October 2002

In this issue

3  Subsystem to influence the allocation of cartridge drives
11  Customized edit
20  ISPF mail edit macro
25  DFHSM back-up control dataset audit routine
48  Receiving SYSMODs FROMNETWORK with SMP/E Version 3.10
59  Creating a C structure from an Assembler DSECT
68  Listing APF libraries
72  MVS news

© Xephon plc 2002
MVS Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: trevore@xephon.com

North American office
Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

Subscriptions and back-issues
A year's subscription to MVS Update, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1999 issue, are available separately to subscribers for £29.00 ($43.50) each including postage.

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

Editor
Trevor Eddolls

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

Contributions
When Xephon is given copyright, articles published in MVS Update are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please download a copy of our Notes for Contributors from www.xephon.com/nfc.

© Xephon plc 2002. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

Printed in England.
Subsystem to influence the allocation of cartridge drives

When is a 3590 cartridge drive not a 3590 cartridge drive? When it is defined to OS/390 as a 3490 and attached to a Library Storage Module (LSM) from StorageTek (STK). LSMs are lovingly referred to here as silos. Silos mount cartridges in response to requests from OS/390 on behalf of active jobs.

In my shop are two 9310 and eight 4310 Nearline LSMs from StorageTek. Attached to the 9310s are IBM 3590 Magstar cartridge drives while STK 3490 cartridge drives are attached to the 4310s. Based on a recommendation from IBM, both device types were defined to the operating system as 3490s. The address range for the 3590s is 0470 – 049F, and are assigned an esoteric name of STAR. The address range for virtual tapes is 0F00 – 0FFF with an esoteric name of VTAPE. All other cartridge addresses are 3490s and have an esoteric name of CART. STAR cartridges are used to contain dumps of DASD in my shop and datasets that would require five or more CART cartridges. CART and VTAPE cartridges are used to contain the other datasets with which everyone is so familiar – the ones that comprise fewer than one hundred blocks of data.

The range of volume serial numbers that can be mounted on a STAR drive begins at 300,000; any lower-valued serial number is mounted on a CART drive. Volume serial numbers above 500,000 go to VTAPE drives. The Host Software Component (HSC) of the Automated Cartridge System (ACS) ensures, generally, that cartridges are mounted on drives that can read them. Everything worked wonderfully until we conducted a disaster recovery test at an IBM recovery centre. HSC code did not seem to work without silos attached to the mainframe so that OS/390 frequently allocated a cartridge drive that was incompatible with the cartridge that was to have been mounted.

In order to influence proper selection of cartridge drives by OS/390, I wrote a subsystem routine, PPGSSI78, for the tape drive selection call SSI function code 78. I located the documented details of function code 78 in an IBM publication named *OS/390 MVS Using the Subsystem*
Interface, SC28-1502. Access to that manual would facilitate someone’s understanding of the logic within PPGSSI78, since one ‘show me’ is worth a thousand ‘tell mes’.

The logic is basically straightforward. The environment at the time of an SSI 78 call contains, for DD statements that reference a cartridge drive, an eligible device array comprising each cartridge unit created during the system-generation process if, and only if, the dataset specified is to be retrieved via the CATALOG without a UNIT specification. For example, if ten units with an esoteric name of CART are present as well as ten units with an esoteric name of STAR, then, when a dataset is retrieved via the CATALOG, there would be twenty 12-byte entries in each eligible device array. If UNIT=CART were specified, then there would be only ten entries in each Eligible Device Array (EDA). There is one EDA for each volume serial number specified as well as implied. For example, for a specification of VOL=(,,99,SER=111111), OS/390 would create an EDA for 111111 as well as 98 more for the implied serial numbers whose first character is XFE'. You read my mind – a total waste of CPU power and virtual storage. That’s only part of the story. Assume that on that same DD statement, multiple units had been requested as in UNIT=(CART,2). You guessed it! OS/390 creates an additional 99 EDAs. The sad part of this story is that allocation uses only the first EDA to decide which unit to allocate, according to someone from the IBM support line. Sorry, I digressed.

PPGSSI78 ripples down through each EDA comparing the device number in it against the addresses that are valid for allocation to each range of volume serial numbers. The table of STAR addresses is at label PGLMAGS. If a unit in the EDA is incompatible with the volume serial number, then it is marked as ineligible and the next entry is processed. This is done until all entries have been processed.

It’s worth mentioning that all requests for a SCRTCH volume are not filtered. The rationale behind this is simply that for such volumes compatible drives will be paired with compatible cartridges.

For a JOB with the name as label PCPJOBNM, the contents of EDAs are displayed along with their associated volume serial number and DD name. This portion of the logic also allows a cartridge with a volume
serial number in the STAR range to be mounted on a CART device. Only authorized users may perform such a feat. A sample job and output is provided below.

JCL FOR THE JOB

//XEPHON JOB ...
...
//DFSUCMN DD DSN=FH.XEPHON.SAMPLE(Ø),DISP=OLD,UNIT=CART, VOL=(,,99),
// LABEL=(1,SL,,EXPDT=99060)
//AFFDD DD UNIT=(CART,2,DEFER)
//DFSULOG DD DSN=FH.XEPHON.G0001V00,UNIT=AFF=AFFDD,DCB=RECFM=VB,
// VOL=(,1,,SER=(193298)),DISP=OLD

OUTPUT IN THE JOB’S JES2 JOB LOG

DFSUCMN 123758 Ø400 YES 14480000 00000000 63 003C FH.XE...
DFSUCMN 123758 Ø401 YES 14480000 00000000 63 003C FH.XE...

DFSUCMN 123758 Ø4FF YES 14480000 00000000 63 003C FH.XE
DFSUCMN Ø400 YES FE000000001 63 003C FH.XE...

AFFDD Ø4FF YES FE0000000062 63 003C FH.XE...
AFFDD Ø400 YES FF7E0580003B 00 0000 SYS02
AFFDD Ø4FF YES FE0000000063 00 0000 SYS020
DFSULOG 193298 Ø400 YES 14480000 00000000 00 0000 FH.XE...
DFSULOG Ø4FF YES FE0000000064 00 0000 FH.XE...

An initialization routine is required in order for PPGSSI78 to be activated. Its name is PPGINT78. It enables PPGSSI78 to receive and process SSI 78 invocations. It issues the IEFSSVT macro to construct an entry in the SSVT table. It invokes the IEFSSI macro in order to allow an operator to control the status of PPFSSI78 and to enable it to receive and process SSI 78 function requests. These are the operator commands that can be used to control PPGSSI78:

SETSSI ADD,S=CLAM,I=PPGINT78
   ( I is the name of an initialization routine )
SETSSI DEACT,S=CLAM
SETSSI ACT,S=CLAM

Note: S – the name of the subsystem – is limited to four characters.
In my shop, PPGSSI78 is activated at the time OS/390 is IPLed as a
result of the inclusion of the following statement in SYS1.PARMLIB(IEFSSN00):

```
SUBSYS SUBNAME(CLAM) INITRTN(PPGIN78)
```

PPGSSI78 must be linked as a re-entrant routine. Both PPGSSI78 and PPGINT78 must be linked with the linkage editor option AC=1 into an authorized library that is in the concatenation of LNKLST libraries.

**PPGIN78**

```
TITLE 'PPGIN78 - PPGSSI78 INITIALIZATION ROUTINE'
*
* THE PURPOSE OF THIS ROUTINE IS TO ACTIVATE PPGSSI78. IT IS
* INVOKED AT IPL TIME AS A RESULT OF THE INCLUSION OF THE FOLLOWING
* STATEMENT IN SYS1.PARMLIB(IEFSSN00):
*
* SUBSYS SUBNAME(GLEN) INITRTN(PPGIN78)
*
* IT ENABLES THE GLEN SUBSYSTEM TO RECEIVE AND PROCESS FUNCTION
* REQUESTS REGARDING ALLOCATIONS OF CARTRIDGE DRIVES.
*
* ITS INITIALIZATION PROCESS CONSISTS OF THE FOLLOWING STEPS:
* 1. USE THE IEFSSVT MACRO WITH THE CREATE OPTION IN ORDER TO
* CONSTRUCT THE SUBSYSTEM VECTOR TABLE.
* 2. USE THE IEFSSI MACRO WITH THE OPTIONS OPTION IN ORDER TO
* ALLOW AN OPERATOR TO CONTROL THE STATUS OF PPGSSI78.
* 3. USE THE IEFSSI MACRO WITH THE ACTIVATE OPTION IN ORDER TO
* INITIALY ENABLE THE GLEN SUBSYSTEM TO RECEIVE AND PROCESS
* SSI 78 FUNCTION REQUESTS.
*
*
* SPACE 2
MACRO
&TAPNAME TAPINFO
DS 0F
PUSH PRINT
PRINT GEN
&TAPNAME DC CLB'&SYSECT'
DC A(&SYSECT)
DC CL6'&SYSTIME'
DC CLB'&SYSDATE'
POP PRINT
MEND
EJECT
PPGIN78 CSECT
SPACE
PPGIN78 AMODE 31
```
PPGINT78 RMODE 24
SPACE
PRINT NOGEN
SPACE
USING PPGINT78,R12  ESTABLISH PPGINT78 ADDRESSABILITY
SPACE
BAKR R14,R0  PRESERVE ENVIRONMENT AT ENTRY
LR R12,R15  PRIME BASE REGISTER
SPACE
L R10,0(R1)  PRIME SSCVT BASE
USING SSCT,R10  ESTABLISH SSCVT ADDRESSABILITY
SPACE
L R11,4(R1)  PRIME JSIPL BASE
USING JSIPL,R11  ESTABLISH JSIPL ADDRESSABILITY
SPACE
STORAGE OBTAIN,LENGTH=PCPWORKL ACQUIRE VIRTUAL STORAGE
ST R13,4(R1)  POINT TO LOWER SAVE AREA
ST R1,8(R13)  POINT TO HIGHER SAVE AREA
LR R13,R1  SECURE POINTER TO WORK AREA
USING PCPWORK,R13  ESTABLISH PCPWORK ADDRESSABILITY
SPACE
MVC PCPWTO(PCPMSGL),PCPMSG INITIALIZE MESSAGE AREA
EJECT
*******************************************************************************
*                                                                                 *
* BUILD AND INITIALIZE THE SUBSYSTEM'S VECTOR TABLE USING                      *
* THE INPUT TABLE GENERATED BY THE IEFSSVTI MACRO. PPGSSI78                    *
* RESIDES IN SYS1.TECHLINK AND MUST BE LOADED INTO GLOBAL                      *
* STORAGE IN ORDER FOR IT TO BE AVAILABLE FOR USAGE BY ALL                    *
* ADDRESS SPACES.                                                            *
*                                                                                 *
* NOTE: THE NAME USED FOR SSVTDATA ON THE IEFSSVTI MACRO                        *
* STATEMENT MATCH THE ONE USED FOR THE SAME PARAMETER                         *
* ON THE INITIAL IEFSSVTI MACRO STATEMENT.                                     *
*******************************************************************************
SPACE
LA R2,PCPTOKEN  POINT TO TOKEN FOR VECTOR TABLE
SPACE
IEFSSVTI REQUEST=CREATE,SUBNAME=SSCTSNNAM,SSVTDATA=PCPFCODE,LOADTOGLOBAL=YES,MAXENTRIES=PCP#NTRS,OUTTOKEN=(2),RETCODE=PCPRC,RSNCODE=PCPRLN,MF=(E,PCPSSVTI)
SPACE
LR R2,R15  PRESERVE RETURN CODE
L R14,PCPETABL(R15) FETCH ADDRESS OF MESSAGE
MVC PCPRMSG,0(R14) COPY MESSAGE TO OUTPUT AREA
SPACE
UNPK PCPDRC,PCPDR+3(2) CONVERT RETURN CODE TO PACKED DECIMAL
TR PCPDRC(2),PCPTRANS-240 CONVERT REASON CODE TO EBCDIC
MVI PCPDRC+2,C' ' REMOVE DE DETRITUS
UNPK PCPDRSN,PCPRSN+2(3) CONVERT REASON CODE TO PACKED DECMIL
TR PCPDRSN(4),PCPRTRANS-240 CONVERT REASON CODE TO EBCDIC
MVI PCPDRSN+4,C' ' REMOVE DE DETRITUS
SPACE
WTO MF=(E,PCPWTO) SHOW RESULTS OF INIT OPERATION
SPACE
LTR R2,R2 TEST IF SUCCESSFUL OPERATION
BNE PCPEXIT BRANCH IF NOT
EJECT

**************************************************************
* * PERMIT AN OPERATOR TO DYNAMICALLY CONTROL PPGSSI78 VIA A *
* SETSSI OPERATOR COMMAND. *
* *
* SETSSI ADD,S=GLEN,I=PPGINT78 *
* SETSSI DEACT,S=GLEN *
* SETSSI ACT,S=GLEN *
*
**************************************************************
SPACE
IEFSSI REQUEST=OPTIONS,SUBNAME=SSCTSNAM,COMMAND=YES, C
RETCODE=PCPRC,RSNCODE=PCPRSN,MF=(E,PCPSSI)
SPACE
LR R2,R15 PRESERVE RETURN CODE
L R14,PCPETABL(R15) FETCH ADDRESS OF MESSAGE
MVC PCPRM5G,Ø(R14) COPY MESSAGE TO OUTPUT AREA
MVC PCPSERV,=CL8'COMMAND' NAME OF FAILING OPERATION TO WTO
SPACE
UNPK PCPDRC,PCPRC+3(2) CONVERT RETURN CODE TO PACKED DECMIL
TR PCPDRC(2),PCPRTRANS-240 CONVERT REASON CODE TO EBCDIC
MVI PCPDRC+2,C' ' REMOVE DE DETRITUS
UNPK PCPDRSN,PCPRSN+2(3) CONVERT REASON CODE TO PACKED DECMIL
TR PCPDRSN(4),PCPRTRANS-240 CONVERT REASON CODE TO EBCDIC
MVI PCPDRSN+4,C' ' REMOVE DE DETRITUS
SPACE
WTO MF=(E,PCPWTO) SHOW RESULTS OF ACTIVATION OPERATION
SPACE
LTR R2,R2 TEST IF SUCCESSFUL OPERATION
BNE PCPEXIT BRANCH IF NOT
EJECT

**************************************************************
* * ACTIVATE PPGSSI78 *
* *
**************************************************************
SPACE
IEFSSI REQUEST=ACTIVATE,SUBNAME=SSCTSNAM,INTOKEN=PCPTOKE, C
RETCODE=PCPRC,RSNCODE=PCPRSN,MF=(E,PCPSSI)
SPACE
L R14,PCPETABL(R15) FETCH ADDRESS OF MESSAGE
MVC PCPRMSG,Ø(R14)  COPY MESSAGE TO OUTPUT AREA
MVC PCPSERV,=CL8'ACTIVATE' NAME OF FAILING OPERATION TO WTO
SPACE
UNPK PCPDRC,PCPRC+3(2)  CONVERT RETURN CODE TO PACKED DECIMAL
TR PCPDRC(2),PCPTANS-240 CONVERT REASON CODE TO EBCDIC
MVI PCPDRC+2,C' ' REMOVE DETRITUS
UNPK PCPDRSN,PCPRS+2(3) CONVERT REASON CODE TO PACKED DECIMAL
TR PCPDRSN(4),PCPTANS-240 CONVERT REASON CODE TO EBCDIC
MVI PCPDRSN+4,C' ' REMOVE DE DETRITUS
SPACE
WTO MF=(E,PCPWTO) SHOW RESULTS OF ACTIVATION OPERATION
EJECT

PCPEXIT
L R4,PCPSAVE+4 RETRIEVE ADDR OF PREVIOUS SAVE AREA
LR R3,R13 POINT TO AREA TO BE FREED
LR R13,R4 BE NICE: POINT TO PREVIOUS SAVE AREA
LA R0,PCPWORKL SET LENGTH OF WORK AREA
SPACE
STORAGE RELEASE,ADDR=(3),LENGTH=(Ø) RELEASE ACQUIRED STORAGE SPACE
SR R15,R15 SET RETURN CODE EQUAL TO ZERO
PR R14 BACK TO DUST
EJECT

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

*   *   CREATE AN INPUT TABLE FOR A STATIC FUNCTION ROUTINE THAT  *   *
*   CONTAINS THE NAME OF THE FUNCTION ROUTINE AND THE FUNCTION  *   *
*   CODES THAT IT SUPPORTS.                                     *   *
************************************************************

SPACE
IEFSSVTI TYPE=INITIAL,SSVTDATA=PCPFCODE,TABLEN=PCPTSIZE
SPACE
IEFSSVTI TYPE=ENTRY,NUMFCODES=1,FCODES=78,FUNCNAME=PPGSSI78
SPACE
IEFSSVTI TYPE=FINAL
EJECT

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

*   *   DATA RELATED TO FUNCTION ROUTINE                                      *   *
*   *   **********************************************************************

SPACE
PPGSSI78 DC CL8'PPGSSI78'
SPACE
DS ØF
PCPM5G DC AL2(PCPM5GL)
DC H'Ø'
DC CL8'CREATE'
DC C' '
DC C'RETCODE='
DC CL3' '
DC C'RSNCODE='
DC CL5' '
PCPRC0 DC CL26'PROCESSING SUCCESSFUL'
PCPMSGL EQU (*-PCPMSG)
SPACE
PCPETABL DC A(PCPRC0)
DC A(PCPRC4)
DC A(PCPRC8)
DC A(PCPRCC)
DC A(PCPRC10)
DC A(PCPRC14)
DC A(PCPRC18)
SPACE
PCPRC4 DC CL26'WARNING'
PCPRC8 DC CL26'INVALID PARAMETERS'
PCPRCC DC CL26'REQUEST FAILURE'
PCPRC10 DC CL26'ERROR LOADING SUBSYSTEM'
PCPRC14 DC CL26'SYSTEM ERROR'
PCPRC18 DC CL26'SSI SERVICE NOT AVAILABLE'
SPACE
PCP#NTRS DC H'1'
PCPRTRANS DC C'Ø123456789ABCDEF'
SPACE
TAPINFO
SPACE
YREGS
EJECT
*****************************************************************************
**                         **
**   DSECTS                **
**                         **
*****************************************************************************
SPACE
PCPWORK DSECT
PCPSAVE DS 18F
SPACE 2
PCPRC DS F
PCPRSN DS F
PCPTOE DS F
SPACE 2
PCPWTO DC AL2(PCPMSGL)
DC H'Ø'
PCPSERV DC CL8' '
DC C' '
DC C'RETCODE='
PCPDRC DC CL3' '
DC C'RSNCODE='
PCPDRSN DC CL5' '
PCPRMSG DC CL26'PROCESSING SUCCESSFUL'
Customized edit

PROBLEM ADDRESSED

Many dialog applications require the input and updating of data lists. ISPF Dialog Manager offers two application programming interfaces (APIs) that provide editing services:

- **EDIT** – to edit a dataset.
- **EDIF** – to perform editing services. The application must provide the routines to supply the individual records to the edit service (the read routine) and to write the edited records to the appropriate storage medium (the write routine).

In both cases the familiar ISPF edit interface is invoked to edit the individual records. Because the EDIF service isolates the editor from the input/output, editing services can be provided for application-
specific data storage media. Edit macros can be specified for both edit services. Edit macros can be programmed to provide advanced editing functionality. In particular, if the edit commands that cause changes to be made to the stored data (the END and SAVE commands) are caught, logical and formal validations can be made to the edited data before they are saved in persistent storage.

The EDIT service is quite straightforward and does not require any further explanation; it suffices to specify the name of the dataset (and member name, if appropriate) to be edited.

The program described in this article shows how the EDIF service interfaces with the read and write routines. The read routine is initially called once (the ‘first call’). It supplies a data record and sets its return code to determine whether it is to be called again (return code 0) or this is the last input record (‘last call’, return code 8). The write routine is called when an edit command is entered specifying that the current data is to be written (eg SAVE command); it is not invoked as the result of internal editing operations (eg insert a new line). The EDIF service calls the write routine for each line to be written (the passed request code indicates whether this is the first line, the last line, etc). A status flag, indicating how the record was changed (for example inserted or shifted), is also passed; the CUSTEDIT program shown in this article does not use this status flag.

SOLUTION

The CUSTEDIT program described in this article runs in the ISPF environment. For simplicity, the external data is passed as a Dialog Manager variable (this avoids the need to perform file processing operations). The program interfaces with its environment using the Dialog Manager variables:

- DATA – the list of logical records (input and output, mandatory).
- MACRO – the name of the initial edit macro (optional).
- LRECL – the maximum record length (the default is 80).
- DATANAME – the name used as title for the editor (maximum 54 characters; the first blank serves as delimiter – the default is blank).
- **PROFNAME** – the profile name (the default is EDITPROF).
- **PANEL** – the name of the panel used for the display (the default is the standard IBM edit panel).
- **NL** – the new line delimiter that separates each logical record in the list (the default is ;).

**EXAMPLE OF USE**

The application example uses CUSTEDIT to edit a list of dataset names. The invoking REXX procedure (GOEDIT) stores the dataset names as the DATA variable in the current ISPF profile. This allows the dataset name list to be retained between ISPF sessions. The EMDSN edit macro passed to CUSTEDIT reroutes ISPF commands that change data (SAVE and END) to a user-defined command (EMDSNSTO edit macro) that validates the existence of the specified dataset names. EMDSNSTO issues an error message if it detects an error. It exits with the standard, built-in SAVE or END command, as appropriate, only when it does not detect any errors.

**CUSTEDIT PROGRAM CODE**

```plaintext
TITLE 'Customized Data Editor'
PRINT NOENCUSTEDIT CSECTCUSTEDIT
AMODE 31CUSTEDIT RMODE ANY* initialize addressing
STM R14,R12,12(R13)  save registers
BASR R12,0  load base register
USING *,R12  set base register
LA R15,SA  A(save-area)
ST R13,4(R15)  backward ptr
ST R15,8(R13)  forward ptr
LR R13,R15  A(new save-area)
SPACE
* Get options
   CALL ISPLINK,(VCOPY,VNL,VLA,VVA,MOVE),VL
SPACE
* Get input (block)
   MVC VN,=CLB'DATA'
   CALL ISPLINK,(VCOPY,VN,VL,VA,LOCATE),VL
   LTR R15,R15
   JNZ EOJ  exit, <DATA> missing
SPACE
```

* convert record length to binary
  L   R1,VLLRECL
  SH  R1,='H'1'
  EX  R1,EXPACK
  CVB R0,PL8
  ST  R0,LRECL
  SPACE
  MVC  AOREC,ABUF    A(first output record)
  SPACE
  CALL  ISPLINK,(VEDIF,DATANAME,PROFNAME,RECFM,LRECL,READRTN,  X
               AWRITRTN,0,LRECL,0,PANEL,MACRO),VL
  LTR  R15,R15       test whether data written
  JNZ  E0J           no, retain original data
  L   R1,ABUF        A(first output record)
  L   R2,AOREC       A(current output record)
  SR  R2,R1          used length of output buffer
  ST  R2,VL          set length
  CALL  ISPLINK,(VREPLACE,VN,VL,BUF),VL
  SPACE
  E0J  DS  0H

* R15: job return code
  L   R13,4(R13)     restore A(old save-area)
  RETURN (14,12),RC=(15)
  SPACE 1

* symbolic register equates
  R0  EQU  0
  R1  EQU  1
  R2  EQU  2
  R3  EQU  3
  R4  EQU  4
  R5  EQU  5
  R6  EQU  6
  R7  EQU  7
  R8  EQU  8
  R9  EQU  9
  R10 EQU 10
  R11 EQU 11
  R12 EQU 12
  R13 EQU 13
  R14 EQU 14
  R15 EQU 15
  SPACE
  EXPACK  PACK  PL8,LRECL(0)
  TITLE 'Routines'
  READRTN DS  0H          read editor record
  BASR  R15,0
  USING *,R15
  STM  R0,R14,RSA
  BASR  R11,0
USING *,R11
LM R2,R5,R0(R1)
* GPR2: A(record read), output
* GPR3: F'record length', output
* GPR4: F'request code', input
** 0 = read next record
** 1 = read first record
* GPR5: A(dialog data area)

L R4,0(R4)
CHI R4,1
JNE NOTFIRST

MVC AIREC,VA 1st call (initialize start address)

NOTFIRST SR R0,R0
IC R0,NL line delimiter
L R9,VL block length (remaining)
L R6,AIREC string-start
LA R7,0(R9,R6) string-end +1
LHI R15,8 preset ReturnCode (for last record)
SRST R7,R6 search for line delimiter
JO *.4 interrupted, continue
JH READEND delimiter not found

* delimiter found

LA R15,0 reset ReturnCode (normal return)

READEND LR R1,R7 load address of found delimiter
SR R1,R6 - record start addr. = record length
ST R1,0(R3) record length
ST R6,0(R2) record address
LA R7,1(R7)
ST R7,AIREC next record

EOF LM R0,R14,RSA restore registers
BR R14 return
DROPl R11
SPACE

WR1RTN DS 0H write editor record
BASR R15,0
USING *,R15
STM R0,R14,RSA
BASR R11,0
USING *,R11
LM R2,R6,0(R1)

* GPR2: A(record), input
* GPR3: F'record length' (RECFM=V)
* GPR4: XL4'status bits', input
* GPR5: F'request code', input
** 0 = write next record
** 1 = write first record
** 2 = write last record
** 3 = write first+last record
** 4 = write no record
* GPR6: A(dialog data area)
  
  L   R5,0(R5)
  CHI R5,4           no write
  JE  WRITEOK
  SPACE
  L   R0,0(R2)       A(record)
  L   R1,0(R3)       record length
  L   R2,AOREC       A(current record in buffer)
  LA  R15,0(R1,R2)   address of record end
  C   R15,ABUFE      test for buffer overflow
  JH  OVERFLOW
  LR  R3,R1         record length
  MVCL R2,R0        move record to buffer
  MVC O(1,R2),NL     set NL delimiter at EOR
  LA  R2,1(R2)       bump address
  ST  R2,AOREC      store address of next record
  WRITEOK LA R15,0   write OK
  WRITEEND LM R0,R14,RSA restore registers
  BR  R14           return
  SPACE
  OVERFLOW LHI R15,20 buffer overflow
  J   WRITEEND       exit write routine

** TITLE 'Data Areas'

** options name list (the following three blocks must match)

VNL DC A((VNL-E-VNL-2)/8,8) no. of entries, entry length
DC CL8'MACRO'     macro name
DC CL8'NL'        new line delimiter character
DC CL8'LRECL'     maximum record length
DC CL8'DATANAME'  data name
DC CL8'PROFNAME'  profile name
DC CL8'PANEL'     panel name

VNLE EQU *

** field lengths

VLA DC A(L'MACRO,L'NL)
VLLRECL DC A(L'LRECL)
DC A(L'DATANAME)
DC A(L'PROFNAME)
DC A(L'PANEL)

** field data area

VVA DS 0C
MACRO DC CL8' '
NL DC C';'
LRECL DC C'000000'    record length converted to fixed integer
DATANAME DC CL54' '
PROFNAME DC CL8'EDITPROF'
PANEL  DC CL8' '

SPACE

VN DS CL8
VL DS F

© 2002. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
VA DS A
SPACE
VCOPY DC CL8'VCOPY'
VREPLACE DC CL8'VREPLACE'
MOVE DC CL8'MOVE'
LOCATE DC CL8'LOCATE'
SPACE
PL8 DS 0D,PL8 work field
RSA DS '15A register save-area
SPACE
VEDIF DC CL8'EDIF'
RECFM DC C'V'
AREADRNT DC A(READRNT)
AWRITRNT DC A(WRITRNT)
SPACE
SA DS 18F
AIREC DS A A(current input record)
AOREC DS A A(current output record)
SPACE
LTORG
SPACE
ABUF DC A(BUF)
ABUFED DC A(BUFE)
BUF DS CL32760
BUFE EQU *
  TITLE 'Dynamic Linkage Glue Routine'
  * The dynamic linkage glue routine converts a static call
  * into a dynamic call.
  * The called routine is loaded once and then
  * subsequently called with a branch.
ISPLINK CSECT
  BASR R15,0 set base register for glue routine
  USING *,R15
  L R15,AISPLINK
  BR R15
ISPLINK1 BASR R15,0 set temporary base register for loader
  USING *,R2
  USING *,R15
  STM R14,R2,TEPSA save work registers
  DROP R15
  LR R2,R15 set GPR2 as base register
  LOAD EP=ISPLINK load ISPLINK
  ST R0,AISPLINK save entry-point address
  LM R14,R2,TEPSA restore work registers
  J ISPLINK continue
  DROP R2
AISPLINK DC A(ISPLINK1) initial jump address to loading routine
TEPSA DS 5F
END
Note: this example does not use all EDIF capabilities; for example, it processes only RECFM=V.

**GOCEDIT – SAMPLE EXEC TO INVOKE CUSTEDIT**

/* REXX: GOCEDIT */
ADDRESS ISPEXEC
n1 = '!' /* inter-record delimiter */
datename = 'User-Datasets' /* edit title */
macro = "EMDSN" /* edit macro name */
lrecl = 44 /* maximum record length */
"VGET (data) PROFILE" /* get current dataset list from profile */
ADDRESS LINK "custedit"
"VPUT (data) PROFILE" /* save dataset list in profile */

The DATA variable contains a list of dataset names having the form SYS1 LOADLIB!SYS1 LINKLIB!

**EMDSN – EDIT MACRO**

/* REXX: EMDSNEND */
/* reroute SAVE and END commands */
ADDRESS ISREDIT
"MACRO"
"DEFINE SAVE ALIAS EMDSNSTO"
"DEFINE END ALIAS EMDSNSTO"
"MEND"

The (optional) edit macro is invoked before editing starts. It can be used, for example, to add a prologue. In this example, it specifies the aliases for the command to be invoked when the SAVE and END edit commands are entered. Although SAVE and END are the primary edit commands that physically store edited data, other less-frequently used edit commands exist (eg CREATE and REPLACE). These secondary commands could also be caught or disabled, as appropriate.

**EMDSNSTO – CATCH A COMMAND THAT STORES DATA (SAVE AND END COMMAND)**

/* REXX: EMDSNSTO */
/* Edit macro for EMDSN store (SAVE+END) */
ADDRESS ISREDIT
"MACRO"
The EMDSNSTO command (edit macro) is invoked when an edit command that stores data is entered (END or SAVE command). It processes each line of the edit space, which represents a dataset name. The command uses SYSDSN() to determine the status of the dataset. The built-in END or SAVE command is invoked to perform a physical save only when no errors are detected, i.e., all specified datasets actually exist. The ZEDITCMD variable contains the command originally entered.

The ISPF TESTH001 message is displayed with text returned from the SYSDSN function if it returns an error.

TESTH001 MESSAGE
TESTH001 '' .ALARM=YES
'&LMSG'

The TESTH001 message (contained in the TESTH00 member) is displayed when the SYSDSN function returns an error condition (a return value other than OK); the contents of the LMSG variable are displayed.

SUMMARY
This article shows how easy it is to customize the ISPF editor to perform application-specific processing. The example described here uses edit...
macros to perform this application-specific processing. This is more flexible than including such processing in the read routine. The read routine in this application performs only elementary input/output (using VCOPY/VREPLACE Dialog Manager services), however, more input/output processing can be used, for example, to access a DB2 database.

Although the edit macros described here are used with an application program, they can also be used with the standard editor (either directly from ISPF or with the EDIT service). Such edit macros can be used to ensure the observance of installation standards, etc.

System Programmer (Germany) © Xephon 2002

ISPF mail edit macro

INSTALLATION
To install the mail macro follow the steps given below:

1. Send the REXX code to your mainframe (ASCII mode in FTP or ASCII and CR/LF in Personal Communications file transfer).
2. Store it as a library member (RECFM=FB, LRECL=80) named MAIL in a library in your SYSPROC or SYSEXEC concatenation.
3. Configure the SMTP server (short instructions given below).
4. Customize the macro to match your site’s needs/conventions.

CONFIGURATION OF THE SMTP SERVER
The steps to configure the SMTP server (it comes with IBM MVS Communications Server) are outlined below. For a full description please refer to OS/390 eNetwork Communications Server – IP Configuration manual (chapter 2.12).
Configuration steps:

1 Update your PROFILE.TCPIP dataset:
   – Include the name of the member containing the SMTP catalogued procedure in the AUTOLOG statement (this will cause SMTP to come up automatically when TCP/IP is started).
   – Add a PORT statement to ensure that TCP port 25 is reserved for the SMTP server.

2 Customize the SMTP procedure (copy hlq.SEZAINST(SMTPPROC) to your PROCLIB and according to the directions you will find there).

3 Update the PARMLIB IKJTSOxx member so that the TRANSREC statement contains the correct nodename.

4 Customize the SMTP configuration dataset (hlq.SEZAINST(SMTPCONF)). I will not describe all the parameters contained there, let us just have a look at the most important:
   – NJENODENAME – the name of your JES2 node.
   – IPMAILERADDRESS – routes mail sent to an unknown recipient to an SMTP server on an IP network (I have our LAN SMTP server here).
   – MAILFILEDSPREFIX, MAILFILEUNIT, MAILFILEVOLUME – SMTP will use these values to allocate temporary datasets.

5 If you want SMTP to act as a gateway between a TCP network and NJE, include a GATEWAY statement in the configuration dataset and run the TSO command SMTPNJE against a dataset containing JES2 (or JES3) parameters. It will create a dataset named userid.SMTPNJE.HOSTINFO. Include it in the SMTPNJE DD statement of the SMTP procedure.

CUSTOMIZATION OF THE MAIL MACRO
Customization of the MAIL edit macro is pretty simple – just edit it and change a few variables (lines 17–23):

- **hostname** – the name of your host (it will be seen in the header of all mail sent from the MAIL macro; actually you can write anything there!).
- **nodename** – the name of your JES2 node (if you have many nodes, choose the one where the SMTP procedure works).
- **smtpjob** – the name of the SMTP server started task (SMTP by default).
- **subject** – the subject of all messages sent by the MAIL macro (to simplify the usage of the macro; I decided to hard-code the subject).

**Usage**

MAIL is an ISPF edit macro. It must be placed in a library in the SYSEXEC or SYSPROC concatenation. To use it:

1. Edit a dataset you want to mail.
2. Select the text range to be mailed using an ‘S’ line command or ‘SS’..’SS’ block commands. If you do not select anything, the whole dataset will be mailed.
3. Go to the command line and enter: MAIL address <ENTER> where **address** is the e-mail address you want to send your text to.
4. ISPF will show you a confirmation message.

**Attention!**

The MAIL macro will notify you only after successfully transmitting the selected text to the SMTP server. You need to check the SMTP server log to make sure your mail was actually sent! (This is because the macro has no way of validating the user supplied e-mail address. The only thing I could check is the presence of the @ sign) – but I did not bother with that.)

© 2002. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
PROGRAMMING-RELATED INFORMATION
The MAIL macro has been written in REXX. I have tried to make its usage as simple as possible. For that reason I accept only one recipient’s address as a parameter. You can change it, of course, if you like. The parameter processing is located in lines 6 - 8. Remember, however, that the length of an ISPF edit macro parameter is limited.

Also for simplicity reasons I have hard-coded the subject. Its default value is:

datasetname from hostname

where datasetname is the name of the currently edited dataset (got from ISPF) and hostname is the name of the host (supplied by you).

You can change it, for example, to accept the subject as a parameter.

The program logic:
1  Get parameters (actually there is only one) – lines 6–8.
2  Set some variables – lines 10–23.
3  Get the selected lines – lines 27–42.
4  Prepare the mail header – lines 46–53.
5  Prepare the text to be mailed – lines 57–66.
6  Write the mail to a temporary dataset – lines 69–75.
7  Transmit the temporary dataset to the SMTP server – lines 79–80.
8  If successful, inform the user – lines 86–93.

MAILMACRO REXX

/ * REXX **************************** by Marcin Grabinski  */
/ * ISPF edit macro that e-mails selected (SS..SS) text to address given as a parameter */
"MACRO (PARMS) NOPROCESS"
ToAddr = WORD(parms, 1) /* recipient of the message */
ADDRESS ISREDIT
"(DSN) = DATASET"
"(MEM) = MEMBER"
"(LRECL) = LRECL"
/* Change the variables below to match your site's requirments */
hostname = 'spinet.com.pl'
nodename = 'NI1'
smtpjob = 'SMTP'
IF mem = '' THEN
  subject = dsn' from 'hostname
ELSE
  subject = dsn('mem') from 'hostname
/* end of required changes */
'PROCESS RANGE S'
SELECT
  WHEN rc = Ø THEN DO
    (CMD) = RANGE_CMD' /* Get the command */
    (LINE1) = LINENUM.ZFRANGE' /* Get first in range */
    (LINE2) = LINENUM.ZLRANGE' /* Get last in range */
  END
  WHEN rc <= 4 THEN DO /* NO S OR SS ENTERED, USE ENTIRE FILE */
    (CMD) = RANGE_CMD' /* Get the command */
    (LINE1) = LINENUM.ZFIRST' /* Get first in range */
    (LINE2) = LINENUM.ZLAST' /* Get last in range */
  END
OTHERWISE /* Line command conflict - Edit will create message */
  EXIT 12
END /* select */
/* write the text with appropriate SMTP header */
QUEUE "HELLO "hostname
QUEUE "MAIL FROM:<USERID()"@hostname">
QUEUE "RCPT TO:<ToAdr>"
QUEUE "DATA"
QUEUE "Date: "DATE()"
QUEUE "From: "USERID()"@hostname"
QUEUE "To: "ToAdr
QUEUE "Subject: "subject".
QUEUE subject': ' /* beginning of the message body */
ADDRESS ISREDIT
DO i = line1 TO line2
  (LINEVAL) = LINE' i
  QUEUE lineval
END
QUEUE "."
QUEUE "QUIT"
QUEUE '');
/* allocate a temporary dataset */
ADDRESS TSO
cmd = '"ALLOCATE DDN(tempdsn) NEW REUSE LRECl('lrecl') RECFM(F B)"'
INTERPRET cmd
/* write the mail to the temp dataset and send it */
"EXECUT* DISKW tempdsn (OPEN FINIS"

---

© 2002, Xerphon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
DFHSM back-up control dataset audit routine

BCDSINVT is a simple program that has been designed to produce a short audit report of the HSM back-up control dataset. We have attempted to architect the program so that it can be easily modified to produce additional information when desired.

The program has been coded to be re-entrant. While it is doubtful that the need to execute multiple instances of BCDSINVT will ever arise, our preference is to code all programs in this style. All of the datasets are accessed in 31-bit mode. A very simple SYNAD routine is provided for the sequential datasets, as well as a simple error routine for the BCDS itself. The BCDS cluster name or names are supplied in the SYSIN data stream. The name or names are saved, and then we utilize dynamic allocation to create a connection to the dataset. This technique was chosen since it allows us to easily deal with a multi-cluster BCDS. We have attempted to code our programs with more messages, and to make each message as meaningful as possible. These messages would typically be located in the literal section of the program. We made a decision to house the messages in their own module or CSECT. The CSECT does not contain any executable code. It is a very simple table
structure consisting of the messages and an addressing scheme to locate them. We developed a macro, $EDTML$, that can be used to access the message table to obtain the messages. The messages module is called ME$SAGE$, and the intent is that the CSECT is linked with BCDSINVT. We wanted code that was somewhat modular.

Several local macros, $ESAPRO$, $ESAEPi$, and $ESASTG$, which provide general entry, exit, and storage definition are included at the end of the program. The reader can replace these with their own macros if they desire. The $EDTML$ macro is included as well. Our hope is that the reader finds this program useful, and it can serve as the basis for future development.

BCDSINVT

TITLE 'BCDSINVT - GENERATE SIMPLE HSM BCDS AUDIT REPORT'
*--------+-----------------------------------------------+-----------------------------* 
* CSECT : BCDSINVT 
* MODULE : BCDSINVT 
* AUTHOR : ENTERPRISE DATA TECHNOLOGIES 
* DATE : 06-08-2002 
* DESC : BCDSINVT IS A SIMPLE UTILITY WHICH CAN BEUSED TO PRODUCE 
* A SHORT AUDIT REPORT OF THE HSM BACKUP CONTROL DATASET. 
* THE PROGRAM HAS BEEN CODED TO PROCESS MULTI-CLUSTER BCDS. 
* MACROS : $ESAPRO $ESAEPi $ESASTG OPEN CLOSE DCB DCBD DCBE 
* PUT GET STORAGE WTO ACB RPL $EDTML 
* DSECTS : 
* INPUT : BCDS - HSM BACKUP CONTROL DATASET 
* OUTPUT : SYSPRINT - OUTPUT MESSAGES 
* REPORT - OUTPUT FILE CONTAINING THE AUDIT REPORT 
* PLIST : NONE 
* CALLS : $EDTML - PRINT LINE SUPPORT 
* NOTES : 31 BIT ADDRESSING USED FOR ALL FILES. 
* PROGRAM IS REENTRANT 
*--------+-----------------------------------------------+-----------------------------* 
EJECT 
BCDSINVT $ESAPRO R11,R12,AM=31,RM=ANY 
SPACE 1 
*--------+-----------------------------------------------+-----------------------------* 
* GENERAL SETUP PRIOR TO ACTUAL PROGRAM LOGIC COMMENCING. 
*--------+-----------------------------------------------+-----------------------------* 
SPACE 1 
LA R14,PRNT_BFE GET LENGTH OF PRINT BUFFER 
STH R14,PRNT_BFR SAVE IT FOR LATER USE 
LA R14,REPT_BFE GET LENGTH OF PRINT BUFFER 
STH R14,REPT_BFR SAVE IT FOR LATER USE
SPACE 1
*------------------------------------------------------------------*
* DETERMINE HOW MUCH BELOW THE LINE STORAGE FOR DCBS IS NEEDED.  *
*------------------------------------------------------------------*
SPACE 1
L  R0,DCB_MDLL    GET THE QUANTITY NEEDED
ST  R0,BASE_24L   SAVE THE LENGTH
XR  R15,R15       SET SUBPOOL NUMBER TO ZERO
ST  R15,BASE_24P  SAVE AS THE SUBPOOL NUMBER
*------------------------------------------------------------------*
* GO OBTAIN BELOW THE LINE STORAGE TO BUILD THE DATASET DCBS IN.  *
*------------------------------------------------------------------*
SPACE 1
STORAGE OBTAIN,
   LENGTH=(R0),
   SP=(R15),
   LOC=BELOW
SPACE 1
ST  R1,BASE_24    SAVE @(OBTAINED AREA)
*------------------------------------------------------------------*
* PRIME ALL OF THE DYNAMIC DCB AND DCBE AREAS WITH THE MODELS FROM  *
* THE PROGRAM STATIC AREA.  *
*------------------------------------------------------------------*
SPACE 1
LR  R0,R1        PRIME REGISTER ZERO
LA  R14,DCB_MDLS GET @(DCB MODEL SOURCE)
L  R1,DCB_MDLL   GET THE LENGTH TO MOVE
L  R15,DCB_MDLL  GET THE LENGTH TO MOVE
MVCL R0,R14      MOVE MODELS TO UNIQUE STORAGE
*------------------------------------------------------------------*
* ALL OF THE DCB AND DCBE CONTROL BLOCKS ARE LOCATED IN BELOW THE  *
* LINE STORAGE. LOCATE OF THE ADDRESSES AND SAVE THEM.  *
*------------------------------------------------------------------*
SPACE 1
L  R14,BASE_24   GET BASE ADDRESS OF UNIQUE AREA
LA  R1(SYSIN_D1(R14) SYSIN DCB SETUP
ST  R1,@DCB_SYSIN SAVE IT
LA  R1(SYSIN_D2(R14) SYSIN DCBE SETUP
ST  R1,@DCBE_SYSIN SAVE IT
SPACE 1
LA  R1(SYSPRINT_D1(R14) SYSPRINT DCB SETUP
ST  R1,@DCB_SYSPRINT SAVE IT
LA  R1(SYSPRINT_D2(R14) SYSPRINT DCBE SETUP
ST  R1,@DCBE_SYSPRINT SAVE IT
LA  R1(SYSPRINT_D3(R14) SYSPRINT LINE COUNTER
ST  R1,@CNTR_SYSPRINT SAVE IT
MVC $4,R1),=AL4(MAX_PLINE-1) INITIALIZE CNTR. TO MAX
SPACE 1
LA R1,REPORT_D1(R14) REPORT DCB SETUP
ST R1,DCB_REPORT SAVE IT
LA R1,REPORT_D2(R14) REPORT DCBE SETUP
ST R1,DCBE_REPORT SAVE IT
LA R1,REPORT_D3(R14) SYSPRINT LINE COUNTER
ST R1,DCB_REPORT SAVE IT
MVC @(4,R1),=AL4(MAX_PLINE-1) INITILIZE CNTR. TO MAX
SPACE 1
*-------------------------------------------------------------*
* LOCATE ADDRESS OF THE DCBS AND POPULATE INTO THE DCBS.      *
* ALSO POPULATE EXIT LIST INFO INTO THE DCBS.                  *
*-------------------------------------------------------------*
SPACE 1
USING IHADCB,R14 SET ADDRESSABILITY
USING DCBE,R15 SET ADDRESSABILITY
L R14,DCB_SYSIN GET @(SYSIN DCB)
L R15,DCBE_SYSIN GET @(SYSIN DCBE)
STCM R15,'B'1111',DCCBDCBE SAVE @(DCB) IN THE DCB
LA R0,EOF_SYSIN GET @(EOD ROUTINE)
STCM R0,'B'1111',DCBCEEDA SAVE IT IN THE DCB
L R14,DCB_SYSPRINT GET @(SYSPRINT DCB)
L R15,DCBE_SYSPRINT GET @(SYSPRINT DCBE)
STCM R15,'B'1111',DCCBDCBE SAVE @(DCB) IN THE DCB
L R14,DCB_REPORT GET @(REPORT DCB)
L R15,DCBE_REPORT GET @(REPORT DCBE)
STCM R15,'B'1111',DCCBDCBE SAVE @(DCB) IN THE DCB
SPACE 1
*-------------------------------------------------------------*
* PRIME ALL OF THE DYNAMIC OPEN/CLOSE AREAS WITH THE MODELS FROM THE *
* PROGRAM STATIC AREA.                                          *
*-------------------------------------------------------------*
SPACE 1
MVC @(SYSIN@@SYSIN_L),M_OPEN PRIME WITH THE MODEL
MVC @(SYSPRINT@@SYSIN_L),M_OPEN PRIME WITH THE MODEL
MVC @(REPORT@@SYSIN_L),M_OPEN PRIME WITH THE MODEL
MVC @#BCDS@@SYSIN_L),M_OPEN PRIME WITH THE MODEL
MVC @#SYSIN@@SYSIN_L),M_CLOSE PRIME WITH THE MODEL
MVC @#SYSPRINT@@SYSIN_L),M_CLOSE PRIME WITH THE MODEL
MVC @#REPORT@@SYSIN_L),M_CLOSE PRIME WITH THE MODEL
MVC @#BCDS@@SYSIN_L),M_CLOSE PRIME WITH THE MODEL
SPACE 1
*-------------------------------------------------------------*
* WE ARE NOW READY TO BEGIN THE PROCESS OF OPENING THE FILES. WE  *
* WILL START WITH THE SYSPRINT FILE, SINCE IT IS USED FOR MESSAGES.
* IF WE ARE NOT ABLE TO OPEN IT UP, WE WILL ISSUE A WTO, SET A RETURN *
* CODE AND THEN EXIT THE PROGRAM.                             *
*-------------------------------------------------------------*
SPACE 1
L R14,DCB_SYSPRINT GET @(SYSPRINT DCB)
OPEN ((R14),(OUTPUT)), +
  MODE=31, +
  MF=(E,@@SYSPRINT)
L R14,@DCB_SYSPRINT    GET @(DCB WE JUST OPENED)
TM DCOBFLOGS,DCBOFOPN  Q. OPEN CLEAN?
BNO SYN_SYSPRINT         A. NO, GO TO SYNAD ROUTINE
XC FLAG_SYSPRINT,FILE_OPEN INDICATE THE DATASET IS OPEN
SPACE 1
*------------------------------------------------------------------------*
* THE SYSPRINT FILE IS NOW OPEN. WE CAN BEGIN THE PROCESS OF OPENING *
* THE OTHER FILES. TRY TO OPEN THE REPORT FILE. *
*------------------------------------------------------------------------*
SPACE 1
L R14,@DCB_REPORT      GET @(REPORT DCB)
OPEN ((R14),(OUTPUT)), +
  MODE=31, +
  MF=(E,@@REPORT)
L R14,@DCB_REPORT      GET @(DCB WE JUST OPENED)
TM DCOBFLOGS,DCBOFOPN  Q. OPEN CLEAN?
BNO SYN_REPORT          A. NO, GO TO SYNAD ROUTINE
XC FLAG_REPORT,FILE_OPEN INDICATE THE DATASET IS OPEN
SPACE 1
*------------------------------------------------------------------------*
* USE $EDTML AND $EDTPL TO OUTPUT THE APPROPRIATE MESSAGE *
*------------------------------------------------------------------------*
SPACE 1
$EDTML 7,(R8),ME@@AGE@
ICM R10,'B'1111','@DCB_SYSPRINT PICK UP THE @(DCB)
ICM R9,'B'1111','@CNTR_SYSPRINT PICKUP @(COUNTER FIELD)
$CALL @EDTPL,((R8),PRNT_BFR,(R10), +
  (R9),=AL4(MAX_PLINE)), +
  BM=BASR, +
  MF=(E,@@CALL)
SPACE 1
*------------------------------------------------------------------------*
* THE REPORT FILE HAS OPENED, SO WE CAN TRY TO OPEN UP THE SYsin FILE *
*------------------------------------------------------------------------*
SPACE 1
L R14,@DCB_SYsin       GET @(REPORT DCB)
OPEN ((R14),(INPUT)), +
  MODE=31, +
  MF=(E,@@SYsin)
L R14,@DCB_SYsin       GET @(DCB WE JUST OPENED)
TM DCOBFLOGS,DCBOFOPN  Q. OPEN CLEAN?
BNO SYN_SYsin           A. NO, GO TO SYNAD ROUTINE
XC FLAG_SYsin,FILE_OPEN INDICATE THE DATASET IS OPEN
SPACE 1
*------------------------------------------------------------------------*
* USE $EDTML AND $EDTPL TO OUTPUT THE APPROPRIATE MESSAGE *
*------------------------------------------------------------------------*

SPACE 1
$EDTML 1,(R8),ME@AGE@
ICM R10,B'1111',@DCB_SYSPRNPT PICK UP THE @(DCB)
ICM R9,B'1111',@CNTN_SYSPRNPT PICKUP @(COUNTER FIELD)
$CALL @EDTPL,((R8),PRNT_BFR,(R10),
   + (R9),=AL4(MAX_PLINE)),
   + BM=BASR,
   + MF=(E,@@CALL)
SPACE 1
*-------------------------------------------------*  
* ALL FILES ARE NOW OPENED. WE CAN PROCESS INPUT FROM THE SYsin FILE *
* AND SAVE THE DATASET NAMES IN A TABLE. *
*-------------------------------------------------*
SPACE 1
MVI TRAN_TAB+X'40',X'40'   SET DELIMITER IN PLACE
LA R7,BCDS_DSS GET @(TABLE ENTRY)
STCM R7,B'1111',BCDS_DSN SAVE THE ADDRESS
LOP_SYSiN DS @H
  L R14,@DCB_SYSPRNPT GET @(SYsin DCB)
GET (R14)
LR R3,R1 POINT TO THE RECORD WE READ
TRT Ø(B0,R3),TRAN_TAB FIND THE FIRST SPACE
BC Ø,ERR_SYSPRNPT ERROR, SHOULD NOT OCCUR
LR R4,R1 POINT TO THE SPACE
SR R4,R3 COMPUTE THE LENGTH
STH R4,Ø(R7) SAVE THE LENGTH
BCTR R4,Ø DECREMENT LENGTH BY 1
EX R4,MVC_DSN SAVE THE ADDRESS
STCM R7,B'1111',BCDS_DSP INC ORMENT POINTER ADDRESS
LA R7,46(,R7) GO GET ANOTHER RECORD
ERR_SYSPRNPT DS @H
SPACE 1
*-------------------------------------------------*  
* USE $EDTML AND $EDTPL TO OUTPUT THE APPROPRIATE MESSAGE *
*-------------------------------------------------*
SPACE 1
$EDTML 10,(R8),ME@AGE@
ICM R10,B'11111',@DCB_SYSPRNPT PICK UP THE @(DCB)
ICM R9,B'11111',@CNTN_SYSPRNPT PICKUP @(COUNTER FIELD)
$CALL @EDTPL,((R8),PRNT_BFR,(R10),
   + (R9),=AL4(MAX_PLINE)),
   + BM=BASR,
   + MF=(E,@@CALL)
SPACE 1
B LOP_SYSPRNPT GO GET ANOTHER RECORD
EOF_SYSPRNPT DS @H
SPACE 1
*-------------------------------------------------*  
* WE COME HERE WHEN WE HAVE PROCESSED ALL OF THE INPUT DATA FROM THE *
* SYSIN DATASET.

SPACE 1
LA R0,AUDT_TBL POINT TO BEGINNING OF TABLE
LA R1,AUDT_LEN GET THE LENGTH OF THE TABLE
LA R14,AUDT_BGN POINT TO THE MODEL TABLE
LA R15,AUDT_LEN GET THE LENGTH
MVCL R0,R14 MOVE MODEL TO DYNAMIC TABLE
LA R1,TBL_TBL POINT TO BEGINNING OF DYN. TABLE
LA R14,AUDT_ONE-AUDT_BGN(R1) POINT TO FIRST ENTRY
ST R14,0(R1) SAVE IT
LA R14,AUDT_ENL GET THE LENGTH OF AN ENTRY
ST R14,4(R1) SAVE IT
LA R14,AUDT_LLL(R1) POINT TO LAST ENTRY
ST R14,8(R1) SAVE IT
L R7,BCDS_DSN POINT TO FIRST TABLE ENTRY
LA R8,BCDS_EZ LENGTH OF TABLE ENTRY
L R9,BCDS_DSP POINT TO LAST TABLE ENTRY

SPACE 1

* NOW WE WILL START TO BUILD THE DYNAMIC ALLOCATE ( SVC 99 ) TEXT *
* UNITS THAT WILL BE USED TO ALLOCATE THE BCDS. *

SPACE 1

LOP_DYNA DS 0H
STM R7,R9,SAVER7R9 SAVE, SO WE CAN RELOAD LATER
MVC TU99_000,=AL2(DALDDNAM) DDNAME
MVC TU99_000+2,=X'0001' KEY VALUE
MVC TU99_000+4,=X'0004' LENGTH OF THE DDNAME
MVC TU99_000+6,=CL4'BCDS' ACTUAL DDNAME
MVC TU99_004,=AL2(DALSTATS) DISP=
MVC TU99_004+2,=X'0001' KEY VALUE
MVC TU99_004+4,=X'0001' LENGTH
MVC TU99_004+6,=X'08' REQUEST SHR
MVC TU99_008,=AL2(DALDSNAM) DSNAME
MVC TU99_008+2,=X'0001' KEY VALUE
MVC TU99_008+4,=X'0001' LENGTH OF THE DSNAME
LA R3,TU99_008+6 @(DSNAME IN TEXT UNIT)
LH R4,0(R7) GET THE LENGTH OF THE DSNAME
BCTR R4,0 DECREMENT IT BY 1
EX R4,MVC_DYN EX. MOVE TO PLACE IN TEXT UNIT

SPACE 1

* NOW WE WILL CHAIN UP THE TEXT UNITS TO THE TEXT POINTERS. *

SPACE 1

LA R1,TU99_000 GET @(TEXT UNIT)
SCCM R1,B'1111',TP99_000 SAVE IT
LA R1,TU99_004 GET @(TEXT UNIT)
SCCM R1,B'1111',TP99_004 SAVE IT
LA     R1,TU99_008    GET @(TEXT UNIT)
STCM   R1,B'1111',TP99_008  SAVE IT
OI     TP99_008,X'80'     TURN ON THE HIGH ORDER BIT
LA     R1,TP99_000     GET @(FIRST TEXT PTR. ADDR.)
STCM   R1,B'1111',RB99_008  SAVE IT IN THE REQUEST BLOCK
MVI    RB99_000,X'14'    LENGTH OF THE REQUEST BLOCK
MVI    RB99_001,S99VRBAL  INDICATE WE WANT TO ALLOCATE
LA     R1,RB99_000    GET THE @(REQUEST BLOCK)
STCM   R1,B'1111',SVC_99RB  SAVE IT IN PLIST
OI     SVC_99RB,X'80'    HIGH ORDER BIT ON
LA     R1,SVC_99RB    POINT R1 AT THE PLIST
SVC    99     ATTEMPT THE ALLOCATE
LTR    R15,R15    Q. ALLOCATE CLEAN?
BNZ    ALLOC_ER    A. NO, GO ISSUE MESSAGE
SPACE 1
*-----------------------+----------------------------------------------*
* USE $EDTML AND $EDTPL TO OUTPUT THE APPROPRIATE MESSAGE *
*-----------------------+----------------------------------------------*

SPACE 1
$EDTML 12,(R8),ME@AGE@
ICM    R10,B'1111',@DCB_SYSPRINT PICK UP THE @(DCB)
ICM    R9,B'1111',@CNTR_SYSPRINT PICKUP @(COUNTER FIELD)
$CALL  @EDTPL,((R8),PRNT_BFR,(R10),
       (R9),=AL4(MAX_PLINE)),
       BM=BASR,
       MF=(E,@@CALL)
SPACE 1
B     ALLOC_OK
ALLOC_ER DS @H

SPACE 1
*-----------------------+----------------------------------------------*
* USE $EDTML AND $EDTPL TO OUTPUT THE APPROPRIATE MESSAGE *
*-----------------------+----------------------------------------------*
SPACE 1
$EDTML 13,(R8),ME@AGE@
ICM    R10,B'1111',@DCB_SYSPRINT PICK UP THE @(DCB)
ICM    R9,B'1111',@CNTR_SYSPRINT PICKUP @(COUNTER FIELD)
$CALL  @EDTPL,((R8),PRNT_BFR,(R10),
       (R9),=AL4(MAX_PLINE)),
       BM=BASR,
       MF=(E,@@CALL)
SPACE 1
B     CLOSE_FILES
ALLOC_OK DS @H

SPACE 1
*-----------------------+----------------------------------------------*
* PRIME THE ACB AND RPL STRUCTURES WITH THE MODEL INFORMATION *
*-----------------------+----------------------------------------------*
SPACE 1
MVC  BCDS ACB(ACB_MOLL),ACB_MODL PRIME THE ACB
MVC  BCDS_RPL(RPL_MOLL),RPL_MODL  PRIME THE RPL
MVC  @MODCB(MOD_MOLL),MOD_MODL  PRIME THE MODCB
MVC  @SHOWCB(SHO_MOLL),SHO_MODL  PRIME THE SHOWCB
SPACE 1
LA  R3,BCDS_RPL         GET @(RPL)
LA  R4,BCDS_ACB         GET @(ACB)
LA  R5,R_BUFF           GET @(ADDRESS OF DATA BUFFER)
SPACE 1
*--------------------------------------------------------------------------*
* MOVE DYNAMIC INFORMATION INTO THE RPL FOR THE BCDS                      *
*--------------------------------------------------------------------------*
SPACE 1
MODCB  RPL=(R3),
       ACB=(R4),
       AREA=(R5),
       AREALEN=4,
       MF=(E,@MODCB)
SPACE 1
*--------------------------------------------------------------------------*
* PICK UP THE ADDRESS OF THE ACB AND OPEN IT UP                           *
*--------------------------------------------------------------------------*
SPACE 1
LA  R5,BCDS_ACB         PRIME REGISTER 5
OPEN  ((R5)),MODE=31
LTR  R15,R15             Q. GOOD OPEN ?
BZ  BCDS_OPN             A. YES, PROCEED
LA  R5,BCDS_ACB         GET @(ACB)
LA  R6,ACB_INFO          GET @(INFO FIELD)
SPACE 1
*--------------------------------------------------------------------------*
* WE ARE COMING HERE ONLY IF WE HAD AN ERROR OPENING THE BCDS              *
*--------------------------------------------------------------------------*
SPACE 1
SHOWCB  ACB=(R5),
       AREA=(R6),
       LENGTH=4,
       OBJECT=DATA,
       MF=(E,@SHOWCB)
$EDTML 16,(R8),ME@AGE@
ICM  R10,B'1111',@DCB_SYSPRINT PICK UP THE @(DCB)
ICM  R9,B'1111',@CNTR_SYSPRINT PICK UP @(COUNTER FIELD)
$CALL @EDTPL,((R8),PRNT_BFR,(R10),
       (R9),=AL4(MAX_PLINE)),
       BM=BASR,
       MF=(E,@@CALL)
B  CLOSE_FILES
SPACE 1
BCDS_OPN  DS  @H
SPACE 1
MVC  FLAG_BCDS,FILE_OPEN

SPACE 1
*--------------------------------------------------------*  
* USE $EDTML AND $EDTPL TO OUTPUT THE APPROPRIATE MESSAGE  *
*--------------------------------------------------------*

SPACE 1
$EDTML 11,(R8),ME@AGE@
ICM R10,B'1111',@DCB_SYSPRINT PICK UP THE @DCB  
ICM R9,B'1111',@CNTR_SYSPRINT PICKUP @COUNTER FIELD  
$CALL $EDTPL((R8),PRNT_BFR,(R10)),  
(R9),=AL4(MAX_PLINE)),  
+ BM=BASR,  
+ MF=(E,@CALL)

SPACE 1
*--------------------------------------------------------*  
* MAIN LOOP TO READ AND PROCESS THE BCDS ENTRIES.         *
*--------------------------------------------------------*

SPACE 1
LOOP_BCD DS @H  
LA R6,BCDS_RPL GET @(RPL)  
GET RPL=(R6)  
LTR R15,R15 Q. READ SUCCESSFUL?  
BZ GOOD_BCD A. YES, DETERMINE RECORD TYPE  
C R15,R00008 Q. RETURN CODE 8?  
BNE CLOSE_FILES A. NO, EXIT FOR NOW  
CLI 15(R6),RPLDVOQ EOD OF FILE?  
BE CLOS_BCD A. YES, GO CLOSE BCDS  
BNZ CLOSE_FILES A. NO, EXIT
GOOD_BCD DS @H  
L R2,R_BUFF GET @(CURRENT RECORD)  
LM R3,R5,@UDT_TBL PICK UP VALUES FOR BXLE LOOP
CHEK_BCD DS @H  
CLC 0(1,R2),4(R3) Q. MATCH TO A TABLE ENTRY  
BNE CHEK_NXT A. NO, CHECK NEXT ENTRY  
CLI 0(R2),X'30' Q. IS IT A TYPE 30?  
BNE CHEK_N30 A. NOT A 30  
CLC 0(4,R2),4(R3) Q. WHICH TYPE OF 30?  
BNE CHEK_NXT A. NOT THIS ONE
CHEK_N30 DS @H  
ICM R14,B'1111',0(R3) GET THE COUNTER  
LA R14,1(R14) BUMP IT UP BY 1  
STCM R14,B'1111',0(R3) NOW SAVE IT  
B CHEK_DON GO GET THE NEXT RECORD
CHEK_NXT DS @H  
BXLE R3,R4,CHEK_BCD BXLE THROUGH THE TABLE  
ICM R14,B'1111',0(R3) GET THE COUNTER  
LA R14,1(R14) BUMP IT UP BY 1  
STCM R14,B'1111',0(R3) NOW SAVE IT
CHEK_DON DS @H  
B LOOP_BCD GET ANOTHER RECORD
SPACE 1
*CLOSE THE CURRENT BCDS, ISSUE A MESSAGE AND THEN DO THE UNALLOCATE *
* VIA SVC 99 SERVICES.

SPACE 1
CLOS_BCD DS 0H
LA R5,BCDS_ACB PRIME REGISTER 5
CLOSE ((R5)),MODE=31
LTR R15,R15 Q. GOOD OPEN ?
BNZ CLOSE_FILES A. NO, GO CLOSE OTHER FILES
SPACE 1
XC FLAG_BCD,FLAG_BCDS INDICATE BCDS CLOSED
SPACE 1

* USE $EDTML AND $EDTPL TO OUTPUT THE APPROPRIATE MESSAGE *

SPACE 1
$EDTML 04,(R8),ME@AGE@
ICM R10,B'1111',@DCB_SYSPRINT PICK UP THE @(DCB)
ICM R9,B'1111',@CNTR_SYSPRINT PICKUP @(COUNTER FIELD)
$CALL @EDTPL,(B8),PRNT_BFR,(R10), +
    (R9),=AL4(MAX_PLINE)), +
    BM=BASR, +
    MF=(E,@@CALL)
SPACE 1

* NOW WE WILL CHAIN UP THE TEXT UNITS TO THE TEXT POINTERS. *

SPACE 1
LA R1,TU99_000 GET @(TEXT UNIT)
STCM R1,B'1111',TP99_000 SAVE IT
LA R1,TU99_000 GET @(TEXT UNIT)
STCM R1,B'1111',TP99_004 SAVE IT
OI TP99_004,X'80' TURN ON THE HIGH ORDER BIT
LA R1,TP99_000 GET @(FIRST PTR. ADR.)
STCM R1,B'1111',RB99_008 SAVE IT IN THE REQUEST BLOCK
MVI RB99_000,X'14' LENGTH OF THE REQUEST BLOCK
MVI RB99_001,S99VRBUN INDICATE WE WANT TO ALLOCATE
LA R1,RB99_000 GET THE @(REQUEST BLOCK)
STCM R1,B'1111',SVC_99RB SAVE IT IN PLIST
OI SVC_99RB,X'80' HIGH ORDER BIT ON
LA R1,SVC_99RB POINT R1 AT THE PLIST
SVC 99 ATTEMPT THE ALLOCATE
LTR R15,R15 Q. UNALLOCATE COMPLETE?
BNZ UNAL_BAD A. NO, ERROR
SPACE 1

* USE $EDTML AND $EDTPL TO OUTPUT THE APPROPRIATE MESSAGE *

SPACE 1
$EDTML 14,(R8),ME@AGE@
ICM R10,B'1111',@DCB_SYSPRINT PICK UP THE @(DCB)
ICM R9,B'1111',@CNTR_SYSPRINT PICKUP @(COUNTER FIELD)
$CALL @EDTPL,((R8),PRNT_BFR,(R10),
   (R9),=AL4(MAX_PLINE)),
   BM=BASR,
   MF=(E,@@CALL)

SPACE 1

B NXT_BCDS GET NEXT BCDs ENTRY
UNAL_BAD DS @H
SPACE 1

*---------------------------------------------------------------------------*
* USE $EDTML AND $EDTPL TO OUTPUT THE APPROPRIATE MESSAGE                *
*---------------------------------------------------------------------------*

SPACE 1

$EDTML 15,(R8),ME@AGE@
ICM R10,B'1111',@DCB_SYSPRINT PICK UP THE @(DCB)
ICM R9,B'1111',@CNTR_SYSPRINT PICKUP @(COUNTER FIELD)
$CALL @EDTPL,((R8),PRNT_BFR,(R10),
   (R9),=AL4(MAX_PLINE)),
   BM=BASR,
   MF=(E,@@CALL)

SPACE 1

B CLOSE_FILES GO CLOSE FILES AND EXIT
LM R7,R9,SERVICE9 RELOAD
BXLE R7,R8,LOP_Dyna GO SEE IF WE HAVE ANOTHER BCDs
SPACE 1

*---------------------------------------------------------------------------*
* WE HAVE PROCESSED ALL OF THE BCDs CLUSTERS THAT WERE SPECIFIED IN       *
* THE SYSIN DATASET. NOW WE CAN OUTPUT THE AUDIT TABLE.                    *
*---------------------------------------------------------------------------*

SPACE 1

LM R3,R5,OUT_TBL PICK UP VALUES FOR BXLE LOOP
LA R5,0(R4,R5) BUMP IT

AUDT_OUT DS @H

LA R0,REPT_BCC POINT TO BUFFER
LH R1,REPT_BFR GET THE LENGTH
ICM R15,B'1111',=XL'4000000' GET PAD AND LENGTH
MVCL R0,R14 PROPAGATE THE BLANKS
ICM R7,B'1111',0(R3) GET THE COUNTER
CVD R7,DUBLWORK CONVERT IT TO DECIMAL
MVCL REPT_BUF(L'EDITPL12),EDITPL12 MOVE IN EDIT PATTERN
ED REPT_BUF(L'EDITPL12),DUBLWORK+3 EDIT THE DATA
MVCL REPT_BUF+L'EDITPL12+2(44),8(R3) GET ENTRY TYPE
SPACE 1

ICM R10,B'1111',@DCB_REPORT
ICM R9,B'1111',@CNTR_REPORT
$CALL @EDTPL,(REPT_BFR,PRNT_BFR,(R10),
   (R9),=AL4(MAX_PLINE)),
   +

© 2002, Xerphon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
BM=BASR, + 
MF=(E,@@CALL)
BXLE R3,R4,AUDT_OUT BXLE THROUGH THE AUDIT TABLE
SPACE 1
B CLOSE_FILES GO TO COMMON EXIT POINT
SYN_SYSPRINT DS ØH
SPACE 1
*---------------------------------------------------------------------*
* SYNAD CONTROL POINT FOR PHYSICAL ERROR ON THE SYSOUT DATASET.      *
* ALL WE WANT TO DO IS PROVIDE A GRACEFUL EXIT FROM THE PROGRAM.      *
*---------------------------------------------------------------------*
SPACE 1
XC FLAG_SYSPRINT,FLAG_SYSPRINT SET FLAG TO INDICATE CLOSED
$EDTML Ø3,(R8),ME@@AGE@
LA R1,WTO_MSG POINT TO THE WTO
WTO TEXT=(R8), +
ROUTCD=(Z), +
MCSFLAG=(HRDCPY), +
DESC=(6), +
MF=(E,(1))
SPACE 1
MVC RET_CODE,RCØØ1Ø SET THE RETURN CODE
B CLOSE_FILES PROCEED TO COMMON EXIT POINT
SYN_REPORT DS ØH
SPACE 1
*---------------------------------------------------------------------*
* SYNAD CONTROL POINT FOR PHYSICAL ERROR ON THE REPORT DATASET.       *
* ISSUE A MESSAGE, SET A RETURN CODE AND THEN EXIT FROM THE PROGRAM.  *
*---------------------------------------------------------------------*
SPACE 1
XC FLAG_REPORT,FLAG_REPORT INDICATE THE DATASET IS CLOSED
SPACE 1
*---------------------------------------------------------------------*
* USE $EDTML AND $EDTPL TO OUTPUT THE APPROPRIATE MESSAGE            *
*---------------------------------------------------------------------*
SPACE 1
$EDTML Ø8,(R8),ME@@AGE@
ICM R1Ø,B'1111',@DCB_SYSPRINT
ICM R9,B'1111',@CNTR_SYSPRINT
$CALL @EDTPL,((R8),PRNT_BFR,(R1Ø), +
(R9),=AL4(MAX_PLINE)), +
BM=BASR, +
MF=(E,@@CALL)
SPACE 1
B CLOSE_FILES PROCEED TO COMMON EXIT POINT
SYN_SYSIN DS ØH
SPACE 1
*---------------------------------------------------------------------*
* SYNAD CONTROL POINT FOR PHYSICAL ERROR ON THE SYSIN DATASET.        *
* ISSUE A MESSAGE, SET A RETURN CODE AND THEN EXIT FROM THE PROGRAM.  *
---------------------------------------------------------------------
*------------------------------------------------------------------*

SPACE 1
XC  FLAG_SYSIN,FLAG_SYSIN  INDICATE THE DATASET IS CLOSED
SPACE 1
*------------------------------------------------------------------*

* USE $EDTML AND $EDTPL TO OUTPUT THE APPROPRIATE MESSAGE  *
*------------------------------------------------------------------*

SPACE 1
$EDTML @09,(R8),ME@AGE@
ICM  R10,B'1111',@DCB_SYSPRINT
ICM  R9,B'1111',@CNTR_SYSPRINT
$CALL @EDTPL,((R8),PRNT_BFR,(R10),
(R9),=AL4(MAX_PLINE)),
  +
  BM=BASR,
  +
MF=(E,@CALL)
SPACE 1

B  CLOSE_FILES  PROCEED TO COMMON EXIT POINT
SPACE 1
*------------------------------------------------------------------*

* COMMON CONTROL POINT FOR CLOSING ALL OF THE FILES.  *
*------------------------------------------------------------------*

SPACE 1
CLOSE_FILES  DS @H
  CLC  FLAG_BCD,FILE_OPEN
  BNE  CLOSE_SYSIN
  LA  R5,BCD_ACB  PRIME REGISTER 5
  CLOSE ((R5)),MODE=31
SPACE 1

*------------------------------------------------------------------*

* CLOSE THE SYSIN DATASET.  *
*------------------------------------------------------------------*

SPACE 1
CLOSE_SYSIN  DS @H
  CLC  FLAG_SYSIN,FILE_OPEN  Q. IS THE FILE OPEN?
  BNE  CLOSE_REPORT  A. NO, SKIP TO NEXT FILE
  LA  R14,@DCB_SYSIN  GET @(SYSIN DCB)
  CLOSE ((R14)),MODE=31
SPACE 1

*------------------------------------------------------------------*

* USE $EDTML AND $EDTPL TO OUTPUT THE APPROPRIATE MESSAGE  *
*------------------------------------------------------------------*

SPACE 1
$EDTML @05,(R8),ME@AGE@
ICM  R10,B'1111',@DCB_SYSPRINT
ICM  R9,B'1111',@CNTR_SYSPRINT
$CALL @EDTPL,((R8),PRNT_BFR,(R10),
(R9),=AL4(MAX_PLINE)),
  +
  BM=BASR,
  +
MF=(E,@CALL)
SPACE 1

*------------------------------------------------------------------*
*--------------------------------------------------------------------------*
* CLOSE THE REPORT DATASET.                                                 *
*--------------------------------------------------------------------------*

SPACE 1
CLOSE_REPORT DS ØH
  CLC  FLAG_REPORT,FILE_OPEN  Q. IS THE FILE OPEN?
  BNE  CLOSE_SYSPRINT        A. NO, SKIP TO NEXT FILE
  LA   R14,@DCB_REPORT       GET @(REPORT DCB)
  CLOSE ((R14)),MODE=31
  SPACE 1

*--------------------------------------------------------------------------*
* USE $EDTML AND $EDTPL TO OUTPUT THE APPROPRIATE MESSAGE                  *
*--------------------------------------------------------------------------*

SPACE 1
$EDTML Ø6,(R8),ME@AGE@
  ICM  R10,B'1111',@DCB_SYSPRINT
  ICM  R9,B'1111',@CNTR_SYSPRINT
  CALL $EDTPL,((R8),PRNT_BFR,(R10), (R9),=AL4(MAX_PLINE)),
       BM=BASR,
       MF=(E,@@CALL)
  SPACE 1

*--------------------------------------------------------------------------*
* CLOSE THE SYSPRINT DATASET.                                              *
*--------------------------------------------------------------------------*

SPACE 1
CLOSE_SYSPRINT DS ØH
  CLC  FLAG_SYSPRINT,FILE_OPEN  Q. IS THE FILE OPEN?
  BNE  CLOSE_SYSPRINT        A. NO, SKIP TO NEXT FILE
  LA   R14,@DCB_REPORT       GET @(SYSPRINT DCB)
  CLOSE ((R14)),MODE=31
  SPACE 1
EXIT_PROGRAM DS ØH
  SPACE 1

*--------------------------------------------------------------------------*
* WE ARE READY TO EXIT THE PROGRAM.                                       *
*--------------------------------------------------------------------------*

SPACE 1
$ESAEP$ RET_CODE
  DROP R14  TELL THE ASSEMBLER
  DROP R15  TELL THE ASSEMBLER
  TITLE 'BCDSINVT - LITERAL POOL'
  SPACE 1
FILE_OPEN EQU DCBOFOPN       USED TO INDICATE FILE STATUS
MAX_PLINE EQU 61            USED TO HELP CONTROL PRINTING
ME@AGE0 DC V(MESS@AGE$)     VCON FOR THE MESSAGES CSECT
@EDTPL DC V($EDTPL)         VCON FOR THE PUT LINE ROUTINE
EDITPL12 DC XL12'402020206B202006B202120'
  TITLE 'BCDSINVT - DEFINE ALL OF THE MODEL DCB INFORMATION'
DCB_MDLS DS ØF

SPACE 1
SYSGIN_D1 EQU *-DCB_MDLS
LET THE ASM CALC. DISPLACEMENT
SPACE 1
SYSGIN DCB DDNAME=SYSIN,
MACRF=(GL),
DSORG=PS,
LRECL=80,
DCBE=SYSIN
SPACE 1
SYSGIN_D2 EQU *-DCB_MDLS
LET THE ASM CALC. DISPLACEMENT
SPACE 1
DCBE(SYSIN) EQU *
DCBE RMODE31=BUFF,
SYNAD=SYSIN,
EODAD=SYSIN
SPACE 1
SYSGPRINT_D1 EQU *-DCB_MDLS
LET THE ASM CALC. DISPLACEMENT
SPACE 1
SYSGPRINT DCB DDNAME=SYSGPRINT,
MACRF=(PM),
DSORG=PS,
RECFM=FBA,
LRECL=133,
DCBE=SYSGPRINT
SPACE 1
SYSGPRINT_D2 EQU *-DCB_MDLS
LET THE ASM CALC. DISPLACEMENT
SPACE 1
DCBE(SYSPRINT) EQU *
DCBE RMODE31=BUFF,
SYNAD=SYSGPRINT
SPACE 1
SYSGPRINT_D3 EQU *-DCB_MDLS
LET THE ASM CALC. DISPLACEMENT
DC F'0'
SPACE 1
REPORT D1 EQU *-DCB_MDLS
LET THE ASM CALC. DISPLACEMENT
SPACE 1
REPORT DCB DDNAME=REPORT,
MACRF=(PM),
DSORG=PS,
RECFM=FBA,
LRECL=133,
DCBE=REPORT
SPACE 1
REPORT_D2 EQU *-DCB_MDLS
LET THE ASM CALC. DISPLACEMENT
DCBE(REPORT) EQU *
DCBE RMODE31=BUFF,
SYNAD=REPORT
SPACE 1
REPORT_D3 EQU *-DCB_MDLS
DC F'0'
LET THE ASM CALC. DISPLACEMENT
SPACE 1
DCB_MDLK EQU *
DCB_MDLL DC AL4(DCB_MDLK-DCB_MDL) LET ASM CALCULATE THE LENGTH
* 
ACB_MDLK ACB AM=VSAM, +
DDNAME=BCDS, +
MACRF=(IN,SEQ), +
RMODE31=ALL
* 
ACB_MOLL EQU *.-ACB_MDLK
* 
RPL_MDLR RPL AM=VSAM, +
ACB=(**.), +
AREA=(**.), +
OPTCD=LOC
RPL_MOLL EQU *.-RPL_MDLR
* 
MOD_MDLK MODCB RPL=RPL_MDLK, +
ACB=ACB_MDLK, +
AREA=RPL_MDLK, +
AREALEN=4, +
MF=L
MOD_MOLL EQU *.-MOD_MDLK
* 
SHO_MDLK SHOWCB ACB=ACB_MDLK, +
AREA=ACB_MDLK, +
LENGTH=4, +
OBJECT=DATA, +
FIELDS=(ERROR), +
MF=L
SHO_MOLL EQU *.-SHO_MDLK
TITLE 'BCDSINV - MAP OUT MACRO LISTS'
M_OPEN OPEN (,),MODE=31, MF=L
M_CLOSE CLOSE (,),MODE=31, MF=L
* 
MVC_DSNK MVC 2(*.,R7),Ø(R3) TARGET OF AN EXECUTE INSTRUCTION
MVC_DYNK MVC Ø(*.,R3),2(R7) TARGET OF AN EXECUTE INSTRUCTION
MVC_MSGS MVC Ø(*.,R1),Ø(R15) TARGET OF AN EXECUTE INSTRUCTION
* 
AUDT_BGN DS A
AUDT_SIZ DS A
AUDT_END DS A
AUDT_ONE DC XL4'00000000' COUNTER
AUDT_TYP DC XL1'21' RECORD TYPE
DS XL3 FILLER
AUDT_TXT DC CL44'DUMP VOLUME RECORD'
AUDT_ENL EQU *.-AUDT_ONE
DC XL4'00000000' COUNTER
DC XL1'22' RECORD TYPE
DS XL3 FILLER
DC CL44'DUMP CLASS RECORD'

© 2002. Reproduction prohibited. Please inform Xephon of any infringement. 41
DC XL4'00000000' COUNTER
DC XL1'26' RECORD TYPE
DS XL3 FILLER
DC CL44'MOVE BACKUP COPY'
DC XL4'00000000' COUNTER
DC XL1'27' RECORD TYPE
DS XL3 FILLER
DC CL44'BACKUP MIGRATED DATASET'
DC XL4'00000000' COUNTER
DC XL1'28' RECORD TYPE
DS XL3 FILLER
DC CL44'PRIMARY VOLUME CODE'
DC XL4'00000000' COUNTER
DC XL1'29' RECORD TYPE
DS XL3 FILLER
DC CL44'DUMP GENERATION RECORD'
DC XL4'00000000' COUNTER
DC XL1'2A' RECORD TYPE
DS XL3 FILLER
DC CL44'ABR RECORD CODE'
DC XL4'00000000' COUNTER
DC XL1'2C' RECORD TYPE
DS XL3 FILLER
DC CL44'BACKUP VOLUME'
DC XL4'00000000' COUNTER
DC XL1'30' RECORD TYPE
DC CL3'BCR' FILLER
DC CL44'BACKUP CONTROL RECORD'
DC XL4'00000000' COUNTER
DC XL1'30' RECORD TYPE
DC CL3'BVR' FILLER
DC CL44'BACKUP CYCLE VOLUME RECORD'

AUDT_LLL EQU *-AUDT_BGN
DC XL4'00000000' COUNTER
DC XL1'30' RECORD TYPE
DC CL3'DCR' FILLER
DC CL44'DUMP CONTROL RECORD'
DC XL4'00000000' COUNTER
DC XL1'FF' RECORD TYPE
DC XL3'FFFFFF' FILLER
DC CL44'DATESET BACKUP RECORDS'

AUDT_LEN EQU *-AUDT_BGN

DROP R12,R11
$ESASTG

DUBLWORK DS D
SPACE 1

*--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-*
* THE FOLLOWING 3 FULLWORDS ARE A SAVE AREA FOR REG 7, 8, AND 9.         *
*--+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-*
SPACE 1

© 2002. Xerphon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
SAVER7R9 DS 3F
SPACE 1
*---------------------------------------------------------------------*
* THE FOLLOWING 1 BYTE FIELDS ARE USED TO INDICATE FILE STATUS       *
*---------------------------------------------------------------------*
SPACE 1
FLAG_SYSIN    DS XL1
FLAG_SYSPRINT DS XL1
FLAG_REPORT   DS XL1
FLAG_BCDS     DS XL1
SPACE 1
*---------------------------------------------------------------------*
* THE FOLLOWING ADDRESS FIELDS WILL CONTAIN THE ADDRESSES OF THE      *
* DATASET DCB AND DCBE VALUES                                         *
*---------------------------------------------------------------------*
SPACE 1
@DCB_SYSIN   DS AL4
@DCBE_SYSIN  DS AL4
@DCB_SYSPRINT DS AL4
@DCBE_SYSPRINT DS AL4
@CNTR_SYSPRINT DS AL4
@DCB_REPORT  DS AL4
@DCBE_REPORT DS AL4
@CNTR_REPORT DS AL4
SPACE 1
*---------------------------------------------------------------------*
* DEFINE STORAGE FOR THE ACB AND RPL STRUCTURES                      *
*---------------------------------------------------------------------*
DS    @OF
BCDS_ACB DS (ACB_MOLL)XL1
DS    @OF
BCDS_RPL DS (RPL_MOLL)XL1
*
R_BUFF DS A
ACB_INFO DS A
DS    @OF
@MODCB DS (MOD_MOLL)XL1
DS    @OF
@showcb DS (SH0_MOLL)XL1
*---------------------------------------------------------------------*
* THE FOLLOWING SIMPLE DATA STRUCTURE WILL HOUSE THE DATASET NAMES     *
* OF THE THE BCDS CLUSTERS. CURRENT TABLE ALLOWS FOR 5.               *
*---------------------------------------------------------------------*
SPACE 1
BCDS_NUM EQU 5
BCDS_ESZ EQU 46
BCDS_DSN DS AL4
BCDS_DSP DS AL4
BCDS_DSS DS (BCDS_NUM*BCDS_ESZ)XL1
TRAN_TAB DS 256XL1    ALLOCATE SPACE FOR TRANSLATE TBL

© 2002. Reproduction prohibited. Please inform Xephon of any infringement. 43
REPT_BFR DS H
REPT_BCC DS XL1
REPT_BUF DS XL132
REPT_BFE EQU *.REPT_BCC
PRNT_BFR DS H
PRNT_BCC DS XL1
PRNT_BUF DS XL132
PRNT_BFE EQU *.PRNT_BCC
WTO_MSG WTO 'PLACE HOLDER',MF=L
@CALL DS 20F
TITLE 'BCDSINV T - ALLOCATE SPACE FOR THE OPEN PARAMETER LISTS'
@SYSIN OPEN (,),MODE=31,MF=L
@SYSIN_L EQU *.@@SYSIN
@SYSPRINT OPEN (,),MODE=31,MF=L
@REPORT OPEN (,),MODE=31,MF=L
@BCDS OPEN (,),MODE=31,MF=L
TITLE 'BCDSINV T - ALLOCATE SPACE FOR THE CLOSE PARAMETER LIST'
@#SYSIN CLOSE (,),MODE=31,MF=L
@#SYSIN_L EQU *.@#SYSIN
@#SYSPRINT CLOSE (,),MODE=31,MF=L
@#REPORT CLOSE (,),MODE=31,MF=L
@#BCDS CLOSE (,),MODE=31,MF=L
TITLE 'BCDSINV T - DYNAMIC FILE ALLOCATION WORK AREA'
SPACE 1
SVC_99RB DS F @(SVC99 REQUEST BLOCK)
RB99_000 DS XL1 LENGTH OF REQUEST BLOCK
RB99_001 DS XL1 REQUEST VERB
RB99_002 DS XL2 FLAGS BYTE # 1
RB99_004 DS F LENGTH XL1
RB99_008 DS F @(TEXT POINTERS)
RB99_012 DS F @(REQUEST BLOCK EXTENSION)
RB99_016 DS F ZERO
*
TP99_000 DS F @(TEXT UNIT)
TP99_004 DS F @(TEXT UNIT)
TP99_008 DS F @(TEXT UNIT)
*
TU99_000 DS 0F START OF THE FIRST TEXT UNIT
  DS AL2
  DS XL2
  DS XL2
  DS XL4
TU99_004 DS 0F
  DS AL2
  DS XL2
  DS XL2
  DS XL1
TU99_008 DS 0F
  DS AL2
  DS XL2
DS XL2
DS XL44
@UDT_TBL DS OF
DS (AUDT_LEN)XL1
TITLE 'BCDSINVNT - Map out the VSAM return-reason codes'
IDARMRC
TITLE 'BCDSINVNT - Map out the DCB area'
DCBD DSORG=(QS)
TITLE 'BCDSINVNT - Map out the DCBE symbolics'
SPACE 1
IHADCBE
SPACE 1
TITLE 'BCDSINVNT - Map out areas for dynamic allocate'
IEFB4D0
IEFB4D2
END BCDSINVNT
IDENTIFY END OF PROGRAM
TITLE 'MESSAGES$ - Messages CSECT'
*------------------------------------------------------------*
* Csect : MESSAGES$
* Module : N/A
* Author : Enterprise Data Technologies
* Date : N/A
* Desc  : MESSAGES$ is a CSECT which contains all of the messages
*         for a program. although it is labeled as a CSECT, it
*         does not contain any executable code. It is a simple
*         data structure that consists of a table and the messages
*         themselves. This module get included at linkage edit time*
* Macros : none
* Dsects : none
* Input : N/A
* Output : N/A
* Plists : N/A
* Calls : N/A
* Notes : the $EDTML macro can be used to lookup a message.
*------------------------------------------------------------*
EJECT
MESSAGES$ CSECT CSECT NAME
MESSAGES$ AMODE 31 SPECIFY AN ADDRESSING MODE
MESSAGES$ RMODE ANY SPECIFY THE RESIDENCY
SPACE 1
DC AL4(A_NEXT-A_FIRST) SIZE OF AN ENTRY
DC AL4((A_END-A_FIRST)/(A_NEXT-A_FIRST)) NUMBER OF ENTRIES
SPACE 1
*------------------------------------------------------------*
* Each entry in the table consists of the message number, and the
* address of the message in the CSECT. The table structure can
* accommodate 255 messages.
*------------------------------------------------------------*
SPACE 1
A_FIRST DC AL1(01),AL4(BCDSINV$01L)
A_NEXT DC AL1(02),AL4(BCDSINVT02L)
DC AL1(03),AL4(BCDSINVT03L)
DC AL1(04),AL4(BCDSINVT04L)
DC AL1(05),AL4(BCDSINVT05L)
DC AL1(06),AL4(BCDSINVT06L)
DC AL1(07),AL4(BCDSINVT07L)
DC AL1(08),AL4(BCDSINVT08L)
DC AL1(09),AL4(BCDSINVT09L)
DC AL1(10),AL4(BCDSINVT10L)
DC AL1(11),AL4(BCDSINVT11L)
DC AL1(12),AL4(BCDSINVT12L)
DC AL1(13),AL4(BCDSINVT13L)
DC AL1(14),AL4(BCDSINVT14L)
DC AL1(15),AL4(BCDSINVT15L)
DC AL1(16),AL4(BCDSINVT16L)
A_END EQU *
SPACE 1
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
* EVERY MESSAGE ENTRY IS DEFINED ACCORDING TO A STANDARD LAYOUT  *
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
SPACE 1
BCDSINVT01L DC Y(BCDSINVT01E-BCDSINVT01M) LENGTH OF THE MESSAGE
BCDSINVT01M DC C' ' CARRIAGE CONTROL BYTE
DC C'BCDSINVT-01I ' ACTUAL MESSAGE
DC C'THE SYSIN DATASET HAS BEEN OPENED'
BCDSINVT01E EQU * END OF THE MESSAGE
*
BCDSINVT02L DC Y(BCDSINVT02E-BCDSINVT02M)
BCDSINVT02M DC C' ' C'PROCESSING INPUT FROM THE SYSIN DATASET'
BCDSINVT02E EQU *
*
BCDSINVT03L DC Y(BCDSINVT03E-BCDSINVT03M)
BCDSINVT03M DC C' ' C'ERROR,ENCOUNTERED PROCESSING THE SYSPRINT DATASET'
BCDSINVT03E EQU *
*
BCDSINVT04L DC Y(BCDSINVT04E-BCDSINVT04M)
BCDSINVT04M DC C' ' C'THE CURRENT BCDS CLUSTER HAS BEEN CLOSED'
BCDSINVT04E EQU *
*
BCDSINVT05L DC Y(BCDSINVT05E-BCDSINVT05M)
BCDSINVT05M DC C' ' C'THE SYSIN DATASET HAS BEEN CLOSED'
BCDSINVT05E EQU *
* BCDSINVT06L DC Y(BCDSINVT06E-BCDSINVT06M)
  BCDSINVT06M DC C' '
      DC C'BCDSINVT-06I '
      DC C'THE REPORT DATASET HAS BEEN CLOSED'
  BCDSINVT06E EQU *
*
  BCDSINVT07L DC Y(BCDSINVT07E-BCDSINVT07M)
  BCDSINVT07M DC C' '
      DC C'BCDSINVT-07I '
      DC C'THE REPORT DATASET HAS BEEN OPENED'
  BCDSINVT07E EQU *
*
  BCDSINVT08L DC Y(BCDSINVT08E-BCDSINVT08M)
  BCDSINVT08M DC C' '
      DC C'BCDSINVT-08E '
      DC C'A PHYSICAL ERROR OCCURRED ON THE REPORT DATASET. '
      DC C'TERMINATING PROGRAM EXECUTION.'
  BCDSINVT08E EQU *
*
  BCDSINVT09L DC Y(BCDSINVT09E-BCDSINVT09M)
  BCDSINVT09M DC C' '
      DC C'BCDSINVT-09E '
      DC C'A PHYSICAL ERROR OCCURRED ON THE SYsin DATASET. '
      DC C'TERMINATING PROGRAM EXECUTION.'
  BCDSINVT09E EQU *
*
  BCDSINVT10L DC Y(BCDSINVT10E-BCDSINVT10M)
  BCDSINVT10M DC C' '
      DC C'BCDSINVT-10E '
      DC C'SYSin RECORD IN ERROR. BYPASSING CURRENT INPUT'
      DC C' RECORD. PROCESSING CONTINUES.'
  BCDSINVT10E EQU *
*
  BCDSINVT11L DC Y(BCDSINVT11E-BCDSINVT11M)
  BCDSINVT11M DC C' '
      DC C'BCDSINVT-11I '
      DC C'DFHSM BACKUP CONTROL DATASET HAS BEEN OPENED'
  BCDSINVT11E EQU *
*
  BCDSINVT12L DC Y(BCDSINVT12E-BCDSINVT12M)
  BCDSINVT12M DC C' '
      DC C'BCDSINVT-12I '
      DC C'DFHSM BACKUP CONTROL DATASET HAS BEEN DYNAMICALLY '
      DC C'ALLOcATED'
  BCDSINVT12E EQU *
*
  BCDSINVT13L DC Y(BCDSINVT13E-BCDSINVT13M)
  BCDSINVT13M DC C' '
      DC C'BCDSINVT-13E '

Receiving SYSMODs FROM NETWORK with SMP/E Version 3.10

INTRODUCTION
With SMP/E Version 3 Release 1 it is now possible to receive input from a network server, in addition to tape and DASD.

This enables the delivery of SMP/E products and services over the Internet or an intranet.
SMP/E provides two new components to implement this new functionality:

- The GIMZIP/GIMUNZIP utilities—the new GIMZIP utility creates portable packages of software. These packages contain SYSMODs, RELFILE datasets, HOLDATA and additional materials such as documentation (README documents).
- The RECEIVE FROMNETWORK operand—this new variation of the RECEIVE command transfers a portable GIMZIP package from an FTP server across the network, extracts the information from the package, and then performs the traditional RECEIVE operations.

This article will describe step-by-step how to use this new SMP/E capability.

CREATING A GIMZIP PACKAGE

For demonstration purposes, we will describe how to create a package containing a USERMOD, a RIMLIB, and a README dataset.

GIMZIP utility

This package will be created using GIMZIP, the SMP/E Packaging Service Routine.

GIMZIP is a separate load module residing in the MIGLIB library and runs independently of the rest of SMP/E processing.

An example SMPE.PKG.SMPPFTFIN dataset looks like:

```plaintext
++USERMOD(AAA092).
++VER(Z038) FMID(H8B7705).
++SRC(IEFU29) DLSTLIB(AOS00).

/**
 /** DOC: IEFU29 SMFDUMP EXIT ROUTINE THAT IS ENTERED WHEN
 /** AN SMF DATASET IS SWITCHED.
 /** IT STARTS A DUMP FOR THE FULL DATASET.
 /**
 ...
 ...
 ...
```
An example SMPE.PKG.RIMLIB dataset looks like:

An example SMPE.PKG.README dataset looks like:

In order to create the package, you should run the GIMZIP utility specifying which datasets you want to include in the package.

All the files generated by GIMZIP are stored in a directory of an Hierarchical File System (HFS). This directory is called the package directory.

You need to tell GIMZIP what kind of information each dataset contains. This is important information that must be known during the RECEIVE processing to correctly handle the content of the package.

This information is coded using GIMZIP Package Control Tags, which are specified in the SYSIN dataset. The Package Control Tags follow XML syntax rules.

Sample JCL to call GIMZIP is shown below:

```
//STEP00 EXEC PGM=IKJEFT1B
//SYSPROC DD DISP=SHR,DSN=SYS1.SBPXEXEC
//SYSTSPRT DD SYSOUT=*  
//SYSTSIN DD *  
  OSHELL rm -r /tmp/pkg  
  OSHELL mkdir /tmp/pkg  
  //*
//STEP01 EXEC PGM=GIMZIP,PARM='LANGUAGE=ENU'
```
Each datasets is compressed by GIMZIP into an archive file, which is a portable image of the original data.

Example GIMZIP SYSOUT looks like:

```
1PAGE 0001   DATE 03/15/02   TIME 15:35:39   GIMZIP 31.07

---------<GIMZIP description="This is a sample package."
---------<FILEDEF name="SMPE.PKG.SMPPTFIN"
--------- description="This is a SMPPTFIN dataset."
--------- type="SMPPTFIN">
--------- </FILEDEF>
---------<FILEDEF name="SMPE.PKG.RIMLIB"
--------- description="This is the Related Installation Materials
--------- library for this package">
--------- </FILEDEF>
---------<FILEDEF name="SMPE.PKG.README"
--------- description="This is a README dataset."
--------- type="README">
--------- </FILEDEF>
--------- </GIMZIP>
GIM475001 DATA SET SMPE.PKG.SMPPTFIN WAS ARCHIVED INTO /tmp/pkg/SMPPTFIN/S0001.SMPE.PKG.SMPPTFIN.pax.Z.
GIM475001 DATA SET SMPE.PKG.RIMLIB WAS ARCHIVED INTO /tmp/pkg/S0002.SMPE.PKG.RIMLIB.pax.Z.
GIM475001 DATA SET SMPE.PKG.README WAS COPIED INTO /tmp/pkg/S0003.SMPE.PKG.README.
```
In addition to the archive files, GIMZIP also creates a packing list, GIMPAF.XML, called the package attribute file.

The package attribute file identifies the archive files included in the package. It also contains an SHA-1 hash value for each file in the package. This hash value is used for data integrity purposes and is checked during the RECEIVE processing.

GIMZIP and RECEIVE use cryptographic services provided by ICSF (Integrated Cryptographic Services Facility) to calculate these hash values. ICSF is a mandatory product in order to use this new SMP/E function.

The package directory
All the files generated by GIMZIP are stored in the package directory.

Example package directory structure is shown below:

```
/tmp/pkg/
```

Directory List

```
Select one or more files with / or action codes.

<table>
<thead>
<tr>
<th>Type</th>
<th>Perm</th>
<th>Changed (GMT)</th>
<th>Owner</th>
<th>Size</th>
<th>Fil</th>
<th>Row 1 of 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Dir</td>
<td>777</td>
<td>02/04/2002 15:12</td>
<td>SMPE</td>
<td>4000</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>_Dir</td>
<td>777</td>
<td>02/04/2002 15:12</td>
<td>SMPE</td>
<td>4000</td>
<td>..</td>
<td></td>
</tr>
<tr>
<td>_File</td>
<td>775</td>
<td>02/04/2002 15:12</td>
<td>SMPE</td>
<td>2880</td>
<td>GIMPAF.XML</td>
<td></td>
</tr>
<tr>
<td>- package attribute file</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_File</td>
<td>775</td>
<td>02/04/2002 15:12</td>
<td>SMPE</td>
<td>4800</td>
<td>GIMPAF.XSL</td>
<td></td>
</tr>
<tr>
<td>_Dir</td>
<td>775</td>
<td>02/04/2002 15:12</td>
<td>SMPE</td>
<td>4000</td>
<td>SMPPTFIN</td>
<td></td>
</tr>
<tr>
<td>_File</td>
<td>775</td>
<td>02/04/2002 15:12</td>
<td>SMPE</td>
<td>32256</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S0002.SMPE.PKG.RIML1.B.pax.Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_File</td>
<td>775</td>
<td>02/04/2002 15:12</td>
<td>SMPE</td>
<td>8181</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S0003.SMPE.PKG.README.pax.Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

The package attribute file contains package definition XML control tags that describe the contents of the package and how the package was created.

GIMPAF.XML looks like:

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="GIMPAF.XSL" ?>
```
At this point the package directory must be made available on the OS/390 FTP server in order to allow remote users to RECEIVE it using the FROMNETWORK operand.
RECEIVING THE PACKAGE ACROSS THE NETWORK

RECEIVE FROMNETWORK command

After the package is built and stored on the FTP server, the new FROMNETWORK operand of the RECEIVE command can be used.

In the RECEIVE FROMNETWORK command, you must tell SMP/E the address of the FTP server which contains the package.

RECEIVE FROMNETWORK JCL looks like:

```/*
//STEP01  EXEC PGM=GIMSMP,PARM='PROCESS=WAIT',REGION=8M
//*
//SMPCSI  DD DISP=SHR,DSN=ZOSI2.GLOBAL.CSI
//*
//SYSPRINT DD SYSOUT=* 
//*
//SMPCNTL DD *
   SET BDY(GLOBAL).
   RECEIVE SYSMODS
      ZONEGROUP(ALLZONES)
      FROMNETWORK(SERVER(SERVER) CLIENT(CLIENT))
   .
/*
//SERVER DD *
<SERVER host="192.168.1.2" - FTP server address
    user="xxxxxxxxx"
       - userid and password used to logon on the FTP server
    pw="zzzzzzzz"
    port="21">
   <PACKAGE file="/tmp/pkg/GIMPAF.XML" - Package Attribute File
      hash="83B596008FB13BE42D6BE31C773A5521991DD2E9"
         - hash value from PAF
      id="pkg01">
   </PACKAGE>
 </SERVER>
/*
//CLIENT DD *
<CLIENT retry="5" passv="yes">
 </CLIENT>
/*
//SMPNTS DD PATHDISP=KEEP,PATH="/tmp/SMPNTS" - SMPNTS
```

During RECEIVE FROMNETWORK processing, SMP/E first transfers the package from the FTP server and stores it in the SMPNTS directory (SMP/E Network Temporary Store).
This directory is a simple HFS directory identified to SMP/E using a DD statement or a DDDEF entry.

After all package files have been transferred and stored in the SMPNTS directory, SMP/E can extract the data from the archive files and perform traditional RECEIVE processing.

**RECEIVE FROMNETWORK** output looks like:

```
1PAGE 0001 - NOW SET TO GLOBAL ZONE        DATE 02/06/02 TIME
11:41:21 SMP/E 31.07  SMPOUT OUTPUT

GIM42401I THE FOLLOWING PARAMETERS WERE SPECIFIED ON THE EXEC
STATEMENT FOR GIMSPM: 'PROCESS=WAIT'.
    SET BDY(GLOBAL).
GIM20501I SET PROCESSING IS COMPLETE. THE HIGHEST RETURN CODE WAS
00.

RECEIVE SYSMODS
    ZONEGROUP(ALLZONES)
    FROMNETWORK(SERVER(SERVER) CLIENT(CLIENT))

GIM47600I PACKAGE pkg01 WAS SUCCESSFULLY STAGED TO THE SMPNTS.
GIM22701I RECEIVE PROCESSING WAS SUCCESSFUL FOR SYSMOD AAA0092.
GIM20501I RECEIVE PROCESSING IS COMPLETE. THE HIGHEST RETURN CODE
WAS 00.

GIM20502I SMP/E PROCESSING IS COMPLETE. THE HIGHEST RETURN CODE WAS
00. SMP/E IS AT LEVEL 31.07.
```

After RECEIVE processing completes, the SMPNTS directory structure will contain a copy of the initial package directory.

**SMPNTS directory structure looks like:**

```
Directory List

/tmp/SMPNTS/pkg01/
Select one or more files with / or action codes.

<table>
<thead>
<tr>
<th>Type</th>
<th>Perm</th>
<th>Changed (GMT)</th>
<th>Owner</th>
<th>Size</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Dir</td>
<td>775</td>
<td>02/06/2002 10:41</td>
<td>SMPE</td>
<td>4000</td>
<td>.</td>
</tr>
<tr>
<td>_Dir</td>
<td>600</td>
<td>02/06/2002 10:41</td>
<td>SMPE</td>
<td>4000</td>
<td>..</td>
</tr>
<tr>
<td>_File</td>
<td>775</td>
<td>02/06/2002 10:41</td>
<td>SMPE</td>
<td>2880</td>
<td>GIMPAF.XML</td>
</tr>
<tr>
<td>_File</td>
<td>775</td>
<td>02/06/2002 10:41</td>
<td>SMPE</td>
<td>4800</td>
<td>GIMPAF.XSL</td>
</tr>
<tr>
<td>_Dir</td>
<td>775</td>
<td>02/06/2002 10:41</td>
<td>SMPE</td>
<td>4000</td>
<td>SMPPTFIN</td>
</tr>
<tr>
<td>_File</td>
<td>775</td>
<td>02/04/2002 15:12</td>
<td>SMPE</td>
<td>32256</td>
<td></td>
</tr>
</tbody>
</table>

RECEIVE TRANSFERONLY command

The previous operations describe typical RECEIVE processing.

However, it is possible to tell SMP/E to only transfer and store the package into the SMPNTS directory and not to update the global zone and the SMPPTS dataset.

This can be done using the TRANSFERONLY operand of the RECEIVE FROMNETWORK command.

RECEIVE FROMNETWORK TRANSFERONLY JCL looks like:

```plaintext
//**
//STEP01 EXEC PGM=GIMSMP,PARM='PROCESS=WAIT',REGION=8M
//**
//SMPCSI DD DISP=SHR,DSN=ZOS12.GLOBAL.CSI
//**
//SYSPRINT DD SYSOUT=*
//**
//SMPCNTL DD *
  SET BDY(GLOBAL).
  RECEIVE SYMSMODS
    ZONEGROUP(ALLZONES)
    FROMNETWORK(SERVER(SERVER) CLIENT(CLIENT) TRANSFERONLY)
      .
    .
  .
//SERVER DD *
<SERVER host="192.168.1.2" - FTP server address
  user="xxxxxxxxx"
    - userid and password used to logon on the FTP server
  pw="zzzzzzzz"
  port="21">
  <PACKAGE file="/tmp/pkg/GIMPAF.XML" - Package Attribute File
    hash="83B59608FB13BE42D6BE31C773A5521991DD2E9"
      id="pkg01">
  </PACKAGE>
</SERVER>
//CLIENT DD *
<CLIENT retry="5" pasv="yes">
</CLIENT>
//**
```
RECEIVE FROMNETWORK TRANSFERONLY output looks like:

```
RECEIVE SYSMODS
   ZONEGROUP(ALLZONES)
   FROMNETWORK(SERVER(SERVER) CLIENT(CLIENT) TRANSFERONLY).

GIM2604W  THE SYSMODS OPERAND IS BEING IGNORED SINCE TRANSFERONLY WAS SPECIFIED ON THE FROMNETWORK OPERAND.
GIM2604W  THE ZONEGROUP OPERAND IS BEING IGNORED SINCE TRANSFERONLY WAS SPECIFIED ON THE FROMNETWORK OPERAND.

GIM4760I  PACKAGE pkg01 WAS SUCCESSFULLY STAGED TO THE SMPNTS.
GIM2050I  RECEIVE PROCESSING IS COMPLETE. THE HIGHEST RETURN CODE WAS 04.

GIM2050I  SMP/E PROCESSING IS COMPLETE. THE HIGHEST RETURN CODE WAS 04. SMP/E IS AT LEVEL 31.07.
```

**RECEIVE FROMNTS command**

The companion to the TRANSFERONLY operand is the ability to RECEIVE a package which is already stored in the SMPNTS directory. This is done with the FROMNTS operand of the RECEIVE command.

**RECEIVE FROMNTS JCL looks like:**

```plaintext
//*
//*STEP01 EXEC PGM=GIMSMP,PARM='PROCESS=WAIT',REGION=8M
//*
//*SMPCSI DD DISP=SHR,DSN=ZOS12.GLOBAL.CSI
//*
//*SYSPRINT DD SYSOUT=*  
//*
//*SMPNTS DD PATHDISP=KEEP,PATH='/tmp/SMPNTS'
//*
```
After the RECEIVE command is completed, there may be associated material in the GIMZIP package that is not extracted by SMP/E.

For example, DOCLIB, RIMLIB, and PGMDIR datasets can be included in a GIMZIP package but corresponding archive files remain (in a compressed format) in the SMPNTS directory after the RECEIVE operation.

To extract the data from these archive files, you should use the GIMUNZIP utility.

You have to tell GIMUNZIP the archive files you want to extract and you have also to specify the names of the target datasets.
Creating a C structure from an Assembler DSECT

In an earlier article entitled *Interfacing Assembler programs with IBM C* (see *MVS Update*, Issue 190, July 2002), examples of Assembler programs invoking C subroutines and C programs invoking Assembler subroutines were demonstrated. This has a very practical application and can prove to be a useful tool. In this article, we’ll take a look at how to create a C structure definition from an Assembler DSECT mapping, and how the DSECT variable names can be properly referenced in a C
program using the structure definition. This capability increases the flexibility in how information can be passed back and forth between Assembler and C programs.

**USING CBC3DSCT TO CREATE A C STRUCTURE**

Most high-level languages provide a method of grouping a series of variables under a common reference. In C, this is accomplished with a ‘structure’. If you are using the IBM C/C++ compiler and you want to exchange information between C programs and Assembler programs, you are provided with a tool that allows you to convert Assembler DSECT mappings into C structure definitions. This tool is the CBC3DSCT conversion utility.

In order to make use of this utility, your Assembler program must be assembled with the ADATA parameter. The output created in the SYSADATA output dataset is then used as input data to the structure definition conversion utility. A sample procedure that combines the assembly and conversion utility steps is provided in CBC.SCBCPRC(EDCDSECT).

For example, if you pass the LENCALC program included with this article through the EDCDSECT procedure as follows:

```plaintext
//PROCS JCLLIB ORDER=(CBC.SCBCPRC)
//STRUCT EXEC EDCDSECT,
// INFILE='assemble.source.code(LENCALC)',
// OUTFILE='c.structure(PARMAREA)',
// DPARM='NOLOWERCASE,EQUATE,SECT(PARMAREA)'
//ASSEMBLE.SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
// DD DSN=CEE.SCEEMAC,DISP=SHR
```

It will generate the following C structure definition in the OUTFILE dataset:

```plaintext
#pragma pack(packed)

struct PARMAREA {
    unsigned char  PARMDATA[256]; /* Parameter data save area */
    int PARMLEN; /* Parameter length returned by GETLEN*/
    int PARMLEN2; /* Parameter length calculated locally*/
};
#pragma pack(reset)
```

The GETLEN C program shows how the fields of the PARMAREA DSECT can be referenced in a C program using the corresponding structure definition. By using this technique to pass parameter data between Assembler and C programs, you can minimize time spent on program modifications if passed parameters are part of the change. You can simply add fields to the DSECT and structure definitions and the program calling protocol can remain intact.

POINTS OF NOTE

Square brackets [ ] are used frequently in C/C++ programs. Although the source code for the GETLEN C program and PARMAREA structure definition in this article show square brackets, when you use this code on your OS/390 system, the left square bracket, [ , should be converted to hexadecimal value X'AD' and the right square bracket, ], should be converted to hexadecimal value X'BD' prior to compilation.

ASSEMBLY, COMPILATION, AND LINKEDIT

Use a standard high-level Assembler job to assemble the LENCALC Assembler program. Be sure to save the resulting object code in an object code dataset for use in the linkedit job.

The GETLEN C program can be compiled using the following JCL:

```
//PROCS JCLLIB ORDER=(CBC.SCBCPRC)
//STEP1 EXEC EDCC,CPARM=LIST,
// CPARM2='RENT,NOSEARCH,SOURCE',
// CPARM3='NOMAR,NOSEQ,NOOPT,LANG_LVL(EXTENDED),LONGNAME,SSCOMM',
// INFIL=c.source.code(GETLEN)
//COMPILE.SYSLIN DD DSN=object.code.pds(GETLEN),DISP=SHR
```

The object code from the C compile job should be run through a pre-link step. The pre-linker (or binder) is required:

- If your C programs will be re-entrant.
If your C programs are using subroutine names that are greater than eight characters.

If you want to link together multiple C object code members created from separate compile steps.

The following pre-link JCL can be used for the GETLEN object code:

```
//PLKEDI EXEC PGM=EDCPRLK,PARM='UPCASE',
//REGION=2048K
//SYSMGS DD DNAME=CEE.SCEEMSGP(EDCPMSGE),DISP=SHR
//SYSLIB DD DUMMY
//SYSSOBJ DD DSN=object.code.pds,DISP=SHR
//SYSSMOD DD DSN=object.code.pds(GETLENP),DISP=SHR
//SYSSOUT DD SYSOUT=* 
//SYSSPRINT DD SYSOUT=* 
//SYSSIN DD *
INCLUDE SYSSOBJ(GETLEN)
```

When the LENCALC program has been assembled and the GETLENP object module has been created from the pre-link step, use the following JCL to create the LENCALC load module:

```
//IEWL EXEC PGM=HEWLH096,PARM='XREF,LIST,MAP,RENT'
//SYSPRINT DD SYSOUT=* 
//SYSSUTI DD UNIT=SYSDA,SPACE=(CYL,(2,1))
//OBJECT DD DSN=object.code.pds,DISP=SHR
//SYSSLIB DD DSN=CEE.SCEELKED,DISP=SHR
//SYSSMOD DD DSN=load.library,DISP=SHR
//SYSSLIN DD *
INCLUDE OBJECT(LENCALC)
INCLUDE OBJECT(GETLENP)
ENTRY LENCALC
NAME LENCALC(R)
```

Comments in the LENCALC source provide sample JCL for running the LENCALC load module.

**GETLEN.C**

```
/*
 * This is a C program subroutine that is invoked from a calling
 * Assembler program that has established the main() enclave. The
 * GETLEN subroutine will be invoked via a standard assembler
 * CALL macro from the calling program.
 */
```
* This subroutine expects one incoming parameter.
  * `PARM1`: is the address of a structure control block. The
  * structure definition has been created using the
  * CBC3DSCT program which is used to create C structure
  * definitions from assembler DSECTs.
  */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

/*
 * If the PARMAREA structure is to be activated through the use of
 * a header file be sure to include the structure definition in
 * member PARMAREA in one of the SYSLIB datasets of the compile
 * JCL. For now, the header file include is commented out and the
 * PARMAREA structure definition is included inline in the code.
 */
#include <parmarea.h>

/*
 * The following PARMAREA structure definition has been created by
 * passing the LENCALC assembler program through the EDCDSECT
 * procedure using the following DPARM options:
 *  DPARM='NOLOWERCASE,EQUATE,SECT(PARMAREA)'
 * 
 * The EDCDSECT procedure can generally be located in CBC.SCBCPRC.
 */
#pragma pack(packed)
struct PARMAREA {
  unsigned char PARMDATA[256]; /* Parameter data save area */
  int        PARMLEN;  /* Parameter length returned by GETLEN*/
  int        PARMLEN2; /* Parameter length calculated locally*/
};
#pragma pack(reset)

/* Indicate to the compiler that standard OS linkage will be used. */
#define __cplusplus
extern "C" int GETLEN(struct PARMAREA*);
#else
#define OS
#define __cplusplus
#endif

int GETLEN(struct PARMAREA *parms) {
  int i;
  /*
   * Determine the length of the character string and update the
   * return length field with the appropriate value.
   */
  i = strlen(parms->PARMDATA);
  parms->PARMLEN = i;
  printf("Parameter data is: %s\n", parms->PARMDATA);
  printf("Parameter length is: %d\n", parms->PARMLEN);
/* * Compare the passed pre-calculated length with the strlen() * value from this program.* /
if (parms->PARMLEN != parms->PARMLEN2)
{
    printf("strlen() length of %d differs from passed length of %d\n",
            parms->PARMLEN,parms->PARMLEN2);
    return(-1);
}
else
{
    printf("strlen() length is consistent with passed length of %d\n",
            parms->PARMLEN2);
    return(0);
}

LENCALC.ASM

******************************************************************************
* The LENCALC program is used to call a C subroutine to calculate *
* and verify the length of an input parameter string. This *
* combination of programs demonstrates the C structure creation *
* program CBC3DSC and its ability to create a C structure *
* definition from an assembler DSECT mapping. *
* *
* The LENCALC assembler program will pass the address of the *
* PARMAREA control block. The GETLEN C subroutine will use the *
* passed parameter address to reference the fields of the *
* PARMAREA control block using the corresponding C structure *
* definition.
* *
* The following macros are required to set up and take down the *
* environment that makes this operation possible:
* CEEENTRY *
* CEEENTRY *
* CEEENTRY *
* CEEENTRY *
* CEEENTRY *
* *
* The following restrictions are in effect for the CEEENTRY macro: *
* With MAIN=YES:
* BASE= can be any register R3-R11 (R11 is the default)
* R12 is to the address of the CEECAA
* R13 is to the address of the CEEDSA
* PARMREG= the parameter list address (R1 is the default)
* The following sample JCL can be used to run the LENCALC program after the GETLEN C program has been compiled and pre-linked:

* //LENCALC EXEC PGM=LENCALC,PARM='ANYPARAMETERDATA'
* //STEPLIB DD DSN=load.library,DISP=SHR
* //SYSPRINT DD SYSOUT=* 

********************************************************************************
LENCALC CEEENTRY PPA=MAINPPA, "Label of CEEPQA mapping macro " **X
AUTO=WORKSIZE, "Size of DSA & local work area " **X
MAIN=YES, "This rtln is main rtln in enclave" **X
EXECOPS=NO, "No runtime options in params " **X
PARMREG=R1, "R1 is the default parm reg " **X
BASE=R11, "R11 is the default base reg " **X
PLIST=HOST "Standard JCL PARM= parm list " **
********************************************************************************
USING WORKAREA,R13 "Set addressability to temp storage"
********************************************************************************
LTR R4,R1 "Any parameter?"
BZ RETURN04 "No - set return code and exit"
L R3,0(,R1) "Get parameter address"
W R3,=X'FFFFFFF' "Turn off x'80' bit"
LTR R3,R3 "Any parameter?"
BZ RETURN04 "No - set return code and exit"
CLC 0(2,R3),=H'0' "Any parameter data?"
BE RETURN04 "No - set return code and exit"
STORAGE OBTAIN,LENGTH=PARMALN,LOC=ANY
LR R9,R1 "Copy storage address"
USING PARMAREA,R9 "Set addressability"
XC PARMDATA(256),PARMDATA "Sanitize the target area"
XR R15,R15 "Clear R15"
ICM R15,B'0011',0(R3) "Capture parm length"
ST R15,PARMLEN2 "Save parm length"
BCTR R15,0 "Reduce by one for EX"
EX R15,PARMVCC "Copy the parm data"
********************************************************************************
CALL GETLEN, "GETLEN is the C subroutine " **X
(PARMAREA), "Address of parameter cntl blk " **X
VL,MF=(E,CALLST)
ST R15,RETCODE "Save the return code"
********************************************************************************
L R15,PARMLEN "Get parameter length"
CVD R15,DL1 "Convert to decimal"
L R15,DL1+4 "Load significant portion"
SRL R15,4 "Dump the 'sign'"
ST R15,DL2 "Save the length"
UNPK DL1(9),DL2(5) "Unpack the value"
NC DL1(8),=8X'0F' "Clear high order nibbles"
TR     DBL1(B)='C'0123456789' Make the value readable
MVC    WTO1WRK(WTO1LN),WTO1LST Copy WTO model
MVC    WTO1WRK+34(4),DBL1+4 Copy readable parm length
WTO    MF=(E,WTO1WRK)     Issue WTO
B      RETURN00     Set proper return code
*****************************************************************************
RETURN  DS  0H
          L  R5,RETCODE     Load return code
          CEETERM  RC=(R5),MF=(E,CEETERMW)
RETURN00 DS  0H
          STORAGE RELEASE,LENGTH=PARMLN,ADDR=(R9)
          B  RETURN     Return
RETURN04 DS  0H
          MVC  RETCODE(4),='F'4'   Set return code value
          B  RETURN     Return
*****************************************************************************
PARMMVC MVC  PARMDATA(*-*),2(R3) Copy parm data
*****************************************************************************
WTO1LST WTO  'LEN=XXX - Parameter length is XXXX'            ,
          X  ROUTCDE=(1),DESC=(6),MF=L
WTO1LN EQU  *-WTO1LST
*****************************************************************************
LTORG ,
*
MAINPPA  CEEPPA     Constants describing the code block
*       =====================================================================*
*       The Workarea and DSA  
*       =====================================================================*
WORKAREA DSECT
          ORG  <+CEEDASZ     Leave space for the DSA fixed part
          CALLLST  CALL  ,(0,0,0,0,0,0,0,0,0),VL,MF=L
          CEETERMW CEETERM MF=L
*
          RETCODE  DS  F       Return code
          WTO1WRK  DS  0D,CL(WTO1LN) WTO work area
          DBL1  DS  2D     A work area
          DBL2  DS  2D     A work area
          DS  0D
          WORKSIZE EQU  *-WORKAREA
PARMAREA DSECT
          PARMDATA  DS  CL256 Parameter data save area
          PARMLEN  DS  F     Parameter length returned by GETLEN
          PARMLEN2 DS  F     Parameter length calculated locally
          PARMALN EQU  *-PARMAREA
*****************************************************************************
CEEDSA     Mapping of the Dynamic Save Area
CEECAA     Mapping of the Common Anchor Area
*
RØ EQU 0
CONCLUSION

If you choose to use Assembler and C programs together when developing applications, you should definitely consider using C structure definitions for passing data referenced in Assembler DSECTs between the two language environments. If this technique is employed, future program maintenance can be minimized especially if additional parameters need to be passed between programs.

Systems Programmer
(Canada) © Xephon 2002

A weekly enterprise-oriented news service is available free from Xephon. Each week, subscribers receive an e-mail listing around 40 news items, with links to the full articles on our Web site. The articles are copyrighted by Xephon – they are not syndicated, and are not available from other sources.

To subscribe to this newsletter, send an e-mail to news-list-request@xephon.com, with the word subscribe in the body of the message.
Listing APF libraries

The following program lists APF-authorized libraries by issuing the macro CSVAPF. The list is displayed on the screen, and includes the dataset name and the volume name. If the dataset is managed by SMS, then an ‘*SMS*’ message appears instead of the volume.

The program has no arguments. The APF library chain is listed fully, in the default order.

This program can be compiled in any module library, and invoked by calling the module. For that purpose I use a REXX EXEC, with the same name as the program (LISTAPF), simply containing that call, something like:

```plaintext
/* REXX */
call "'module.loadlib(listapf)"
```

LISTAPF SOURCE CODE

```plaintext
*======================================================================================================================*
* LISTAPF - Program to list APF authorized libraries by issuing macro CSVAPF.                          *
*======================================================================================================================*
LISTAPF CSECT
LISTAPF AMODE 31
LISTAPF RMODE 24
   SAVE (14,12)
   LR R12,R15
   USING LISTAPF,R12
   ST R13,SAVE+4
   LA R11,SAVE
   ST R11,8(R13)
   LR R13,R11
   B CONTINUE
   DC CL16' LISTAPF 1.1 '  
   DC CL9'&SYSDATE'
   *
   CONTINUE DS 0H
   BAL R11,GETSTOR      Acquire default storage
   LA R13,SAVE1        Savearea for csvapf
   *
   GETAPF EQU *        Issue csvapf request
```
CSVAPF REQUEST=LIST,  
   ANSAREA=(R9),  
   ANSLEN=(R8),  
   RSNCODE=RSNCODE,  
   RETCODE=RETCODE
CLC RETCODE,=AL4(CSVAPFRC_OK)  Request ok?
BE LOOP0  Yes, branch ahead
CLC RETCODE,=AL4(CSVAPFRC_WARN)  Warning?
BNE ERRORS  No, error
NC RSNCODE,=AL4(CSVAPFRSNCODEMASK)  Clear high order bits
CLC RSNCODE,=AL4(CSVAPFRSNNOTALLDATEReturned)  More length?
BNE ERRORS  No, error
L R2,APFHTLEN-APFHDR(4)  Get required length
BAL R11,RELSTOR  Release old storage
ST R2,AREALEN  Store needed length
BAL R11,GETSTOR  Acquire new storage
B GETAPF  And request list again

*   LOOP0 EQU *  
   TPUT LINETIT,72  Send header line
   L R10,0(R9)  R10 = number of entries
   L R7,12(R9)  Jump header
   LA R9,0(R7,R9)  R9 = first entry
*   LOOP1 EQU *  
   MVC LINEVOL(6),4(R9)  Move volume
   MVC LINEFILE(44),10(R9)  and filename to line
   TPUT LINE0,72  send line
   LH R7,0(R9)  point to next entry
   LA R9,0(R7,R9)  
   BCT R10,LOOP1  loop to next entry
*   EXIT EQU *  
   BAL R11,RELSTOR  Exit program
   L R13,SAVE+4  Release storage
   LM R14,R12,12(R13)
   SR R15,R15
   BR R14
*=============================================================================*
* SUBROUTINES  
*=============================================================================*
GETSTOR EQU *  
   LA R8,AREALEN  
   STORAGE OBTAIN,  
      LENGTH=(R8),  
      ADDR=(R9)  
   ST R9,AREAADDR  
   BR R11

* RELSTOR EQU * Release storage
  LA R8,AREALEN
  L R9,AREAADDR
  STORAGE RELEASE,
    LENGTH=(R8),
    ADDR=(R9)
  BR R11

* ERRORS EQU * If any error occurred,
  MVC ZREG,RETCODE
  UNPK ZOUT9,ZREG5
  NC ZOUT8,ZTR1
  TR ZOUT8,ZTR2
  MVC ERRORRC,ZOUT8
  MVC ZREG,RSNCODE
  UNPK ZOUT9,ZREG5
  NC ZOUT8,ZTR1
  TR ZOUT8,ZTR2
  MVC ERRORRN,ZOUT8
  TPUT LINERR0,72
  TPUT LINERR1,72
  B EXIT

*============================================================================*
* WORK AREAS
*============================================================================*
SAVE DS 18F Standard save area
SAVE1 DS 18F save area for csvapf
AREALEN DC F'8192' default csvapf length
AREAADDR DS F addr of csvapf answer
RETCODE DS F csvapf returncode
RSNCODE DS F csvapf reasoncode
LINETIT DC CL72'mList of APF Authorized datasets'
LINEØ DS ØCL72 output line
  DC CL4' '
LINEFILE DS CL50
LINEVOL DS CL19
LINERRØ DS ØCL72 error output line
  DC C'Error processing request (macro CSVAPF)'
  DC CL40' '
LINERR1 DS ØCL72
  DC C'Return Code (hex): '
ERRORRC DC CL8' '
  DC CL4' '
  DC C'Reason Code (hex): '
ERRORRN DC CL8' '
  DC CL40' '
ZTR1 DC X'0F0F0F0F0F0F0F'
Call for papers

Why not share your expertise and earn money at the same time? *MVS Update* is looking for technical articles and hints and tips that experienced MVS users have written to make their life, or the lives of their users, easier. We would also be interested in articles about performance and tuning.

We will publish it (after vetting by our expert panel) and send you a cheque when the article is published. Articles can be of any length and can be sent or e-mailed to Trevor Eddolls at trevore@xephon.com, or any of the addresses shown on page 2. You can download a free copy of our *Notes for Contributors* from our Web site. Point your browser at www.xephon.com/nfc.
IBM has announced its Workload Simulator for mainframes running z/OS and OS/390, which performs stress, performance, regression, function, and capacity planning tests. It can simulate user-specified terminals and the associated messages, helping to decrease the number of terminals and reducing terminal operator time, and supports SNA, CPI-C (LU 6.2), and enhanced TCP/IP.

Included in Workload Simulator is Test Manager, an integrated user interface utility that guides the user through the test process. It helps with the development and management of test cases, the automation of test runs, and the analysis of test results.

Workload Simulator is the fifth component of IBM’s Total Cost of Operation Application Development Tools (TCO-AD Tools), adding to File Manager for z/OS and OS/390, Fault Analyzer for z/OS and OS/390, Debug Tool, and Application Monitor for z/OS and OS/390.

For further information contact your local IBM representative. URL: http://www.ibm.com.

Computer Associates has begun shipping Version 1.1 of its Advantage EDBC for ODBC and JDBC-based access to mainframe data from Linux, Unix, and Windows systems.

Enhancements to Version 1.1 include native 2.1-compliant support for JDBC, support for Windows 2000/XP, Unix, Linux, and z/OS platforms, plus improved performance, fault-tolerance, and scalability, and additional support for customers’ existing security facilities.

The software operates as a multi-threaded server to manage mainframe I/O requests from networked clients, interfacing with existing mainframe security to provide concurrent access to native VSAM, CICS/VSAM, IMS, DB2, Advantage CA-IDMS, and Advantage CA-Datacom data sources.

For VSAM and IMS, there’s an optimized mainframe SQL engine to process requests with optimum efficiency. For relational DBMS data sources, it exploits native SQL engines while negotiating any dialect differences.

It runs on z/OS, and on AIX, HP-UX, Linux, and Windows for the ODBC/JDBC client.


William Data Systems has announced Version 2 of FTPalert, which runs under z/OS, and requires an IBM TCP/IP stack.

Version 2 is considerably changed from previous versions, both in the data it captures and in the way that the data is presented. New on-line displays are available for client activity with extended filtering capabilities.

FTPalert provides facilities that support message automation, SAF security, and online reporting of FTP activity.