDRESS OF CODE TO PROCESS SPECIFIC  
SPACE  
CLI PPGTNAME,C' ' TEST IF TASK NAME IS PRESENT  
BE PPCPSVC BRANCH IF NOT  
CLC PPGTASID,TTEØØ5HA TEST IF CORRECT TASK  
BNE PPGSKIP BRANCH IF NOT  
BR R15 TYPE OF EXPLICIT TRACE ENTRY  
SPACE  
PPCPSVC OI PPGSW,2 DO NOT TRY TO COMPUTE OFFSET IN PGM  
BR R15 ONWARD!  
EJECT  
***********************************************************************  
* FORMAT ENTRIES IN SYSTEM TRACE TABLE                               *  
***********************************************************************  
SPACE 1  
PATPRINT EQU *  
SPACE 1  
MVC PRINTOUT,CLEAR BLANK OUTPUT AREA  
MVC PRINTOUT(4),PPGNAME SHOW ENTRY'S DESIGNATION  
SR R4,R4 CLEAR REMAINDER REGISTER  
LA R3,32 SEGMENT SIZE  
DR R4,R3 COMPUTE NUMBER OF SEGMENTS  
LTR R5,R5 TEST IF QUOTIENT EQUALS ZERO  
BZ PATLSEG BRANCH IF SO  
SPACE 2  
PATSEG LA R8,8 NUMBER OF GROUPS PER SEGMENT  
LA R1Ø,PRINTOUT+1Ø POINT TO OUTPUT AREA  
SPACE 1  
TM PPGSW,1  
BNO PATGROUP  
ST R7,PATDOUBL STOW CURRENT ADDRESS OF DATA  
UNPK PRINTOUT(9),PATDOUBL(5)  
MVI PRINTOUT+8,C' '  
TR PRINTOUT(8),PATTRANS-24Ø MAKE DATA ADDRESS READABLE  
SPACE 1  
PATGROUP MVC PPGSAFE(4),Ø(R7) PREPARE TO TRANSLATE DATA TO EBCDIC  
UNPK Ø(9,R1Ø),PPGSafe(5) PREPARE TO TRANSLATE DATA TO EBCDIC  
MVI B(R1Ø),C' ' BLANK SUPERFLUOUS CHARACTER  
TR Ø(8,R1Ø),PATTRANS-24Ø CONVERT DATA TO EBCDIC  
LA R1Ø,9(R1Ø) NEXT AVAILABLE OUTPUT LOCATION  
LA R7,4(R7) NEXT SOURCE DATA  
BCT R8,PATGROUP COMPLETE A LINE OF DATA  
BAS R14,NEXT MAKE IT PRETTY  
PUT PATOUT,CLEAR TRANSCRIBE DATA  
BCT R5,PATSEG PROCESS ALL SEGMENTS  
SPACE 1  
LTR R4,R4 TEST IF REMAINDER EXISTS  
BE PUTSPACE BRANCH IF NOT  
EJECT  
PATLSEG MVC PRINTOUT,CLEAR BLANK OUTPUT AREA
LA R10,PRINTOUT+10 FIRST AVAILABLE OUTPUT LOCATION
SPACE 1
TM PPGSW,1
BNO PPGNOA1
ST R7,STOW CURRENT ADDRESS OF DATA
UNPK PRINTOUT(9),PATDOUBL(5)
MVI PRINTOUT+8,C' '
TR PRINTOUT(8),PATTRANS-24Ø MAKE DATA ADDRESS READABLE
SPACE 1
PPGNOA1 MVC PRINTOUT(4),PPGNAME SHOW ENTRY'S DESIGNATION
SPACE 1
LR R5,R4 DIVIDEND
SR R4,R4 ZERO REMAINDER
LA R8,4 BYTES PER GROUP
DR R4,R8 COMPUTE NUMBER OF GROUPS
LTR R5,R5 TEST IF QUOTIENT IS ZERO
BE PATLGRUP BRANCH IF SO
SPACE 1
PATPART MVC PPGSAFE(4),Ø(R7) PREPARE TO TRANSLATE DATA TO EBCDIC
UNPK Ø(9,R10),PPGSAFE(5)
MVI Ø(R10),C' ' CLEAR GARBAGE
TR Ø(8,R10),PATTRANS-24Ø MAKE DATA LEGIBLE
LA R10,9(R10) UPDATE RECEIVING FIELD ADDRESS
LA R7,4(R7) UPDATE SOURCE FIELD ADDRESS
BCT R5,PATPART PROCESS ALL COMPLETE GROUPS
SPACE 1
LTR R4,R4 TEST FOR ZERO REMAINDER
BE PATXSCRB BRANCH IF SO
SPACE 1
PATLGRUP LR R5,R4 SAVE REMAINDER
AR R5,R5 DOUBLE FOR RECEIVING FIELD
LR R1,R5 SAVE FOR USE AS AN INDEX
SLL R5,4
OR R5,R4 SET LENGTHS OF RECEIVE AND SEND FLDS
MVC PPGSAFE(4),Ø(R7) PREPARE TO TRANSLATE DATA TO EBCDIC
STC R5,**+L'++1 SET LENGTH IN UNPK INSTRUCTIN
UNPK Ø(Ø,R10),PPGSAFE(Ø) UNPACK DATA
BCTR R1,Ø REDUCE COUNT FOR TR INSTRUCTION
EX R1,PATTPART TRANSLATE REMAINING DATA
LA R10,1(R1,R10) POINT TO NEXT OUTPUT LOCATION
MVI Ø(R10),C' ' CLEAR TRASH
PATXSCRB BAS R14,NEXT SURROUND DATA WITH ASTERISKS
SPACE 1
PUT PATOUT,CLEAR TRANSCRIBE DATE TO SYSPRINT
EJECT
***********************************************************************
* ATTEMPT TO PROVIDE THE LOCATION WITHIN A MODULE WHERE AN         *
* INTERRUPTION HAS OCCURRED FOR SELECTED EXPLICIT ENTRIES.           *
***********************************************************************
SPACE 1
PUTSPACE CLI PPGTNAME,C' ' TEST IF SPECIFIC NAME REQUESTED
BE PPGT4 BRANCH IF NOT

CLI   PPGTYPE,X'70'       TEST IF EXPLICIT ENTRY
BL   PPGT4              BRANCH IF NOT
TM   PPGSW,2            TEST IF COMPUTATION OF OFFSET POSSIBL
BO   PPGT4              BRANCH IF NOT
SPACE
ST   R11,PATDOUBL       STOW RETURN ADDRESS
BAS  R8,PATMLOC         PROVIDE ADDITIONAL INFORMATION
L    R11,PATDOUBL       RETRIEVE RETURN ADDRESS
PPGT4 DS   0H
MVC   PRINTOUT,CLEAR    CLEAR OUTPUT AREA
PUT   PATOUT,CLEAR       SEPARATE RECORDS WITH A BLANK LINE
SPACE 1
NI   PPGSW,255-2        RESET TRASH SWITCH
BR   R11                PROCESS REQUESTED RECORDS
SPACE 1
NEXT  MVI   PRINTOUT+9,C'*'     DELIMIT DATA WITH ASTERISKS
      MVI   PRINTOUT+81,C'*'     FOR NEATNESS
      BR   R14                RETURN
      EJECT

***********************************************************************
* FORMAT INFORMATION REGARDING INTERRUPTION *
***********************************************************************
SPACE 1
PATMLOC  MODESET MODE=SUP,KEY=ZERO WALK ANYWHERE
SPACE
LA    R1,1               SET AUTHORIZATION
AXSET AX=(R1)            INDEX TO ONE
LH    R5,PPGTASID        FETCH ASID OF MONITORED TASK
SSAR  R5                 USE DATA IN TRACE'S ADDRESS SPACE
SPACE 1
SAC   512                SET UNIVERSAL ACCESS MODE
SPACE 2
***********************************************************************
* LOCATE MODULE CONTAINING ADDRESS IN SVC'S OLD PSW *
***********************************************************************
SPACE 1
L    R4,PPGTASCBC       RETRIEVE ADDRESS OF ASCB
USING ASCB,R4           ESTABLISH ASCB ADDRESSABILITY
L    R4,ASCBXTCB        CURRENT TCB ADDRESS
DROP  R4                FORGET ASCB
USING TCB,R4            ESTABLISH TCB ADDRESSABILITY
L    R11,TCBTIO         FETCH ADDRESS OF TASK I/O TABLE
CLC   PPGTNAME,Ø(R11)   TEST FOR THE NAME OF A MATCHING JOB
BNE   ++2                DIE WHEN IN LIMBO
SPACE 1
L    R11,TCBJPQ         ADDR OF LAST CDE IN JOB PACK AREA Q
DROP  R4                FORGET TCB
SPACE 1
MVC   PRINTOUT,CLEAR    CLEAR OUTPUT AREA
LA    R15,CLEAR          POINT TO OUTPUT AREA
USING PATDSECT,R15      ESTABLISH PATDSECT ADDRESSABILITY
MVC   PATINTC(55),=CL55'INTERRUPT AT 12345678 IN PGM 12345678 L
OFFSET 12345678

SPACE 1
MVC PATDPGM,=CL8'UNKNOWN' SET CONSTANT IN PROGRAM NAME AREA
MVC PATDOFF,=CL8'UNKNOWN' SET CONSTANT IN OFFSET AREA
USING CDENTRY,R11 ESTABLISH CDE ADDRESSABILITY
TM CDATTR,CDMIN TEST IF MINOR CDE
BO PGNXTCDE BRANCH IF NOT
SPACE
UNPK PATDPSW(9),PATGILL(5) TRANSFER RIGHT HALF OF PSW TO OUT
TR PATDPSW2,PATTRANS-24Ø MAKE IT PRETTY
MVI PATC1,C' ' REMOVE TRASH
SPACE 1
L R0,PATGILL ADDRESS OF INTERRUPTION
TM PATQILL,X'8Ø' TEST IF 31-BIT AMODE
BO PAT31BIT BRANCH IF SO
N R0,PAT0FF CLEAR HI-ORDER BYTE
PAT31BIT N R0,PAT7FFF CLEAR HI-ORDER BIT
ST R0,PATGILL REVISE IT
EJECT
PATGETXL L R7,CXLMJP ADDR OF EXTENT LIST OF THIS MODULE
USING XTLST,R7 ESTABLISH XTLST ADDRESSABILITY
CLC PATGILL,XTLMSBAD COULD BLOCK CONTAIN FAILING INST?
BL PGNXTCDE BRANCH IF NOT
SPACE 1
L R0,PATGILL ADDRESS OF INTERRUPTION
S R0,XTLMSBAD COMPUTE OFFSET INTO BLOCK
L R1,XTLMSBLA FETCH LENGTH OF MODULE
N R1,PAT7FFF CLEAR HI-ORDER BIT
CR R0,R1 TEST IF OFFSET WITHIN BLOCK
BL PATOK BRANCH IF SO
SPACE 1
PGNXTCDE ICM R11,15,CDCHAIN NEXT CDE ON CHAIN
BE PPGDDODYN AT END TRY ANOTHER AREA
TM CDATTR,CDMIN TEST IF MINOR CDE
BNO PATGETXL BRANCH IF NOT
B PGNXTCDE ELSE TRY TRY AGAIN
SPACE 1
PATOK ST R0,PATGILL SAVE OFFSET
UNPK PATDOFF(9),PATGILL(5)
MVI PATDOFF+8,C' ' CONVERT PACKED DATA TO EBCDIC
PHCDNAME MVC PATDPGM,CDNAME MODULE ACTIVE AT TIME OF ABEND
B PPGOTIT DISPLAY IT
SPACE
PPGLSLPA L R14,PATGILL POINT TO BEGINNING OF ENTRIES
USING LPDE,R14 ESTABLISH LPDE ADDRESSABILITY
L R0,PPGRODPC RETRIEVE NUMBER OF ENTRIES
PPCPLOOP TM LPDEATTR,LPDEMIN TEST IF THIS IS A MINOR LPDE
BO PPCPHLS BRANCH IF SO
SPACE
CLC PATGILL,LPDEMT TEST IF WITHIN RANGE
BE PPCPMVCN BRANCH IF MATCH
BL   PPCPHLS  BRANCH IF LOW
SPACE
L    R1,LPDEXTLN  FETCH LENGTH OF MODULE
A    R1,LPDENTP  ADD ADDRESS OF ENTRY POINT
C    R1,PATGILL  TEST IF THIS MODULE IS WITHIN ADDR
BL   PPCPHLS  BRANCH IF NOT
SPACE
PPCPMVCN
MVC   PATDPGM,LPDENAME  STOW NAME IN OUTPUT AREA
L    R0,PATGILL  FETCH ADDRESS OF INTERRUPT
S    R0,LPDENTP  COMPUTE OFFSET
ST   R0,PATGILL  STOW IT
UNPK  PATDOFF(9),PATGILL(5)  CONVERT TO PACKED DECIMAL
MVI   PATDOFF+8,C' '  REMOVE DETRITUS
TR    PATDOFF,PATTRANS-24Ø  CONVERT PACKED DATA TO EBCDIC
SPACE
B    PPGOTIT  RETURN TO NORMAL SPACE
SPACE
PPCPHLS  LA    R14,4Ø(R14)  POINT TO NEXT ENTRY
BCT   R0,PPCPLOOP  SCAN ALL ENTRIES
B    PPGOTIT  LIES, ALL LIES
EJECT
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*        ATTEMPT TO LOCATE MODULE IN THE DYNAMIC LINK PACK AREA       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
SPACE
PPGDODYN
L    R1,CVTPTR  IN THE BEGINNING...
      USE CVT,R1  ESTABLISH CVT ADDRESSABILITY
L    R1,CVTECVT  FETCH ADDRESS OF EXTENDED CVT
DROP  R1  FORGET CVT
      USE ECVT,R1  ESTABLISH ECVT ADDRESSABILITY
L    R11,ECVTDLPF  FETCH ADDRESS OF FIRST DYNAM LPA CDE
DROP  R1  FORGET ECVT
TM    CDATTR,CDMIN  TEST IF MINOR CDE
BO    PCNXTCDE  BRANCH IF NOT
SPACE 1
L    R0,PATGILL  ADDRESS OF INTERRUPTION
S    R0,XTLMSBAD  COMPUTE OFFSET INTO BLOCK
CLM   R0,7,XTLMSBLN  TEST IF OFFSET WITHIN BLOCK
BL    PATOK  BRANCH IF SO
SPACE 1
PCNXTCDE  ICM   R11,15,CDCHAIN  NEXT CDE ON CHAIN
BE    PPGNLINK  AT END, TRY NEXT QUEUE
TM    CDATTR,CDMIN  TEST IF MINOR CDE
BNO   PCPGETXL  BRANCH IF NOT
B    PCNXTCDE  ELSE TRY TRY AGAIN
SPACE 1
DROP  R14  FORGET LPDE
SEARCH THE LINK PACK AREA FOR A MODULE

SEARCH THE LINK PACK AREA FOR A MODULE

SEARCH THE LINK PACK AREA FOR A MODULE

SEARCH THE LINK PACK AREA FOR A MODULE

SEARCH THE LINK PACK AREA FOR A MODULE

SEARCH THE LINK PACK AREA FOR A MODULE

SEARCH THE LINK PACK AREA FOR A MODULE

SEARCH THE LINK PACK AREA FOR A MODULE

SEARCH THE LINK PACK AREA FOR A MODULE

SEARCH THE LINK PACK AREA FOR A MODULE

SEARCH THE LINK PACK AREA FOR A MODULE

SEARCH THE LINK PACK AREA FOR A MODULE

SEARCH THE LINK PACK AREA FOR A MODULE
LOCATE THE ADDRESS SPACE IDENTIFIER OF A TASK WHOSE NAME IS POINTED TO BY GENERAL PURPOSE REGISTER THREE.

SPACE 1
PPGFIND L R8,CVTPTR IN THE BEGINNING...
USING CVT,R8 ESTABLISH CVT ADDRESSABILITY
L R5,CVTASVT FETCH ADDRESS OF ASVT
DROP R8 FORGET CVT
SPACE 1
USING ASVT,R5 ESTABLISH ASVT ADDRESSABILITY
L R4,ASVTMAXU MAXIMUM NUMBER OF ADDRESS SPACES
SPACE 1
PPGLOC TM ASVTENTRY,ASVTAVAL TEST IF ENTRY IS AVAILABLE
B0 PPGGRUVE BRANCH IF SO
SPACE 1
L R6,ASVTENTRY RETRIEVE ADDRESS OF ASCB
USING ASCB,R6 ESTABLISH ASCB ADDRESSABILITY
SPACE 1
ICM R1,15,ASCBJBNI POINTER TO INITIATED JOBNAME
BZ PPGJBNI BRANCH IF NONEXISTENT
SPACE 1
CLC Ø(8,R1),Ø(R3) TEST IF CORRECT JOB
BNE PPGGRUVE BRANCH IF NOT
B PPGGOTIT ELSE CONTINUE
SPACE 1
PPGJBNI ICM R1,15,ASCBJBNS POINTER TO START/MOUNT/LOGON TASK
BZ PPGGRUVE FORMAT IT
SPACE 1
CLC Ø(8,R1),Ø(R3) TEST IF CORRECT JOB
BE PPGGOTIT BRANCH IF SO
SPACE 1
PPGGRUVE LA R5,4(R5) NEXT ENTRY
BCT R4,PPGLOC LOOP POWER
SPACE
MVC PPGWTO+8+26(8),Ø(R3) SHOW PROBLEM
PPGWTO WTO 'PPGFISTE UNABLE TO LOCATE MSJNLPCM'
LA R15,16 SET CATASTROPHIC ERROR CODE
PR R14 BACK TO DUST
SPACE 1
PPGGOTIT LH R5,ASCBASID OBTAIN ASID OF ADDRESS SPACE
BR R2 RETURN TO CALLER
SPACE
DROP R6,R5 FORGET ASCB AND ASVT
EJECT

***********************************************************************
* PROCESS ALL TTE BUFFERS, THEN TERMINATE.                          *
***********************************************************************

SPACE
PPGDUST L R1,PPGBUFNO RETRIEVE NUMBER OF BUFFERS ACQUIRED
XC PPCURSOR,PPCURSOR INITIALIZE OFFSET TO CURRENT TTE
BCT R1,PPGDUBUF BRANCH IF MORE WORK IS TO BE DONE
SPACE
MVC PRINTOUT,CLEAR CLEAR OUTPUT AREA
MVC PRINTOUT(7),=CL7'CHOW...' SHOW SUCCESSFUL
PUT PATOUT,CLEAR TERMINATION
SPACE
CLOSE (PATOUT) CLEAN IT UP
SPACE 1
SR R15,R15 CLEAR RETURN CODE
PR R14 BACK TO DUST
SPACE 2
***********************************************************************
* AFTER EACH TTE BUFFER IS PROCESSED, INITIALIZE ANCHORS TO  *
* THE NEXT ONE SO THAT IT MAY BE PROCESSED.                      *
***********************************************************************
SPACE
PPGDUBUF ST R1,PPGBUFNO REVISE NUMBER OF BUFFERS
L R8,PPGBUFS FETCH ADDRESS OF CURRENT BUFFER
SH R8,PPG4Ø96 POINT TO NEXT ONE
ST R8,PPGHOLDA STOW ADDRESS OF CURRENT ENTRY
ST R8,PCAR STOW ADDRESS OF CURRENT ENTRY
ST R8,PPGBUFS STOW ADDRESS OF CURRENT BUFFER
B PPGLOOP PROCESS THE NEXT BUFFER
EJECT
***********************************************************************
* FORMAT A PC TRACE TABLE ENTRY                                     *
***********************************************************************
SPACE
USING TTE,R7 ESTABLISH TTE ADDRESSABILITY
SPACE
PPGPC DS ØH
LA R1,PRINTOUT
USING PPGPCD,R1 ESTABLISH PPGPCD ADDRESSABILITY
SPACE
MVC PPCDKEYC,=CL8'PSW KEY' CONSTANT TO OUTPUT AREA
MVC PPCDNUMC,=CL4'PC#' CONSTANT TO OUTPUT AREA
UNPK PPGHOLDL(5),TTEPCPKN(3) PROCESS PSW KEY AS WELL AS PC NO
TR PPGHOLDL(4),PATRANS-24Ø CONVERT VALUE TO EBCDIC
MVC PPCDKEY,PPGHOLDL PSW KEY TO OUTPUT AREA
MVC PPCDNUM,PPGHOLDL+1 PC NUMBER TO OUTPUT AREA
SPACE

Editor's note: this article will be continued in the next edition.

John Gizelar
Systems Programmer (USA)

© Xephon 2001
Landmark Systems has begun shipping its TMON for Unix System Services monitor, which is designed to make it possible to change parameters and abort processes without the need to leave the monitoring console.

It identifies problems, bottlenecks, and availability issues in OS/390 USS resources and enables immediate action to be taken to correct them.

This product, claims the vendor, makes it possible to monitor and tune USS applications without having to know about the Unix environment. It uses the same interface as other Landmark TMON products to allow the viewing of integrated performance data from its suite of OS/390 products running on multiple MVS images across the sysplex.

Users can access CICS, DB2, IMS, MVS, TCP/IP, or VTAM data to monitor performance from one workstation. Specifically, it provides a snapshot of the activity on the current image or sysplex, emphasizing resources being consumed by Unix workloads and identifying potential performance problems.

For further information, contact:
Landmark Corporation, 12700 Sunrise Valley Drive, Reston, Virginia 20191-5804, USA.
Tel: (703) 464 1300
Fax: (703) 464 4918

* * *

IBM has announced that Tivoli NetView Performance Monitor (NPM) for OS/390 now takes advantage of the value-based pricing model.

Tivoli NPM for OS/390 monitors, records, and reports network communication, performance, and utilization, through both Java-based GUI and traditional 3270 SNA displays.

Version 2 Release 6 offers new SNMP alerts with enhanced support for Cisco routers and APPN sessions, enhanced APPN interface statistics, new support for Cisco Internet Protocol, CIP to LU mapping, usability enhancements to the 3270 display panels, recovery of the SNMP router collection on TCP/IP failure, recovery of GUI interface connection on TCP/IP failure, and the ability to launch the Java GUI from the NetView Management Console.

For further information contact:
Tivoli Systems, 9442 Capital of Texas Highway, North Austin, TX 78759, USA.
Tel: 512 436 8000
Fax: 512 794 0623
Tivoli Systems, Sefton Park, Bells Hills, Buckinghamshire, SL2 4HD, UK.
Tel: 01753 896 896
Fax: 01753 896 899
http://www.tivoli.com

* * *