March 1999

In this issue

3 Automating VTAM application maintenance
11 Centralized network console – part 2
21 IP Version 6
25 Re-creating USS and Logon Mode table macros – part 2
45 Checking FTP success
48 A mailbox system for SMTP under MVS TCP/IP – 5
64 TCP/SNA news

© Xephon plc 1999
Contributions
Articles published in *TCP/SNA Update* are paid for at the rate of £170 ($250) per 1000 words and £90 ($140) per 100 lines of code for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*.

TCP/SNA Update on-line
Code from *TCP/SNA Update* can be downloaded from our Web site at http://www.xephon.com; you will need the user-id shown on your address label.

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.

Subscriptions and back-issues
A year’s subscription to *TCP/SNA Update*, comprising four quarterly issues, costs £130.00 in the UK; $190.00 in the USA and Canada; £136.00 in Europe; £142.00 in Australasia and Japan; and £140.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the March 1991 issue, are available separately to subscribers for £33.00 ($48.00) each including postage.
Automating VTAM application maintenance

At our installation we execute a number of host applications that access on-line databases to perform the majority of their work. Generally, all on-line databases are backed up nightly. Our standard method was to schedule a batch job (at the appropriate time) that instructed the Operations staff to shut down the application and respond to a WTOR when this task was completed. This WTOR program was executed as the first step in the batch job. Once the operator responded to the WTOR, the subsequent steps of this batch job performed whatever routine maintenance was desired. Frequently this was a one-step database back-up. Furthermore, once the back-up was successfully completed, the original task was immediately restarted.

Over time, this WTOR technique became a maintenance problem. It created a great deal of console message traffic and just keeping up with the back-up schedule eventually became a full-time job. Operations asked all of the responsible parties to develop a more automated solution for each of the applications that they supported. One such application was Verimation’s host e-mail package, Memo. The solution that we implemented for Memo took advantage of the fact that, as a VTAM application, Memo maintains an open VTAM ACB to communicate with the network users. The program that follows was my solution to this issue. I developed a program, VTAMTEND, that automates the shutdown of VTAM applications. This program could be used to replace the WTOR program that was executed as the first step in the back-up job.

This program required two parameters in order to execute properly. These are the started task name and the VTAM ACB name. The program can be invoked with the following three lines of JCL:

```
//STEP01 EXEC PGM=VTAMTEND,PARM='SEIMemoT,SEIMemo4'
//SYSPRINT DD SYSOUT=*  
//SYSSDUMP DD SYSOUT=*  
```

The first parameter is the VTAM ACB name and the second parameter is the started task name. The program requires both parameters separated by a comma. If the proper syntax is not provided, the
program will terminate with a condition code of twenty (20) and will issue the following message:

**AJC0020I PARAMETER SPECIFICATION ERROR**

After the input parameters are parsed, the VTAMTEND program will attempt to open the ACB name. The following messages will be issued:

**AJC0001I ATTEMPTING OPEN FOR APPLID = SEIMemoT**
**AJC002I ACB OPEN FLAG = 02  ACB ERROR FIELD = 58**

The OPEN macro that the program issues for the ACB named SEIMemo fails, and the ACB error field of 58 tells us that there is a duplicate ACB already open. This is exactly what we are expecting. The program specifically checks for this error code to determine that the application is executing. The next step is to issue an MVS STOP command for the started task name that was provided by the second input parameter. The following message is issued:

**AJC0003I STOP  CMD ISSUED FOR TASK = SEIMemo4**

At this point the program will issue a STIMER wait for 15 seconds and subsequently retry the ACB OPEN macro. When the OPEN macro is successful the program knows that the application is down and will indicate so by issuing the following message:

**AJC0000I OPEN SUCCESSFUL - TASK WAS NOT ACTIVE**

While there are certain situations when this is *not* always true, with well behaved applications like Memo we have not encountered any problems with this logic. After a successful OPEN, the program will CLOSE the ACB and terminate with a condition code of 0. If the program issues an OPEN macro for an ACB that is not defined to VTAM, the following message will be issued:

**AJC002I ACB OPEN FLAG = 02  ACB ERROR FIELD = 5A**
**AJC008I OPEN FAILED - APPLID WAS NOT FOUND**

Subsequently, the VTAMTEND program will terminate with a condition code of eight (8). Notice that the message IDs and the condition codes that VTAMTEND uses are synchronized – see Figure 1. The condition codes could be used in the JCL ‘COND=’ parameter to determine whether subsequent steps should be executed. Alternatively, the message IDs could be interrogated by an automation package such as NetView.
These are the only conditions that cause the program to end. In all other situations the program will STIMER and retry the OPEN macro.

By executing the VTAMTEND program as the first step of a batch job, we can bring down a VTAM application without requiring manual intervention by an operator. By using this program in conjunction with an ‘AUTOCMDS’ program, we can STOP Memo, dump the databases, and, subsequently, restart the task without operator intervention. As an example of how we use this program, I have included the batch JCL we use to DUMP the Memo databases. Since this program was written, Verimation has developed a 24x7 feature for Memo. This feature provides a more elegant solution by allowing the Memo databases to be dumped while Memo is still executing.

We are still using the VTAMTEND program at our installation. It has provided a simple and inexpensive solution to automating the Memo database dump process.

As a final note, the VTAMTEND program needs to run authorized because of the SVCs that are issued. It should be linked with the AC=1 parameter and should be executed from an APF authorized library.

**VTAMTEND**

```
VTAMTEND CSECT
*---------------------------------------------------------------------*
*              REGISTER EQUATES                                       *
*---------------------------------------------------------------------*
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
```

![Message Condition code Error](chart.png)

**Figure 1: Messages and condition codes**
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
*
USING *,R15
BEGIN  SAVE (14,12),,VTAMTEND-&SYSDATE-&SYSTIME
BALR R12,Ø  POINT BASE TO HERE
USING *,R12  USE R12 AS BASE REGISTER
ST  R13,SAV+4  STORE SAVE AREA
LA  R13,SAV  LOAD ADDRESS OF SAVE AREA
*
  L  R1,Ø(,R1)  GET POINTER TO PARM
  LH  R2,Ø(,R1)  GET LENGTH OF PARM
  LTR  R2,R2  WAS A PARM SPECIFIED
  BZ  EXIT2Ø  NO, SET RETURN CODE 2Ø, TERMIONATE
  SR  R9,R9  CLEAR REG 9
  LR  R9,R1  LOAD PARM ADDRESS
  AR  R9,R2  ADD PARM LENGTH TO PARM ADDRESS
  BCTR  R9,RØ  ... DECREMENT LENGTH BY 1 FOR COMMA
  BCTR  R9,RØ  ... DECREMENT LENGTH BY 1 LENGTH
  BCTR  R9,RØ  ... DECREMENT LENGTH BY 1 FOR EX
  MVC  PROCNAME(Ø),1(R4)  #-EXECUTED-#
  EX  R9,*-6  COPY ACBNAME
*
  SR  R3,R3  CLEAR REG 3
  LR  R3,R2  LOAD PARM LENGTH
  SR  R3,R9  SUBTRACT LENGTH OF ACBNAME
  BCTR  R3,RØ  ... DECREMENT LEN BY 1 FOR COMMA
  BCTR  R3,R0  ... DECREMENT LEN BY 1 LENGTH
  BCTR  R3,RØ  ... DECREMENT LEN BY 1 FOR EX
  MVC  PROCNAMETO(Ø),1(R4)  #-EXECUTED-#
EX R3,=*-6  COPY PROCPNAME TO STOP COMMAND
MVC TASK(Ø),1(R4)  #-EXECUTED-#
EX R3,=*-6  COPY PROCPNAME TO STOP COMMAND
LA R2,7,1(R3)  CALCULATE START COMMAND LENGTH
STH R2,START  STORE LENGTH IN PARM LIST
VTAMTEST LA R1,OPENMSG
WTO MF=(E,(1))
LTR R15,R15
BNE ABEND888  WTO FAILURE
OPEN (VTAMACB)  ATTEMPT TO OPEN VTAM ACB
TM VTAMACB+48,X'10'  ANY ERRORS ENCOUNTERED
BO CLOSEACB  NO, VTAM MUST BE UP & RUNNING
OPENFAIL LA R2,VTAMACB  POINT TO THE ACB
USING IFGACB,R2  ESTABLISH ACB ADDRESSABILITY
UNPK HEX(3),ACBOFLGS(2)
TR HEX(3),TRTAB-X'F0'
MVC WTOFLG(2),HEX
UNPK HEX(3),ACBERFLG(2)
TR HEX(3),TRTAB-X'F0'
MVC WTOERR(2),HEX
LA R1,ERRORMSG
WTO MF=(E,(1))
LTR R15,R15
BNE ABEND888  WTO FAILURE
CLI ACBERFLG,X'58'  IS ACB ALREADY OPEN ??
BE STOPØ1  ...THEN, STOP THE TASK NOW....
CLI ACBERFLG,X'5A'  IS ACB NOT FOUND ??
BE EXITØ8  ...EXIT, VTAM IS UP, ACB NOT ACTIVE
DROP R2
SWAIT STIMER WAIT,BINTVL=WAITTIME ELSE WAIT A FEW SECONDS
B VTAMTEST  AND TRY AGAIN
SPACE 2
CLOSEACB WTO 'AJCØØØØI OPEN SUCCESSFUL - TASK WAS NOT ACTIVE'
CLOSE (VTAMACB)  CLOSE THE VTAM ACB
B EXITEOJ  EXIT
STOPØ1 EQU *
L R11,16  GET CVT POINTER
L R11,Ø(R11)  GET POINTER TO TCB/ASCB WORDS
L R11,Ø(R11)  GET TCB POINTER
USING TCB,R11  PROVIDE TCB ADDRESSABILITY
MODESET MODE=SUP  GET INTO SUPERVISOR STATE
MODESET EXTKEY=SUPR
SR RØ,RØ  CLEAR REGISTER Ø
LA R1,START  POINT TO START COMMAND PARM
SVC 34  ISSUE OPERATOR COMMAND
MODESET EXTKEY=TCB,WORKREG=15
MODESET MODE=PROB  RETURN TO PROBLEM STATE
DROP R11  DROP TCB ADDRESSABILITY
LA R1,STOPMSG
WTO MF=(E,(1))
LTR R15,R15

BNE ABEND888 WTO FAILURE
SR R15,R15 SET RETURN CODE TO ZERO
B SWAIT STOP ISSUED, WAIT, CHECK ACB AGAIN

SPACE 2

ABEND888 ABEND 888,DUMP

SPACE 2

EXITØ4 WTO 'AJCØ004I OPEN FAILED - APPLID IS ALREADY OPEN'
LA R15,4 SET RETURN CODE TO 4
B EXITEØJ QUIT

EXITØ8 WTO 'AJCØ008I OPEN FAILED - APPLID WAS NOT FOUND'
LA R15,8 SET RETURN CODE TO 8
B EXITEØJ QUIT

EXIT2Ø WTO 'AJCØ02ØI PARAMETER SPECIFICATION ERROR'
LA R15,2Ø SET RETURN CODE TO 2Ø

EXITEØJ L R13,4(R13) TERMINATE JOB STEP
RETURN (14,12),RC=(15) TERMINATE JOB STEP

*---------------------------------------------------------------------*
* WORK AREAS, STORAGE, CONSTANTS, AND CONTROL BLOCKS                  *
*---------------------------------------------------------------------*

LTORG

SPACE 2

SAV DS 18F
OPENMSG DS 0F
DC AL2(MSG1ND-*)
DC X'0000'
DC C'AJCØ001I ATTEMPTING OPEN FOR APPLID = '
APPL DC CL8'
PAD DC CL6'
MSG1ND EQU *

SPACE 2

STOPMSG DS 0F
DC AL2(MSG2ND-*)
DC X'0000'
DC C'AJCØ003I STOP CMD ISSUED FOR TASK = '
TASK DC CL8'
DC CL6'
MSG2ND EQU *

SPACE 2

ERRORMSG DS 0F
DC AL2(MSGEND-*)
DC X'0000'
DC C'AJCØ002I ACB OPEN FLAG = '
WTOFLG DC C'XX'
DC C'ACB ERROR FIELD = '
WTOERR DC C'XX'
PAD2 DC CL6'
MSGEND EQU *

SPACE 2

COMMA DC C','
HEX DS CL7
SAMPLE BATCH JCL TO DUMP MEMO DATABASES

//MemoDUMP JOB (accounting info),
// 'Memo-Dump',
// CLASS=A,
// MSGLEVEL=(1,1),
// MSGCLASS=X
/*JOBPARM S=(*)
********************************************************************
//* USING THE VTAMTEND FACILITY, STOP THE Memo E-MAIL SYSTEM & WAIT *
********************************************************************
//* STEP01 EXEC PGM=VTAMTEND,
// PARM='SEIMemo,SEIMemo'
// SYSPRINT DD SYSOUT=* 
// SYSUDUMP DD SYSOUT=* 
/*
********************************************************************
//* DUMP THE CHKPT, DATABASE & CATALOG                             *
********************************************************************
//* DUMP EXEC PGM=UTLUDMP0,
// REGION=4096K, 
// TIME=1430
// STEPLIB DD DSN=Memo.LOAD,DISP=SHR
// SYSUDUMP DD SYSOUT=* 
// CHKPT DD DSN=Memo.CHKPT, 
// DISP=SHR 
//* PRIMARY GROUP 
// DATAP DD DSN=Memo.DATAP,
/ * DISP=SHR
// CATBASEP DD DSN=Memo.CATBASEP,
// DISP=SHR
// CATCHGP DD DSN=Memo.CATCHGP,
// DISP=SHR
// CATULDP DD DSN=Memo.CATULDP,
// DISP=SHR
// FILEDBP DD DSN=Memo.FILEDBP,
// DISP=SHR
// FILEIXP DD DSN=Memo.FILEIXP,
// DISP=SHR
// ** ALTERNATE GROUP
// DATAA DD DSN=Memo.DATAA,
// DISP=SHR
// CATBASEA DD DSN=Memo.CATBASEA,
// DISP=SHR
// CATCHGA DD DSN=Memo.CATCHGA,
// DISP=SHR
// CATULDA DD DSN=Memo.CATULDA,
// DISP=SHR
// FILEDBA DD DSN=Memo.FILEDBA,
// DISP=SHR
// FILEIXA DD DSN=Memo.FILEIXA,
// DISP=SHR
// WTOLOG DD DSN=Memo.WTOLOG,
// DISP=SHR
// ** --> NOTE| GENERATION DATASET FOR DUMP
// DUMP DD DSN=Memo.DAILY.DUMP(+1),
// DISP=(&CATLG,DELETE),
// UNIT=CRTG,
// DCB=((SC.MODEL.DSCB),BLKSIZE=3276Ø,LRECL=X,TRTCH=COMP)
//********************************************************************
// USING THE AUTOCMDS FACILITY, RESTART THE Memo E-MAIL SYSTEM
//********************************************************************
// *
//STEP03 EXEC PGM=AUTOCMDS
//CMDFILE DD DSN=cmd.input.data.set(SEIMemo),
// DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
/*
*

SAMPLE JCL TO ASSEMBLE AND LINK VTAMTEND

//jobname JOB (9999,9999),........
/*ROUTE PRINT LOCAL
/*
/* Characters in lower case are user specified
/*
Centralized network console – part 2

This month we continue the code to centralize at a specific point the management of different systems.

CLIST NCCSC20

/* NCCSC20 */
/***********************************************************************/
/* Called by CLIST NCCSC10. */
/* It executes : - Writing log Batch-Utility NCC. */
/***********************************************************************/
trace ?e
ARG dsnrsh codmsg rshid esito ncctxt
/***********************************************************************/
/* Routine for allocation/writing Log Utility NCC. */
/***********************************************************************/
nlog = 1
dlog = NCC.SA01.EVENT.LOG
dsnlog = NCC.SA01.EVENT.LOG||nlog
rgh.1 = copies(')',70)
rgh.2 = date(e)||' - '||time()
rgh.3 = 'RSH Sent for Event "'||codmsg||'".'
rgh.4 = 'Utility Remote Shell dsn "'||dsnrsh||'".'
rgh.5 = 'Job number "'||esito||'".'
rgh.6 = 'RSH Id "'||rshid||'".'
rgh.7 = 'Msg "'||ncctxt||'".'
NCC_Event_Log:
/*******************************************************************/
/* Event Writing into dataset log                                    */
/*******************************************************************/
ADDRESS NETVIEW
"FREE F(LLOG"nlog")"
"ALLOC F(LLOG"nlog") DA('"dsnlog"') MOD"
zz = rc
if zz¬=Ø then do
typfunc = 'Allocation'
Signal NCC_EventLog
Return
End
else do
ADDRESS MVS
"EXECIO 7 DISKW LLOG"nlog" (STEM rgh. FINIS"
zz=rc
if zz¬=Ø then do
typfunc = 'Update    '
Call NCC_EventLog
nlog = nlog + 1
if nlog = 2 then do
dsnlog = dlog||nlog
Call NCC_Event_Log
End
End
ADDRESS NETVIEW
"FREE F(LLOG"nlog")"
Return zz
NCC_EventLog:
say time() '********************************************************'
say time() '***                                                  ***'
say time() '*** NCCSCØØ  -  NetView Centralized Console.         ***'
say time() '***             typfunc Event Log in error.        ***'
say time() '***             dsnlog' RC='zz'                     ***'
say time() '***             Inform system support staff.         ***'
say time() '***                                                  ***'
say time() '********************************************************'
ADDRESS NETVIEW
"FREE F(LLOG"nlog")"
Return

CLIST NCCSC30

/* NCCSC3Ø */
/***********************************************************************/
/* Called by CLIST NCCSC0Ø.                                        */
/* It execute : - Carries out the function of refreshing global     */
/* variables that are related to the events of                      */
/* centralized console for the qualified operators.                */
/***********************************************************************/
Trace ?e
Arg rvgb env ncctxt
/***************************************************************************/
/* dsnopr = Table of qualified operators for the procedure NCC */
/*****************************************************************************/
dsnopr = 'NCC.SA01.UTL.OPR'
"FREE F(NCCOPR)"
"ALLOC DATASET('dsnopr') F(NCCOPR) SHR FREE"
if rc ≠ Ø then do
  nfile = dsnopr
  signal NCC_Acloc_Error
End
say time() ' NCC - Reading operator table 'dsnopr '...' ADDRESS MVS
  "NEWSTACK"
  "EXECIO * DISKR NCCOPR (STEM NCCO. FINIS"
  if rc ≠ Ø then do
    nfile = dsnopr
    signal NCC_Read_Error
  End
  "DELSTACK"
do i = 1 to ncco.Ø
  rtyp = substr(ncco.i,1,1)
  if rtyp ≠ '_' then iterate
  ropr = word(ncco.i,2)
  Call NCC_Opr_On
End
Return
NCC_Opr_On:
/***************************************************************************/
/* Routine to verify if qualified operator is logged-on. */
/*****************************************************************************/
say time() ' NCC - Refresh function in progress ...'
n = Ø
ADDRESS NETVIEW
NCC_Oprcode:
  'TRAP DISPLAY MESSAGES DSIØØ8I STATION:'
  "LIST "ropr"
  if rc ≠ Ø then do
    rtc = rc
    'TRAP NO MESSAGES'
    'FLUSHQ'
    SIGNAL NCC_ErrorA
  End
  'WAIT 1Ø SECONDS FOR MESSAGES'
Select
  when (EVENT()='M') then do
    'MSGREAD'
    'TRAP NO MESSAGES'
'FLUSHQ'

Select
When (MSGID()='DSIØØ8I') then do
  say time() '**************************************************'
  say time() '***                                            ***'
  say time() '*** NCCSC3Ø  -  NetView Centralized Console.   ***'
  say time() '***             Operator 'ropr' not logged-on. ***'
  say time() '***             Refresh function stopped.      ***'
  say time() '***                                            ***'
  say time() '**************************************************'
End

When (MSGID()='STATION:') then do
  nexec = 'NCCXCVG'env
  "EXCMD "ropr","nexec" "rvgb $'rvgb ncctxt
  say time() '*************************************************'
  say time() '***                                           ***'
  say time() '*** NCCSC3Ø  -  NetView Centralized Console.  ***'
  say time() '***             Operator 'ropr' logged-on.    ***'
  say time() '***             Refresh function active.      ***'
  say time() '***                                           ***'
  say time() '*************************************************'
End
Otherwise Do
  say time() '**************************************************'
  say time() '***                                            ***'
  say time() '*** NCCSC3Ø  -  NetView Centralized Console. ***'
  say time() '***      Condition Trap message not foreseen. ***'
  say time() '***                                            ***'
  say time() '**************************************************'
  ncctxt = 'Condition Trap message not foreseen.'
  'Inform System Support staff.'
  'GLOBALV PUTT ncctxt'
End
End
End
When EVENT() = 'G' then do
  'TRAP NO MESSAGES'
  'FLUSHQ'
  SIGNAL NCC_ErrorB
End
When (EVENT() = 'T') then do
  'TRAP NO MESSAGES'
  'FLUSHQ'
  SIGNAL NCC_Timeout
End
Otherwise do
  say time() '**************************************************'
  say time() '***                                            ***'
  say time() '*** NCCSC3Ø  -  NetView Centralized Console. ***'

say time() '*** Anomaly in Trap message. ***
say time() '*** Inform System Support staff. ***
say time() '***
say time() '******************************************************************************
ncctxt = 'Anomaly in Trap message. Refresh function stopped.' ,
'Inform System Support staff.'
'GLOBALV PUTT ncctxt'
Return 'KO'
End

Return
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
RIEND
Return 'KO'
End

else Signal NCC_Oprcode

NCC_Alloc_Error:
say time() '***********************************************************'
say time() '***
NCCSC3Ø - NetView Centralized Console.***
say time() '***
Error in the allocation phase ***
say time() '***
of the file: 'nfile' ***
say time() '***
Inform System Support staff. ***
say time() '***

Return

NCC_Read_Error:
say time() '***********************************************************'
say time() '***
NCCSC3Ø - NetView Centralized Console. ***
say time() '***
Error in read phase of the file: 'nfile' ***
say time() '***
Inform System Support staff. ***
say time() '***

Return

CLIST NCCXCVGI
/* NCCXCVGI */
/************************************************************************
/* Called by CLIST NCCSC3Ø. */
/* It executes : - Refresh function for NCC Events. */
/************************************************************************
trace ?e
Arg rvgb rvgb2 ncctxt
c1= '$'
interpret rvgb "='In Course'"
interpret c1||rvgb "='IH CR HR'"
interpret "'GLOBALV PUTT 'rvgb',"rvgb"
interpret "'GLOBALV GETT 'rvgb',"rvgb"
'GLOBALV PUTT ncctxt'
return

CLIST NCCXCVGS
/* NCCXCVGS */
/************************************************************************
/* Called by CLIST NCCSC3Ø. */
/* It executes : - Refresh function NCC Events after the selection */
/************************************************************************
trace ?e
Arg rvgb rvgb2 ncctxt
c1= '$'
interpret rvgb "='Selected'"
interpret cl||rvgb "='IH CG HR'"
interpret "'GLOBALV PUTT 'rvgb'.$'rvgb"
interpret "'GLOBALV GETT 'rvgb'.$'rvgb"
'GLOBALV PUTT ncctxt'
return

CLIST NCCXDEL
/* NCCXDEL */
/***************************************************************************/
/* After the function of FTP and remote shell it schedules the */
/* execution of this CLIST in order to delete the files used to */
/* interface with the DEC server (Centralized console for all */
/* systems). */
/***************************************************************************/
Trace ?e
Parse Arg dsnwrk
ADDRESS NETVIEW
"FREE F(NCCDWRK)"
/***************************************************************************/
/* Delete dataset "TEMP.NCCS.UTILITY.TXT.Tmicrosec" Type=work */
/* Delete dataset "TEMP.NCCS.UTILITY.RSH.Tmicrosec" Type=work */
/***************************************************************************/
ADDRESS NETVIEW
"ALLOC FI(NCCDWRK) DATASET('"dsnwrk'"') MOD"
"FREE FI(NCCDWRK) DATASET('"dsnwrk'"') DELETE"
say time() '**************************************************************************'
say time() '***                                                   ***'
say time() '*** NCCXDEL - NetView Centralized Console.            ***'
say time() '*** Erasure files of work:                            ***'
say time() '***                 'dsnwrk'                           ***'
say time() '**************************************************************************'
Exit

CLIST NCCSC40
/* NCCSC40 */
******************************************************************************
/* Called in log-on phase of the NetView operators. */
/* It reads the values of the global variables from the NetView */
/* environment and creates files for alignment to operator session. */
******************************************************************************
Trace ?e
Arg opr
microsec = substr(time(1),10,6)
******************************************************************************
/*dsnvgb = it files the values that the global variable contains */
dsnvgb = 'TEMP.NCCS.VGBRFH.T'microsec
ADDRESS NETVIEW
'Trap suppress messages CMN272I'
"ALLOC DATASET("dsnvgb") FILE(NCCVGB) SPACE(1,Ø) DSORG(PS)"
,"RECFM(FB) LRECL(8Ø) BLKSIZE(616Ø) TRACKS"
,"UNIT(WORKA) NEW CATALOG"
if rc ¬= Ø then do
  nfile = dsnvgb
  signal NCC_AllocVgb_Error
  "FREE F(NCCVGB)"
  Exit
End
ORDERV GETT $nccvgØ1,$nccvgØ2,$nccvgØ3,$nccvgØ4,$nccvgØ5,$nccvgØ6,'
  "$nccvgØ7,$nccvgØ8,$nccvgØ9,$nccvg1Ø,$nccvg11,$nccvg12,$nccvg13'
'GLOBALV GETT nccvgØ1,nccvgØ2,nccvgØ3,nccvgØ4,nccvgØ5,nccvgØ6,nccvgØ7'
  'nccvgØ8,nccvgØ9,nccvg1Ø,nccvg11,nccvg12,nccvg13,opncc,ncctxt'
ADDRESS MVS
"NEWSTACK"
Queue 'NCCTXT = ncctxt
Queue 'NCCVGØ1 = nccvgØ1
Queue 'NCCVGØ2 = nccvgØ2
Queue 'NCCVGØ3 = nccvgØ3
Queue 'NCCVGØ4 = nccvgØ4
Queue 'NCCVGØ5 = nccvgØ5
Queue 'NCCVGØ6 = nccvgØ6
Queue 'NCCVGØ7 = nccvgØ7
Queue 'NCCVGØ8 = nccvgØ8
Queue 'NCCVGØ9 = nccvgØ9
Queue 'NCCVG1Ø = nccvg1Ø
Queue 'NCCVG11 = nccvg11
Queue 'NCCVG12 = nccvg12
Queue 'NCCVG13 = nccvg13
Queue 'nccvgØ1 = nccvgØ1
Queue 'nccvgØ2 = nccvgØ2
Queue 'nccvgØ3 = nccvgØ3
Queue 'nccvgØ4 = nccvgØ4
Queue 'nccvgØ5 = nccvgØ5
Queue 'nccvgØ6 = nccvgØ6
Queue 'nccvgØ7 = nccvgØ7
Queue 'nccvgØ8 = nccvgØ8
Queue 'nccvgØ9 = nccvgØ9
Queue 'nccvg1Ø = nccvg1Ø
Queue 'nccvg11 = nccvg11
Queue 'nccvg12 = nccvg12
Queue 'nccvg13 = nccvg13
/* Write Global variables for NCC event */

"EXECIO * DISKW NCCVGB(FINIS)"
if rc ≠ Ø then do
  nfile = dsnvgb
  signal NCC_WriteVgb_Error
  "DELSTACK"
  ADDRESS NETVIEW
  "FREE FILE(NCCVGB)"
  Exit
End

"DELSTACK"
ADDRESS NETVIEW
"FREE FILE(NCCVGB)"
'Trap suppress messages DSI268I'

/*EXEC CLIST NCCSC45 for update global variables into session operator*/

"EXCMD "opr",NCCSC45" dsnvgb
Exit

NCC_AllocVgb_Error:
say time() '**************************************************************************'
say time() '*** NCCSC4Ø - NetView Centralized Console. ***'
say time() '*** Error in phase of allocation of the file: ***'
say time() '*** 'nfile' ***'
say time() '*** Alignment Refresh stopped. ***'
say time() '*** Inform System Support staff. ***'
say time() '*** ***'
say time() '**************************************************************************'
Return

NCC_WriteVgb_Error:
say time() '**************************************************************************'
say time() '*** NCCSC4Ø - NetView Centralized Console. ***'
say time() '*** Error in phase of writing of the file: ***'
say time() '*** 'nfile' ***'
say time() '*** Alignment Refresh stopped. ***'
say time() '*** Inform System Support staff. ***'
say time() '*** ***'
say time() '**************************************************************************'
Return

CLIST NCCSC45

/* NCCSC45 */

/* Called by CLIST NCCSC4Ø. */
/* It read the file of the global variables and update operator */
/* environment */
/* Trace ?e */
Arg dsnvgb
ADDRESS NETVIEW
'Trap suppress messages CNM272I'
"ALLOC DATASET("dsnvgb") FILE(NCCVGBO) SHR FREE"
if rc ¬= Ø then do
   nfile = dsnvgb
   signal NCC_AllocVgb_RFH_Error
   "FREE F(NCCVGBO)"
   Exit
End

say time() ' NCC - Read Events for Alignment ...'
ADDRESS MVS
"NEWSTACK"
"EXECIO * DISKR NCCVGBO (STEM NCCV. FINIS"
if rc ¬= Ø then do
   nfile = dsntab
   signal NCC_Read_RFH_Error
   ADDRESS NETVIEW
   "FREE FILE(NCCVGBO)"
   "DELSTACK"
   Exit
End

"DELSTACK" ADDRESS NETVIEW
>Status
/* Update global variables into all sessions operator */
/status
do i = 1 to nccv.Ø
   'Trap suppress messages NCC* $NCC* DSI372I'
   Interpret nccv.i
   vgbx = word(nccv.i,1)
   'GLOBALV PUTT 'vgbx
End

say time() ' NCC - Alignment Events executed ...'
"FREE FILE(NCCVGBO)"
Exit

NCC_AllocVgb_RFH_Error:
say time() '*****************************************************************************'
say time() '***
NCC_Read_RFH_Error:
say time() '*****************************************************************************'
Return
IP Version 6

The initial core IPv6 protocols were laid down in 1995, and by 2010 the Internet should be 100% compliant with IPv6.

So what is IPv6, and why is it so important?

IPv6, also known as v6 or IPng, promises to change the way corporate networks and the Internet work. It is the next-generation networking protocol. It completely redefines IP addressing and routing. It will eliminate some of IPv4’s shortfalls that have appeared as a result of the explosive increase in global Internet usage. It promises easier-to-obtain IP addresses and faster and cheaper routing, and supports features such as encryption and Quality Of Service (QOS).

Although IPv6 will take some considerable time yet to be used by everyone, now is the time for IT departments to implement the new protocol on test networks, to achieve familiarity with IPv6, and to address incompatibilities within an enterprise.

Transferring from IPv4 to IPv6 won’t be easy, although backward compatibility between IPv6 and IPv4-based applications and routers will help ease the transition. IT staff, programmers, and network engineers are going to be most affected by the new protocol, complete with steep learning curve.
ADDRESSES
IPv4 currently uses 32-bit addresses, which in theory allow for more than 4 billion hosts. However, simultaneously giving out many addresses greatly reduces the number of those available. Even if 4 billion hosts were available, before too long the addresses would run out.

IPv6, however, uses 128-bit addresses, theoretically offering a maximum of 340 trillion trillion trillion hosts. The reality though, as with IPv4, is that the number of addresses will be fewer than the theoretical maximum. However, even accepting pessimistic estimates based on routing inefficiencies and other factors, one might reasonably expect literally thousands of addresses per square metre of earth to be available. Optimists are suggesting telephone-number figures such as 665 million billion addresses per square metre!

TUNNELLING AND COMPATIBILITY
IPv6 has been created, not just to simplify routing and reduce network administration, but to meet the current and future use of IP networks in business.

In order to ease the transition to this new protocol, the specification allows for IPv4-in-IPv6 tunnelling, allowing v6 packets to travel over networks supporting the older protocol.

ASSIGNING ADDRESSES
With IPv4 it has always been difficult to tell, geographically or in network topography, where an address is located. Subsequently, Internet backbone routers have had to maintain huge tables of where to send information for any given address, with the inevitable squeeze on performance caused by an inefficient and inelegant network design.

IPv6’s obvious solution is to ensure that addresses are given out in an elegant hierarchy. The larger ISPs will be assigned huge blocks of addresses, and will pass out from these smaller blocks to subscribers, who can then pass on smaller blocks still, theoretically ad infinitum.
The hierarchical model will reduce the cost and complexity of Internet routers with the ability to represent large blocks of addresses with a single entry in routing tables. Backbone routers will then only have to look at small elements of an address to determine the packet’s destination.

HEADERS
A further inefficiency of IPv4 is the variable size of each packet header, making more work for the router because it has to look at more information than it actually needs to forward a packet to its destination. IPv6 uses a fixed-size header of 24 bytes, and also uses ‘chaining’, a system that allows for multiple headers at the beginning of a packet. Headers are an essential part of every piece of information exchanged, which identify the data’s source, destination etc, but where v4 includes many general-purpose fields in a large packet header, v6 and chaining ensure packets contain only the headers they actually need.

In order to keep the header as simple as possible, the essentials of the packet data – source, destination etc – reside in the standard v6 header, then one field of the header defines whether the payload begins after the header, or whether there is another header.

The chaining mechanism provides the capability to define additional features such as QOS, routing information, encryption (security encapsulation), and fragmentation. All of these headers have the same ‘next header’ field, which defines how the succeeding data should be treated – ie as the payload or as an additional header.

EXTENSIBILITY
IPv4 currently lacks the ability to add features, such as encryption, at the protocol level. IPv6 has been designed such that the protocol can be extended as needed without having to completely rebuild it, avoiding the ungainly patching necessary in IPv4.

UPDATING YOUR SYSTEM
A substantial level of updating of applications to support v6 addresses
by your in-house developers will, unfortunately, be necessary for seamless compatibility, because most OSs require applications to know about IP addresses. Many protocols such as TCP, ICMP, and even RIP are being upgraded to comply with IPv6, and much work is being done to maximize application compatibility with v6. Although much of this work is on-going and will require further updates as changes are made to the protocol specifications, there is an IPv6-compatible version of the Apache Web server already available.

Many networks should see an end to Dynamic Host Configuration Protocol (DHCP), a quasi-automatic address provision system, with a resulting reduction in network administration. IPv6 supports autoconfigurable hosts by using a combination of the Remote Desktop Protocol (RDP) with the host Media Access Control address for part of the IPv6 address. This should lessen the need for protocols such as DHCP; however, platform or network-specific information will still require such a mechanism.

**PROGRESS**

Just how quickly IPv6 will be implemented is open to debate: that it will be implemented is not.

However, the transition to the new protocol will not be easy. The purchasing of new equipment shouldn’t present too much of a problem: one simply ensures that it supports IPv6, and at the same time, older, non-upgradeable equipment should be identified for ultimate disposal. The learning curve for engineers, however, will be steep, with new skills to master from basic router configuration to troubleshooting and beyond.

Businesses may well be in for an expensive time; contracting in expertise in any new technology is always expensive. Having your own in-house expert in IPv6 could mean substantial savings later on.

After staff, your network infrastructure will require the greatest level of change. You will find you need to upgrade routers with newer firmware, and replace IPv4 static routers with v6 equivalents. Newer and more robust routing protocols, such as OSPF (Open Shortest Path
First), which is IPv6 compliant, will supersede older protocols such as RIPng.

Hopefully, the OS vendors will eventually produce some migration tools to shift your client hardware to v6, although Microsoft and IBM are notably silent on this matter.

Finally, developers should be thinking about IPv6 issues now, if their applications make any use whatsoever of network communications over IP. IPv6 is likely to crash some, maybe many, applications when it appears; that and some hardware too. It’s really not too early to start planning today.

A prototype installation of IPv6 should be started now. Functioning – prototype or experimental – IPv6 stacks are available for most popular OSs.

Moving to IPv6 is not going to be easy. It is a huge improvement over v4, providing simplified routing, increased address space, and greater extensibility. It may well be 10 years away today, but building and experimenting today with v6 networks, and laying basic foundations for the new protocol in your current applications, should dramatically ease the transition to IPv6 we will all be making in the future.

Nick Nourse (UK) © Xephon 1999

Re-creating USS and Logon Mode table macros – part 2

This month we continue the code to create USS and Logon Mode table macro instructions from members in SYS1.VTAMLIB.

GENUSTAB.SRC

//*********************************************************************
//* JOBSTREAM THAT WAS USED TO RECREATE AND AUTHENTICATE A USS TABLE  *
//*********************************************************************
/*STEP1 EXEC ASMACLG
//SYSLIB DD DSN=SYS1.SISTMAC1,DISP=SHR
// DD DSN=SYS1.MACLIB,DISP=SHR
//C.SYSIN DD *

TITLE 'CREATE CONTROL STATEMENTS TO REBUILD USSTSNA'
***********************************************************************
*        THIS UTILITY PROGRAM ISSUES A LOAD FOR USSTSNA, THEN         *
*        RECREATES THE CONTROL STATEMENTS THAT CAN BE USED TO         *
*        REBUILD IT. FOR RECREATING ANOTHER USS TABLE, CHANGE         *
*        USSTSNA ON THE LOAD STATEMENT TO ITS NAME.                   *
*        NOTE: ENTRIES FOR VTAM OPERATOR MESSAGES ARE NOT RECREATED.  *
***********************************************************************

GENUSTAB CSECT
PRINT NOGEN
STM 14,12,12(13)
LR R12,R15
USING GENUSTAB,R12
LA R15,S       CHAIN
ST R13,S+4
ST R15,8(R13)
LR R13,R15
OPEN (PGDCB,OUTPUT)      PREPARE TO TRANSCRIBE DATA
PUT PGDCB,=CL8Ø'USSTSNA  USSTAB FORMAT=DYNAMIC'
LOAD EP=USSTSNA          ===>  MEMBER IN VTAMLIB TO BE LOADED
LR R9,RØ               POINT TO ENTRY POINT
L R8,8(R9)           ADDRESS OF 1ST USSCMD BLOCK
EJECT

***********************************************************************
*        PROCESS A CMD BLOCK                                          *
***********************************************************************

PGCMDLUP MVC PGHOLD,PNGCLEAR CLEAR OUTPUT AREA
MVC PGHOLD(L'PGCMD'),PGCMD SET CONSTANT IN OUTPUT AREA
SPACE
L R4,4(R8)           ADDRESS OF COMMAND CHARACTER STRING
LA R3,PGHOLD+2Ø       POINT TO LOC OF CMD IN CNTL STMT
SR R1,R1             CLEAR REGISTER FOR MOVE
IC R1,Ø(R4)          FETCH LENGTH OF CHARACTER STRING
LA R2,Ø(R1,R3)       NEXT AVAILABLE OUTPUT LOCATION
BCTR R1,RØ           REDUCE FOR MOVE
EX R1,PGMVCMDDB       MOVE COMMAND INTO BODY OF STMT
EX R1,PGMVCMDN        MOVE COMMAND INTO NAME OF STMT
LR R3,R2              POINT TO NEXT AVAILABLE LOCATION
SPACE
LTR R4,R4             TEST IF FORMAT IS BAL
BP PGDOREP            BRANCH IF NOT
MVC Ø(L'PGFORMAT,R2),PGFORMAT SHOW THAT FORMAT IS BAL
LA R3,L'PGFORMAT(R3)  POINT TO NEXT AVAILABLE LOCATION
SPACE
PGDOREP LA R4,Ø(R4)   CLEAR HIGH-ORDER BIT
C R4,8(R8)            TEST IF IDENTICAL REPLACEMENT STRING
BE PGDOPUT BRANCH IF SO
SPACE
L R4,8(R8) ADDRESS OF REPLACEMENT STRING
MVC Ø(L'PGREP,R3),PGREP SET REPLACEMENT CONSTANT
LA R3,L'PGREP(R3) POINT TO NEXT AVAILABLE LOCATION
SR R1,R1 CLEAR REGISTER FOR MOVE
IC R1,Ø(R4) FETCH LENGTH OF CHARACTER STRING
BCTR R1,R0 REDUCE FOR MOVE
EX R1,PGMVREP MOVE REPLACEMENT INTO BODY OF STMT
EJECT
***********************************************************************
*     PROCESS A PARM BLOCK    *
***********************************************************************
SPACE
PGDOPUT PUT PGDCB,PGHOLD TRANSCRIBE CONTROL STATEMENT
ICM R7,15,12(R8) ADDRESS OF 1ST USSPARM BLOCK
BNE PGDOPARM BRANCH IF AVAILABLE
PGNXTCMD ICM R8,15,Ø(R8) ADDRESS OF NEXT USSCMD BLOCK
BNE PGCMDLUP BRANCH IF AVAILABLE
B PGUSSMSG PROCESS USSMSG ENTRIES
SPACE
PGDOPARM MVC PGHOLD,PGCLEAR CLEAR OUTPUT AREA
MVC PGHOLD(L'PGPARM),PGPARM SET CONSTANT IN OUTPUT AREA
SPACE
L R4,4(R7) ADDRESS OF KEYWORD CHARACTER STRING
LA R3,PGHOLD+22 POINT TO LOC OF CMD IN CNTL STMT
SR R1,R1 CLEAR REGISTER FOR MOVE
IC R1,Ø(R4) FETCH LENGTH OF CHARACTER STRING
LA R2,Ø(R1,R3) NEXT AVAILABLE OUTPUT LOCATION
BCTR R1,R0 REDUCE FOR MOVE
EX R1,PGMVREP MOVE KEYWORD INTO BODY OF STMT
LR R3,R2 POINT TO NEXT AVAILABLE HOLD LOC
SPACE
ICM R4,15,12(R7) TEST IF THERE IS A DEFAULT STRING
BE PGTSTREP BRANCH IF NOT
SPACE
MVC Ø(L'PGDEFALT,R3),PGDEFALT 'DEFAULT' CONSTANT TO HOLD
LA R3,L'PGDEFALT(R3) NEXT AVAILABLE LOCATION
IC R1,Ø(R4) FETCH LENGTH OF CHARACTER STRING
LA R2,Ø(R1,R3) NEXT AVAILABLE OUTPUT LOCATION
BCTR R1,R0 REDUCE FOR MOVE
EX R1,PGMVREP MOVE KEYWORD INTO BODY OF STMT
LR R3,R2 POINT TO NEXT AVAILABLE HOLD LOC
SPACE
PGTSTREP ICM R4,15,8(R7) TEST IF IT'S THE SAME AS THE DEFAULT
BE PGTSTNXT BRANCH IF NOT
SPACE
C R4,4(R7) TEST IF IT'S THE SAME AS THE DEFAULT
BE PGTSTNXT BRANCH IS SO
SPACE
MVC Ø(L'PGREP,R3),PGREP 'REP' CONSTANT TO HOLD
LA    R3,'PGREP(R3)      NEXT AVAILABLE LOCATION
IC    R1,'R4)      FETCH LENGTH OF CHARACTER STRING
LA    R2,'R1,R3)      NEXT AVAILABLE OUTPUT LOCATION
BCTR  R1,R0)      REDUCE FOR MOVE
EX    R1,'PGMVREP      MOVE KEYWORD INTO BODY OF STMT
S
PGTSTNXT PUT   PGDCB,PGHOLD      TRANSCRIBE CONTROL STATEMENT
ICM   R7,'R7)      TEST IF ANY USSPARMS LEFT
BNE   PGDOPARM      BRANCH IF SO
B     PGNXTCMD      LOOP POWER
EJECT

***********************************************************************
*        PROCESS USSMSG(TERMINAL OPERATOR MESSAGES)  *
***********************************************************************

SPACE
PGUSSMSG L     R8,'R9)           RETRIEVE POINTER TO 1ST USS MESSAGE
SR    R6,R6           ZERO NUMBER OF USS MESSAGE
LA    R9,'R5)           SET MAXIMUM NUMBER OF USSMSG ENTRIES
S
MVC   PGHOLD,PGCLEAR      CLEAR DETRITUS FROM OUTPUT AREA
PGDOUMSG CV      R6,'PGDOUBLE      ALTER RADIX OF USSMSG
UNPK   PGHOLD+19('R3),PGDOUBLE+6('R2)      THENCE TO EBCDIC
O      PGHOLD+21,'R240)      COMPLETE NUMERIC
MVC   PGHOLD+2('R2),PGHOLD+20)      RETAIN NUMBER OF MESSAGE
MVC   PGHOLD+9('PGUMMSGC),PGUMMSGC 'USSMSG' CONSTANT TO HOLD
S
ICM   R7,'R7)      POINT TO DEFINITION OF USS MESSAGE
BE    PGNXUMSG      BRANCH IF MESSAGE IS NOT DEFINED
BP    PGCOMPLX      BRANCH IF COMPLEX ENTRY
S

***********************************************************************
*        PROCESS A SIMPLE-MINDED USSMSG ENTRY  *
***********************************************************************

SPACE
MVC   PGHOLD+22('PGUMBUFC),PGUMBUFC STOW BUFFER HOLD
MVC   PGHOLD+22+L'PGUMBUFC('R2),PGHOLD+20) MSG NUMBER TO HOLD
PGDOHEX BAL   R2,'PGPUT      WRITE CONTROL STATEMENT
S
MVC   PGHOLD+9('PGUMFULL),PGUMFULL ALIGN MSG ON FUL WRD BNDRY
BAL   R2,'PGPUT      WRITE CONTROL STATEMENT
S
MVC   PGHOLD('PGUMAL2),PGUMAL2 GENERATE LENGTH OF MESSAGE
MVC   PGHOLD+3('R2),PGUMHOLD SET NUMBER OF MESSAGE IN HOLD MSG
MVC   PGHOLD+22('R2),PGUMHOLD SET NUMBER OF MESSAGE IN HOLD MSG
MVC   PGHOLD+29('R2),PGUMHOLD SET NUMBER OF MESSAGE IN HOLD MSG
BAL   R2,'PGPUT      WRITE ALIGMENT STATEMENT
S
MVC   PGHOLD('PGUMSGS),PGUMSGS SET STARTING NAME
MVC   PGHOLD+3('R2),PGUMHOLD SET MESSAGE NUMBER
S
LH    R5,'R7)      SIZE OF USS MESSAGE
LA R7,2(R7)  START OF USS MESSAGE
SPACE
SR R4,R4  CLEAR REMAINDER REGISTER
LA R3,24  SEGMENT SIZE
DR R4,R3  COMPUTE NUMBER OF SEGMENTS
LTR R5,R5  TEST IF QUOTIENT EQUALS ZERO
BZ PGLSTSEG  BRANCH IF SO
SPACE
PGSEGMT LA R2,6  NUMBER OF GROUPS PER SEGMENT
MVC PGHOLD+9(L'PGUMHEX),PGUMHEX SET CONSTANT IN HOLD AREA
LA R1Ø,PGHOLD+17  POINT TO OUTPUT AREA
SPACE
PGGROUP UNPK Ø(9,R1Ø),Ø(5,R7)  PREPARE TO TRANSLATE DATA TO EBCDIC
TR Ø(8,R1Ø),PGTRANS-24Ø CONVERT DATA TO EBCDIC
LA R1Ø,8(R1Ø)  NEXT AVAILABLE OUTPUT LOCATION
LA R7,4(R7)  NEXT SOURCE DATA
BCT R2,PGGROUP  COMPLETE A LINE OF DATA
MVI Ø(R1Ø),C'***'  SET ENDING QUOTE
BAS R2,PGPUT  PRINT DATA
BCT R5,PGSEGMT  PROCESS ALL SEGMENTS
SPACE
LTR R4,R4  TEST IF REMAINDER EXISTS
BE PGDOEQU  BRANCH IS NOT
SPACE
PGLSTSEG LA R1Ø,PGHOLD+17  POINT TO OUTPUT AREA
MVC PGHOLD+9(L'PGUMHEX),PGUMHEX SET CONSTANT IN HOLD AREA
SPACE 1
LR R5,R4  DIVIDEND
SR R4,R4  ZERO REMAINDER
LA R2,4  BYTES PER GROUP
DR R4,R2  COMPUTE NUMBER OF GROUPS
LTR R5,R5  TEST IF QUOTIENT IS ZERO
BE PGLSTGRP  BRANCH IF SO
SPACE 1
PGPART UNPK Ø(9,R1Ø),Ø(5,R7)  DATA TO EBCDIC
TR Ø(8,R1Ø),PGTRANS-24Ø MAKE DATA LEGIBLE
LA R1Ø,8(R1Ø)  UPDATE RECEIVING FIELD ADDRESS
LA R7,4(R7)  UPDATE SOURCE FIELD ADDRESS
BCT R5,PGPART  PROCESS ALL COMPLETE GROUPS
SPACE 1
LTR R4,R4  TEST FOR ZERO REMAINDER
BE PGENDCMR  BRANCH IF SO
SPACE 1
PGLSTGRP LR R5,R4  SAVE REMAINDER
AR R5,R5  DOUBLE FOR RECEIVING FIELD
LR R1,R5  SAVE FOR USE AS AN INDEX
SLL R5,4  ALIGN IT PROPERLY
OR R5,R4  SET LENGTHS OF RECEIVE AND SEND FIELDS
STC R5,*,L'*+1  STOW JUXTAPOSED LENGTHS INTO SECOND
UNPK Ø(Ø,R1Ø),Ø(Ø,R7)  BYTE OF UNPK INSTRUCTION
BCTR R1,Ø  REDUCE COUNT FOR TR INSTRUCTION

EX R1,PGTPART  TRANSLATE REMAINING DATA
LA R1Ø,1(R1,R1Ø)
PGENDCMR MVI 0(R1Ø),C'***'  SET ENDING QUOTE
B  PGXSCRIB  GO TO WRITE
SPACE
PGNXUMSG LA R6,1(R6)  INCREMENT NUMBER OF USSMSG
LA R8,4(R8)  POINT TO POINTER TO NEXT USSMSG
BCT R9,PGDOUMSG  PROCESS ALL USSMSG ENTRIES
EJECT
*******************************************************************************
*   GENERATE DELIMITER OF CONTROL STATEMENTS, THEN TERMINATE.  *
*******************************************************************************
SPACE
MVC  PGHOLD,PGCLEAR  SHINE OUTPUT AREA
MVC  PGHOLD+9(L'PGUMENTC),PGUMENTC SET ENDING CONTROL STATMNT
BAL  R2,PGPUT WRITE CONTROL STATEMENT
MVC  PGHOLD+9(3),PGUMENTC+3 MAY AS WELL GENERATE AN 'END'
BAL  R2,PGPUT STATEMENT FOR THE ASSEMBLER
CLOSE (PGDCB) CLEAN-UP
SVC 3 RETURN TO DUST
SPACE 3
*******************************************************************************
*   GENERATE AN EQU STATEMENT THAT WILL BE USED TO COMPUTE  *
*   THE LENGTH OF A USS MESSAGE.  *
*******************************************************************************
SPACE
PGXSCRIB BAS R2,PGPUT WRITE DATA STATEMENT
PGDOEQU MVC  PGHOLD+9(L'PGUMEQU),PGUMEQU STOW EQUATE INTO HOLD AREA
MVC  PGHOLD(L'PGUMSGE),PGUMSGE SET ENDING NAME
MVC  PGHOLD+3(2),PGUMHOLD SET NUMBER OF USS MESSAGE
BAL  R2,PGPUT TRANSCRIBE RECORD
B  PGNXUMSG ONWARD!
EJECT
*******************************************************************************
*   PROCESS A NOT SO SIMPLE-MINDED USSMSG ENTRY  *
*******************************************************************************
SPACE
PGCOMPLX MVC  PGHOLD+22(L'PGUMBUFC),PGUMBUFC STOW BUFFER HOLD
MVC  PGHOLD+30(L'PGUMBUFM),PGUMBUFM FORMAT FOR BUFFER WITH OP
MVC  PGHOLD+34(2),PGUMHOLD STOW MESSAGE NUMBER IN MESSAGE
TM  1(R7),X'80' TEST IF SCAN OPTION
MVC  PGHOLD+36(L'PGUMSCAN),PGUMSCAN ASSUME SO
L  R7,4(R7) POINT TO USS MESSAGE
BO  PGDOHEX BRANCH IF SO
MVC  PGHOLD+36(L'PGUMLUNM),PGUMLUNM ELSE SET LUNAME
B  PGDOHEX AND ENTER COMMON CODE
SPACE 3
*******************************************************************************
*   TRANSCRIBE A CONTROL STATEMENT; BLANK PGHOLD.  *
*******************************************************************************
SPACE
PGPUT PUT PGDCB,PGHOLD WRITE CONTROL STATEMENT
MVC PGHOLD,PGCLEAR CLEAR DETRITUS FROM OUTPUT AREA
BR R2 RETURN TO CALLER
EJECT

***********************************************************************
*                        CONSTANTS AND OTHER SUCH JUNK                    *
***********************************************************************

SPACE
PGDCB DCB DDNAME=USSTSNA,DSORG=PS,LRECL=8Ø,BLKSIZE=8Ø,RECFM=F, L
MACRF=PM
SPACE 2
PGDOUBLE DS D S DC 18F'Ø'
PGTRANS DC C'Ø123456789ABCDEF'
SPACE
PGTPART TR Ø(*-*,R1Ø),PGTRANS-24Ø ****** EXECUTE ONLY ********
SPACE
PGMVREP MVC Ø(*-*,R3),1(R4)  ********* EXECUTE ONLY *********
PGMVCMDB MVC Ø(*-*,R3),1(R4)  ********* EXECUTE ONLY *********
PGMVCMDN MVC PGHOLD(*-*),1(R4)  ********* EXECUTE ONLY *********
SPACE
PGUMHOLD DS CL2
PGUMEQU DC CL7'EQU  *'
PGUMFULL DC CL8'DS  ØF'
P Gum HEX DC CL8'DC  X'''
P Gum LUNUM DC CL8',LUNAME)'
P Gum SCAN DC CL6',SCAN)'
P Gum MSGS DC CL6'MSGØØS'
P Gum SGSE DC CL6'MSGØØE'
P gum AL2 DC CL34'MSGØØ  DC AL2(MSGØØE-MSGØØS)'
P gum MSGC DC CL11'USSMSG MSG='
P gum BUF DC CL11',BUFFER=MSG'
P gum BUF M DC CL4'(MSG'
P gum END DC CL6'USSEND'
SPACE
PGCMD DC CL2Ø'         USSCMD CMD='
P G PARM DC CL22'         USSPARM PARM='
P G REP DC CL5',REP='
P G FORMAT DC CL11',FORMAT=BAL'
P G DEFAULT DC CL9',DEFAULT='
P G CLEAR DC C'  '
P G HOLD DS CL8Ø
SPACE
YREGS
SPACE
END

//L.SYSPRINT DD DUMMY
//G.STEPLIB DD DISP=SHR,DSN=T5.AGØ3Z.VTAMLIB
//G.USSTSNA DD DISP=(,PASS),DSN=&USSTSNA,SPACE=(CYL,1),UNIT=DISK
//G.ABNLTERM DD SYSOUT=* 
//G.SYSUDUMP DD SYSOUT=*
/*
STEP2 EXEC ASMACL,PARM.L='LIST,LET,XREF,RENT,REUS'
0030000
SYSLIB DD DSN=SYS1.SISTMAC1,DISP=SHR
0040011
   DD DSN=SYS1.MACLIB,DISP=SHR
0040011
C.SYSIN DD DISP=(OLD,PASS),DSN=&USSTSNA
0050000
L.SYSLMOD DD DSN=&VTAMLIB(USSTGEN),DISP=(,PASS),SPACE=(CYL,(1,,1)),
01230005
   UNIT=DISK
L.SYSPRINT DD DUMMY
*/
X EXEC PGM=IMASPZAP,REGION=100K
SYSLIB DD DISP=(OLD,PASS),DSN=&VTAMLIB
SYSPRINT DD DSN=&PSNAMTAB,DISP=(,PASS),SPACE=(CYL,1),UNIT=DISK
   DUMPT USSTGEN ALL
/*
Y EXEC PGM=IMASPZAP,REGION=100K
SYSPRINT DD DSN=&PSNAMGEN,DISP=(,PASS),SPACE=(CYL,1),UNIT=DISK
SYSLIB DD DISP=SHR,DSN=T5.AG03Z.VTAMLIB
   DUMPT USSTSNA ALL
/*
********************************
** IEBIBALL UTILITY **
********************************
LBRCOMP EXEC LIBRCOMP
SYSUT1 DD DSN=&PSNAMTAB,DISP=(OLD,DELETE)
SYSUT2 DD DSN=&PSNAMGEN,DISP=(OLD,DELETE)
REPORT DD SYSOUT=* 
SYasin DD *
   REP LST=CHANGES,DDNAME=REPORT,
   OLDLINE=((1,'OLD>'),(6,(1,90)),(100,'<OLD'>)),
   NEWLINE=((5,'NEW>'),(10,(1,90),NEW),(110,'<NEW'>)),
   TITLE=('--- LOCATE CHANGES IN CARD STREAMS ---')
   OLDFILE STRING=(1,72)
   NEWFILE STRING=(1,72)
   * 
PRINT EXEC PGM=IEBGENER,REGION=118K
SYSPRINT DD DUMMY
SYsin DD DUMMY
   SYsut2 DD SYSOUT=*,DCB=(LRECL=80,BLKSIZE=80,RECFM=F),CHARS=GB12
   SYsut1 DD DISP=(OLD,DELETE),DSN=&USSTSNA

IEFACTRT.SRT
./ ADD NAME=IEFACTRT
   TITLE 'IEFACTRT - STEP AND JOB TERMINATION ACCOUNTING EXIT'
IEFACTRT CSECT
FUNCTION: This routine is part of the OS Initiator and is given control during termination of each job step and of each job. It writes the *-enclosed boxes which become part of each job's system message block output. For each job step, it gathers information about CPU and device usage and passes this information to the cost calculating routine, ISDACTRT. In return, it receives and prints out the step cost. At the end of the job, it calls ISDACTRT to process the accumulated step information, and prints out the total estimated cost of the job.

ENTRY: Entered at IEFACTRT by a call from the IEFACTLK routine of the Initiator.

INPUT: RØ - contents indicate whether step or job term entry;
(RØ) = 12 -> step entry
(RØ) = 16 -> job entry
(RØ) = 20 -> type 30 SMF record
R1 - points to the exit data table of addresses (see the listing of EXDDSECT below);
R12 - currently points to the initiator linkage control table. It is from this table that the OS SCT & JCT addresses are extracted. There is very little documentation about R12 at entry; if problems arise with new versions of OS, it could be that the assumptions herein about R12 are no longer valid.

OUTPUT: All output from IEFACTRT is in the form of SMBS written through IEFYS, an alias of IEFTB724 which is located in SYS1.AOSB3.
*** Note *** IEFACTRT is and must be designed to operate without the presence of JES2, or without a JES2 JCT for a given job. In either case, the lack of a JCT address allows CPU and device usage processing, but prevents any cost calculation.

EXITS: The only exit is a BR 14 to IEFACTLK.

SUBROUTINES: ISDACTRT to compute costs; IEFYS to print system message blocks.
*** Note *** It is the instance of calling IEFYS that makes the 64 word SAVEAREA necessary, due to OS's weird saving and SAVEAREA linking conventions. Possibly someday, someone will come up with a better way. (Perhaps BAKR is it.)
* EJECT
* ATTRIBUTES: REENTRANT, REUSABLE. (MANDATORY)
* *
* MACROS USED: LOCAL - PLINE
* FROM SYS1.MACLIB - STORAGE, WTO, CALL, CVT, IEFUCBOB,
* IEZDEB, IEFJSSIB, IEZJSCLB, IHAPSA,
* IKJCIB, IEFTIOT1, IEFASCTB, IECDIOCM,
* AND IFASMFR.
* FROM SYS1.AMODGEN - IFGRPL, IEFAJII, AND IEFJMR.
* FROM SYS1.HASPSRC (JES2 SOURCE) - $HASPEQU, $BUFFER, $CAT,
* $TQE, $JCT, $SJB, $SCAT, $MIT, AND $XECB.
* *
* -> NOTE THE IMPLICATION HERE THAT CHANGES TO THE ABOVE
* HASP CONTROL BLOCKS (FREQUENT AT SOME INSTALLATIONS)
* MAY NECESSITATE A RE-COMPILATION OF THIS ROUTINE.
* *
* WORK AREAS: THE AREA DESCRIBED BY THE DSECT "WORKAREA" IS OB-
* TAINED FROM SP 253 AT EACH ENTRY AND FREED AT EACH EXIT.
* THE AREA DESCRIBED BY "KEEPSECT" IS OBTAINED FOR EACH JOB
* FROM SP 253 DURING THE FIRST STEP ENTRY, AND FREED DURING
* THE JOB ENTRY.
* *** NOTE *** THE ADDRESS OF "KEEPSECT" IS SAVED FOR THE
* DURATION OF THE JOB IN THE JMRUCOM FIELD OF THE SMF
* COMMON EXIT TABLE.
* *
* REGISTER USAGE: THE FOLLOWING REGS ARE UNAVAILABLE FOR WORK:
* R4 - USED TO ADDRESS SPECIFIC SMF TYPE 3Ø SECTIONS;
* R7 - SECOND BASE REGISTER;
* R8 - PLNK - LINK REG FOR RETURN FROM PRINT ROUTINE;
* R9 - POINTS TO "KEEPSECT" AFTER INITIALIZATION;
* R10 - POINTS TO THE CURRENT SMF RECORD;
* R11 - BASE REG FOR IEFACTRT AND ISDACTRT;
* R12 - BASE REG FOR ISDREADY CSECT;
* R13 - POINTS TO "WORKAREA";
* *
* SPACE 1
* GBLC &SCOST
* EJECT
* MACRO
* &TAPNAME TAPINFO
* DS 0F
* PUSH PRINT
* PRINT GEN
* &TAPNAME DC CL8'&SYSECT'
* DC A(&SYSECT)
* DC CL6'&SYSTIME'
* DC CL8'&SYSDATE'
* POP PRINT
* MEND
* EJECT

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
MACRO

&NAME PLINE &RETURN

**********************************************************************
*                                                                    *
*       THIS MACRO ALLOWS A GENERAL CALL TO THE PRINTER ROUTINE      *
*       IN ORDER TO PRINT THE INFORMATION IN MSG. IF NO OPERAND      *
*       IS SPECIFIED, A SIMPLE BAS IS GENERATED. IF A LABEL IS       *
*       SPECIFIED, THE ADDRESS OF THAT LABEL IS LOADED INTO THE      *
*       REGISTER NAMED PLNK, AND RETURN IS MADE TO THAT POINT.       *
*       IF THE FIRST CHARACTER OF THE OPERAND IS '(', IT IS          *
*       ASSUMED THAT PLNK ALREADY CONTAINS THE RETURN ADDRESS.       *
*                                                                    *
**********************************************************************

AIF (T'&NAME EQ 'O').NONAME

&NAME DS ØH

_NONAME AIF (T'&RETURN EQ 'O').BAL

AIF ('&RETURN'(1,1) EQ '(').REG

LA PLNK,&RETURN ! PICK UP RETURN ADDRESS

.REG B PRINTER ! GO PRINT MSG

MEXIT

.BAL BAS PLNK,PRINTER ! PICK UP RETURN AND GO PRINT MSG

MEND

TITLE 'IEFACTRT - STEP AND JOB TERMINATION ACCOUNTING EXIT'

**********************************************************************
*                                                                    *
*              B E G I N   M A I N   P R O G R A M                    *
*                                                                    *
**********************************************************************

SPACE 1

IEFACTRT CSECT ,

STM R14,R12,12(R13) PRESERVE REGISTERS AT ENTRY
LR BASE,R15 PRIME BASE REGISTER

USING IEFACRTB,BASE,R7 TWO BASE REGS

LA R7,2048(R11) INITIALIZE SECOND

LA R7,2048(R7) BASE REGISTER

SPACE 1

LR R2,RØ SAVE EXIT TYPE INDICATOR

LR R9,R1 SAVE EXIT DATA TABLE ADDRESS

CH R2,=H'2Ø' TEST IF SMF TYPE 3Ø RECORD PRESENT

BNE CLAMQUIK BRANCH IF NOT

SPACE 1

USING PSA,RØ ESTABLISH PSA ADDRESSABILITY

USING EXDDECT,R9 FOR TEMPORARY USE ONLY

SPACE 1

* ESTABLISH ADDRESSABILITY TO SAVE AND WORK AREAS

* SPACE 1

L R1,EXDCOMTB ADDRESS OF COMMON EXIT TABLE

USING JMR,R1 ESTABLISH ADDRESSABILITY

ICM R1,15,JMRUCOM ADDRESS OF KEEPSECT
BE    CLAMQUIK       DEPART IF SUBSEQUENT JOB TERM ENTRY
DROP   R1           REMOVE ADDRESSABILITY
LA    R1,KEEPLEN(R1)  POINT TO WORKAREA
SPACE 1

*  SET UP SAVE AREA LINKAGE AND ADDRESSABILITY
SPACE 1
ST   R1,8(R13)       CHAIN FORWARD
ST   R13,4(R1)        CHAIN REVERSE
LR   R3,R13           SAVE POINTER TO SAVE AREA ABOVE
LR   WORK,R1          GET ADDR OF WORKAREA
USING WORKAREA,WORK    AND TELL ASSEMBLER
CLC  CLAMLOVE,=CL4'LOVE' TEST IF POINTER TO WORKAREA IS VALID
BNE  PPGEORR          BRANCH IF NOT
XC   TEMPD1(CLEARLEN),TEMPD1 CLEAR OUT WORK AREA
ST   R3,SAVELAST      SAVE POINTER TO UPPER SAVEAREA
ST   R9,ADDREXD      SAVE EXD ADDRESS FOR LATER USE
EJECT

*  ESTABLISH ADDRESSABILITY FOR ISDRATES AND SMF RECORD
SPACE 1
ST   R12,ADDRLCT     SAVE ADDR OF LCT FOR LATER USE
L    RAT,VISDRATE    GET ADDRESS OF ISDRATES CSECT
USING ISDRATES,RAT    AND TELL ASSEMBLER
L    SMF,EXDRDW      GET REC DESCRIPTOR WORD ADDRESS
USING SMFRECD3Ø,SMF   ESTABLISH ADDRESSABILITY TO TYPE 3Ø
SPACE 1
L    R9,EXDCOMTB    GET ADDR OF COMMON EXIT TABLE
USING JMR,R9         ESTABLISH TEMP ADDRESSABILITY
ICM  R9,15,JMRUCOM   ADDRESS OF KEEPSECT
USING KEEPSECT,KEEP  ESTABLISH PERM ADDRESSABILITY
SPACE 1

*  SET UP MSG LENGTH AND ADDRESS FIELDS FOR CALLS TO IEFYS
*
SPACE 1
LA   R3,L'MSG       STOW LENGTH OF ALL MESSAGES
STH  R3,MSGLEN     IN PARAMETER AREA
LA   R3,MSG        STOW ADDRESS OF MESSAGE AREA
ST   R3,MSGADDR   IN IEFYS PARAMETER AREA
SPACE 1
TIME  BIN
STM  R0,R1,TERMTIME  SAVE TIME AND DATE OF TERMINATION
EJECT

***********************************************************************
*                                                                     *
*        DETERMINE RECORD TYPE                                        *
*                                                                     *
***********************************************************************
SPACE 1
ICM  R4,15,SMF3ØSOF   OFFSET TO SUBSYSTEM SECTION
BZ   DEPART         BRANCH IF UNAVAILABLE

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
AR  R4,SMF    COMPUTE ID SECTION ADDRESS
USING SMF3ØPSS,R4    SET SUBSYSTEM SECTION ADDRESSABILITY
SPACE 1
MVC  CLAMTYPE,SMF3ØTYP+1    SAVE REASON FOR LAST ENTRY
CLI  SMF3ØTYP+1,4    TEST IF STEP TOTALS
BE  STEPEND    BRANCH IF SO
CLI  SMF3ØTYP+1,5    TEST IF JOB TERMINATION
BNE  DEPART    BRANCH IF NOT
EJECT

******************************************************************************
*                                                                     *
*        THIS ROUTINE IS ENTERED FOR JOB TERMINATION                    *
*                                                                     *
******************************************************************************

SPACE 1
JOBEND  BAS  PLNK,PRINTHDR    GO WRITE FIRST LINE OF STARS

******************************************************************************
*                                                                     *
*        PROCESS JOB TERMINATION RECORDS (ACCOUNTING SECTION)           *
*                                                                     *
******************************************************************************

SPACE 1
STCM  SMF,7,CLAMJOB    SAVE LAST BUFFER ADDRESS
ICM  R4,15,SMF3ØAOF    OFFSET TO ACCOUNTING SECTION
BZ  FREEKEEP    CLEAN-UP IF NON-EXISTENT
AR  R4,SMF    COMPUTE ACCOUNTING SECTION ADDRESS
USING SMF3ØACS,R4    SET ACCOUNTING SECTION BASE
SPACE 1
PLINE ,    SEPARATOR LINE
MVC  BLANK2,CMRISSD    GLORY! TO OUTPUT AREA
PLINE ,    AND TRANSCRIBE IT
PLINE ,    SEPARATOR LINE
SPACE 1
MVC  HGJOBNAM,CGJOBNAM    MOVE CONSTANTS INTO
MVC  HGJOBNO,CGJOBNO    OUTPUT AREA
MVI  HGPRIO,C'P'
MVI  HGCLASS,C'Q'
MVC  HGPD,CGPD
MVC  HGSYSTEM,CGSYSTEM
MVC  HGACOUNT(L'CGACOUNT),CGACOUNT
MVC  HGPGMR,CGPGMR
PLINE ,    TITLE LINE
EJECT
MVC  HGSYSID,SMF3ØSID    MOVE SYSTEM-ID TO OUTPUT AREA
LH  RØ,SMF3ØALN    FETCH LENGTH OF ACCOUNTING SECTION
SR  R1,R1    CLEAR REGISTER FOR INSERT
ICM  R1,1,Ø(R4)    LENGTH OF PRIORITY FIELD
BZ  CMGRPRNTA    BRANCH IF AT END OF SECTION
BCTR  R1,Ø    REDUCE LENGTH FOR MOVE
EX R1, MGPRIO  MOVE PRIORITY TO OUTPUT AREA
LA R1, 1(R1)  REESTABLISH LENGTH
LA R4, 1(R1, R4)  NEXT ACCOUNTING FIELD
SR R0, R1  REDUCE BY LENGTH OF FIELD PROCESSED
BZ CMGPRNTA  BRANCH IF AT END OF ACCOUNTING SECT
SPACE 1
ICM R1, 1, Ø(R4)  LENGTH OF ACCOUNTING FIELD
BZ CMGPRNTA  BRANCH IF AT END OF SECTION
CLM R1, 1, =AL1(L'MGACOUNT)  TEST IF FIELD SIZE EXCEEDS MAXIMUM
BNH CMGO  BRANCH IF ACCOUNTING FIELD WILL FIT
ICM R1, 1, =AL1(L'MGACOUNT)  SET MAXIMUM LENGTH ALLOWABLE
CMGO BCTR R1, Ø  REDUCE LENGTH FOR MOVE
EX R1, MGACOUNT  MOVE ACCOUNT NUMBER TO OUTPUT AREA
ICM R1, 1, Ø(R4)  LENGTH OF ACCOUNTING FIELD
LA R4, 1(R1, R4)  NEXT ACCOUNTING FIELD
SR R0, R1  REDUCE BY LENGTH OF FIELD PROCESSED
BZ CMGPRNTA  BRANCH IF AT END OF ACCOUNTING SECT
SPACE 1
ICM R1, 1, Ø(R4)  LENGTH OF PROGRAMMER/DEPARTMENT FLD
BZ CMGPRNTA  BRANCH IF AT END OF SECTION
BCTR R1, Ø  REDUCE LENGTH FOR MOVE
EX R1, MGPD  MOVE PGM/DEPT FIELD TO OUTPUT AREA
EJECT
***********************************************************************
*                                                                     *
*        PROCESS JOB TERMINATION RECORDS ( IDENTIFICATION SECTION )    *
*                                                                     *
***********************************************************************
SPACE 1
CMGPRNTA L R4, SMF3ØIOF  OFFSET TO IDENTIFICATION SECTION
AR R4, SMF  COMPUTE IDENTIFICATION SECTION ADDR
USING SMF3ØID, R4  SET ID SECTION ADDRESSABILITY
SPACE 1
MVC HGJOBNAM, SMF3ØJBN  JOB NAME TO OUTPUT AREA
MVC HGJOBNO-1(5), SMF3ØJNM+3  JES2 JOB NUMBER
SPACE
LA PLNK, HGJOBN-1  POINT TO START OF JOB NUMBER
LA R15, 4  SET LOOP COUNT
CAROLING CLI Ø(PLNK), C'Ø'  TEST FOR LEADING ZERO
BNE CMGCAROL  BRANCH IF REAL NUMBER
MVI Ø(PLNK), C' '  BLANK ALL LEADING ZEROS
LA PLNK, 1(PLNK)  POINT TO NEXT DIGIT OF JOB NUMBER
BCT R15, CAROLING  LOOP POWER
SPACE
CMGCAROL MVC HGCLASS, SMF3ØCLS  JOB CLASS
MVC HGPGMR(20), SMF3ØUSR  PROGRAMMER'S NAME FIELD
SPACE 1
PLINE ,  JOB STATISTICAL INFORMATION
PLINE ,  SEPARATOR LINE
SPACE 1
MVC  HJSSTART,CJSSTART  MOVE JOB START AND STOP
MVC  HJEND,CJEND          TIME-CONSTANTS TO OUTPUT AREA
MVC  HJELAPSD,CJELAPSD    SPACE 1
PLINE ,                   SEND TIME TITLES
SPACE 1
TM    SMF3ØSTD+3,X'ØF'    TEST IF JOB START DATE IS VALID
BNO   CMG2SKP             BRANCH IF NOT
SPACE 1
L     R15,SMF3ØSIT        TIME JOB SELECTED
BAS   R8,CONVERT          CONVERT TIME
MVC   HJSTIME,CLAMWORK    MOVE IT TO OUTPUT AREA
SPACE 1
L     R15,TERMTIME        TIME JOB ENDED
BAS   R8,CONVERT          CONVERT TIME
MVC   HJETIME,CLAMWORK    MOVE IT TO OUTPUT AREA
SPACE 1
BAS   R8,CMGILL           ADD TWENTY-FOUR HOURS/DAY TO TIME
B     CMG2                SPACE 1
CMG2SKP  MVC   HJEPTIME,CMRUNOME   SHOW ERROR
B     CMG2PUT             TO ALL CONCERNED
SPACE 1
CMG2    BAS   R8,CONVERT          CONVERT ELAPSED TIME
MVC   HJEPTIME,CLAMWORK    MOVE IT TO OUTPUT AREA
SPACE 1
ICM   R14,15,SMF3ØSTD     DATE JOB SELECTED
BAS   R8,CMRGREG          CONVERT DATE
MVC   HJSDATE,CLAMHOLD    MOVE IT TO OUTPUT AREA
SPACE 1
L     R14,TERMDATE        DATE JOB ENDED
BAS   R8,CMRGREG          CONVERT DATE
MVC   HJEDATE,CLAMHOLD    MOVE IT TO OUTPUT AREA
SPACE 1
CMG2PUT  PLINE ,                   JOB START AND END DATE AND TIME
        PLINE ,                   SEPARATOR LINE
        EJECT
*
*        SET UP FOR FINAL (JOB) CALL TO ISDACTRT
*
SPACE 1
L     R15,ADDREXD         ADDRESS OF IEFACTRT PARAMETER LIST
USING EXDDSECT,R15        PROVIDE ADDRESSABILITY TO PARM LIST
L     R15,EXDCOMTB        ADDRESS OF COMMON EXIT TABLE
USING JMR,R15             ESTABLISH ADDRESSABILITY
BAS   R1,CLAMSJB          OBTAIN ADDRESS OF SJB IN GPR3
BZ    FREEKEEP            BRANCH IF UNABLE TO LOCATE SJB
USING SJB,R3               SPACE 1
FINDJOBN CLC   SMF3ØJBN,SJBJOBNM   TEST IF SAME JOB
BE    FILLJCT             BRANCH IF SO
DROP R15
ICM R3,15,SJBSJB
BNZ FINDJOBN
B FREEKEEP

SPACE 1
PPGERROR WTO 'OIR714I IEFACRT - ADDRESS OF KEEPSECT IS INVALID'
B CLAMQUIK

SPACE 1
FILLJCT L R3,SJBJCT
USING JCT,R3

L R15,KEEPEXCP
A R15,JCTXOUT
ST R15,KEEPEXCP

MVC PRNTCOPY,JCTCPYCT
MVC PRNTLNES,JCTLINES
MVC PUNCHCRD,JCTPUNCH
MVC CRDSREAD,JCTCARDS

EJECT

***********************************************************************
*                                                                      *
*        PRINT OUT FINAL COSTS                                        *
*                                                                      *
***********************************************************************

CALL ISDACTRT,ID=16

L R2,CALFACPU
L R3,RETOCOST

MVC COSTLINE,JCPUINFO
BAS PLNK,PRINCOST

L R2,KEEPEXCP
L R3,RETXCOST
LR R4,R3

MVC COSTLINE,JXCPINFO
BAS PLNK,PRINCOST

ICM R2,15,CRDSREAD

BZ NOREAD
L R3,RETICOST
AR R4,R3

MVC COSTLINE,JIPTINFO
BAS PLNK,PRINCOST

ICM R2,15,PUNCHCRD

BZ NOPUNCH
L R3,RETCCOST

NOREAD DS ØH

ICM R2,15,PUNCHCRD

BZ NOPUNCH
L R3,RETCCOST

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
AR    R4,R3               TALLY COSTS
MVC   COSTLINE,JPUNINFO   FORMAT
BAS   PLNK,PRINCOST       GO PRINT COST
SPACE 1

NOPUNCH DS 0H
ICM   R2,15,PRNTLNES      GET # OF LINES GENERATED
BZ    NOPRINT             IF NONE - SKIP NEXT
L     R3,RETLCOST         GET COST OF LINES
AR    R4,R3               TALLY COSTS
MVC   COSTLINE,JPRTINFO   FORMAT
BAS   PLNK,PRINCOST       PRINT THE LINE
SR    R2,R2               CLEAR DE TRASH
SPACE 1

NOPRINT DS 0H
ICM   R2,3,KEEPTPR        GET # OF SPECIFIC TAPE MOUNTS
BZ    NOSPEC              IF NONE - SKIP NEXT
L     R3,RETSCOST         COST OF SPECIFIC TAPE MOUNT
AR    R4,R3               TALLY COSTS
MVC   COSTLINE,JTAPETPR   FORMAT
BAS   PLNK,PRINCOST       PRINT THE LINE
SR    R2,R2               CLEAR DE TRASH
SPACE 1

NOSPEC DS 0H
ICM   R2,3,KEEPPTM        GET # OF NON-SPECIFIC TAPE MOUNTS
BZ    NONONSPEC           IF NONE - SKIP NEXT
L     R3,RETNCOST         COST OF NON-SPECIFIC TAPE MOUNT
AR    R4,R3               TALLY COSTS
MVC   COSTLINE,JTAPEPTM   FORMAT
BAS   PLNK,PRINCOST       PRINT THE LINE
SR    R2,R2               CLEAR DE TRASH
SPACE 1

NONONSPEC DS 0H
ICM   R2,3,KEEPUSCT       GET # OF TAPE DEVICES USED
BZ    DOCIAO              IF NONE - SKIP NEXT
SR    R3,R3               COST OF USING A TAPE DEVICE
AR    R4,R3               TALLY COSTS
MVC   COSTLINE,JTAPEUSE   FORMAT
BAS   PLNK,PRINCOST       PRINT THE LINE
SPACE 1

DOCIAO  L     R3,RETCOST          CRU COST
AR    R3,R4               TALLY COSTS(CPU+EXCPS+MOUNTS)
MVC   COSTLINE,JCRUINFO   FORMAT
BAS   PLNK,CMRCOST        PRINT
EJECT
*
GIVE HASP THE FINAL COST
SPACE 1

FREEKEEP MVC   MSG+1(79),MSG       LINE OF *S
BAS   PLNK,PRINTER       ADD TO END OF SMBS
SPACE 1
L     R6,ADDRExD         FETCH ADDRESS OF EXD
USING EXDDSECT,R6 PROVIDE REFERENCE POINT
L R6,EXDCOMTB FETCH ADDRESS OF COMMON EXIT TABLE
USING JMR,R6 PROVIDE REFERENCE POINT
XC JMRUCOM,JMRUCOM CLEAR ADDRESS OF KEEPSECT
DROP R6 BACK INTO FOG
SPACE 1
* FREE ACQUIRED STORAGE, AND QUIT
SPACE 1
L R8,SAVELAST FETCH POINTER TO PREVIOUS SAVE AREA
SPACE 1
STORAGE RELEASE,LENGTH=KEEPLEN+WORKLEN,ADDR=(KEEP),SP=KEEPSP
SPACE 1
LR R13,R8 RESTORE POINTER TO FORMER SAVE AREA
B CLAMQUIK QUIT
EJECT
***********************************************************************
*                                                                     *
*        THIS ROUTINE IS ENTERED FOR STEP TERMINATION                 *
*                                                                     *
***********************************************************************
SPACE 1
STEPEND DS ØH
SR RØ,RØ CLEAR TRASH
C RØ,KEEPBMP TEST FOR REENTRY
BNE DEPART BRANCH IF SO
SPACE 1
ST SMF,KEEPBMP SAVE POINTER TO FIRST SMF BUFFER
SPACE 1
BAS PLNK,PRINTHDR STARS≥ BY GOLLY.
EJECT
***********************************************************************
*                                                                     *
*        PROCESS IDENTIFICATION SECTION                               *
*                                                                     *
***********************************************************************
SPACE 1
MVC HJOBNAME,CJOBNAME MOVE TITLES TO OUTPUT AREA
MVC HSTEPNAM,CSTEPNAM
MVC HSTEPNUM,CSTEPNUM
MVC HPGMNAME,CPGMNAME
MVC HSTART,CSTART
PLINE , PRINT STEP TITLES
SPACE 1
L R4,SMF3ØIOF OFFSET TO IDENTIFICATION
AR R4,SMF COMPUTE ID SECTION ADDRESS
USING SMF3ØID,R4 SET ID SECTION ADDRESSABILITY
SPACE 1
MVC HJOBNAME,SMF3ØJBN MOVE JOB NAME,
MVC HSTEPNAM,SMF3ØSTM STEP NAME,
MVC HPGMNAME,SMF30PGM PROGRAM NAME, ETC TO OUTPUT
MVC HSTEPNUM-1(4),PATSTEP# PATTERN FOR STEP NUMBER
LH R1,SMF30STN RETRIEVE STEP NUMBER
CVD R1,DOUBLE CONVERT TO PACKED DECIMAL
ED HSTEPNUM-1(4),DOUBLE+6 STEP NUMBER TO EBCDIC
SPACE 1
TM SMF30STN+3,X'0F' TEST IF SELECTION DATE IS VALID
BNO CMG1SKP BRANCH IF NOT
SPACE 1
L R14,SMF30STD DATE INITIATOR SELECTED STEP
BAS R8,CMRGREG CONVERT IT
MVC HSDATE,CLAMHOLD AND MOVE IT INTO OUTPUT AREA
SPACE 1
L R14,TERMDATE DATE RECORD MOVED TO SMF BUFFER
BAS R8,CMRGREG CONVERT IT
MVC HEDATE,CLAMHOLD AND MOVE IT INTO OUTPUT AREA
SPACE 1
L R15,SMF30SIT TIME SELECTED BY INITIATOR
BAS R8,CONVERT CONVERT TIME FROM 1/100 SEC TO HHMMS
MVC HSTIME,CLAMWORK START TIME
SPACE 1
L R15,TERMTIME FETCH END TIME
BAS R8,CONVERT CONVERT TIME FROM 1/100 SEC TO HHMMS
MVC HETIME,CLAMWORK END TIME
SPACE 1
BAS R8,CMGILL COMPUTE ELAPSED TIME
B CMG1 SUCCESSFUL RETURN
EJECT
CMG1SKP MVC HJEPTIME,CMRUNOME CANNOT COMPUTE VALUES
B CMG1PUT INFORM WHOMSOEVER IS CONCERNED
SPACE 1
CMG1 BAS R8,CONVERT CONVERT TIME FROM 1/100 SEC TO HHMMS
SPACE 1
CMG1PUT PLINE , PLINE ,
EJECT
***********************************************************************
* PROCESS PROCESSOR ACCOUNTING SECTION *
***********************************************************************
SPACE 1
L R4,SMF30COF OFFSET TO PROCESSOR ACCT SECTION
AR R4,SMF COMPUTE ID SECTION ADDRESS
USING SMF30CAS,R4 SET PROCESSOR SECTION ADDRESSABILITY
SPACE 1
MVC HELAPSED,CELAPSED MOVE CONSTANTS INTO OUTPUT AREA
MVC HCPUTIME,CCPUTIME
MVC HSRBTIME,CSRBTIME
SPACE 1
MVC HTIMELAP,CLAMWORK ELAPSED TIME 
L R15,SMF3ØCPT STEP TIME UNDER TCB IN HUNDRETHS 
BAS R8,CONVERT FORMAT TIME 
MVC HTIMECPU,CLAMWORK AND PLACE IT INTO OUTPUT AREA 
SPACE 1 
L R15,SMF3ØCPS STEP TIME UNDER SRB IN HUNDRETHS 
BAS R8,CONVERT FORMAT TIME 
MVC HTIMESRB,CLAMWORK AND PLACE IT INTO OUTPUT AREA 
PLINE , 
EJECT 
*********************************************************************** 
*                                                                     * 
*        PROCESS PERFORMANCE SECTION                                  * 
*                                                                     * 
*********************************************************************** 
SPACE 1 
L R4,SMF3ØPOF OFFSET TO PERFORMANCE SECTION 
AR R4,SMF COMPUTE PERFORMANCE SECTION ADDRESS 
USING SMF3ØPRF,R4 SET PERFORMANCE SECTN ADDRESSABILITY 
SPACE 1 
MVC HSCPU,PATSERV MOVE PERFORMANCE EDIT PATTERNS 
MVC HSSRB,PATSERV INTO OUTPUT AREA 
MVC HSIO,PATSERV 
MVC HSMSO,PATSERV 
SPACE 1 
MVC HSERVICE,CSERVICE MOVE PERFORMANCE CONSTANTS 
MVC HSRBSERV,CSRBSERV INTO OUTPUT AREA 
MVC HIOSERV,CIOSERV 
MVC HMSOSERV,CMSOSERV 
SPACE 1 
PLINE , 
EJECT 

Editor’s note: this article will be continued in the next issue.

Systems Programmer (USA) © Xephon 1999
Checking FTP success

The FTP utility can be used in batch mode. The only problem you get is that it always gives you a return code of 0 (zero), so you can’t check by using $? whether the FTP went wrong or not.

I developed some code that checks the output of an FTP. In our company we only use PUT or APPEND files in an FTP, we don’t use the ‘GET’ statement, because we don’t know whether the file(s) is/are ready to send.

```bash
# Name Script   : /home/oper/check_ftp
# Last change   : 13-04-95  pst creation
# Description   : check if ftp was ok
#--------------------------------------------------------------------------------
if [ $# -ne 3 ]
then
    echo "ntestftp WRONG PARAMETER"
    echo "The next parameters are required for check_ftp"
    echo "- ftp output, to check"
    echo "- destination computer"
    echo "- project"
    echo "Example: check_ftp /tmp/aqlftp afts5 aql"
    exit 1
fi
RETC=0
CHECK_VALUE=1
SAVE_DSN=""
# Read file, look for put or append + adjusting message: ...transfer
# complete
# if no 226 Transfer complete (AIX) or
# 250 TRANSFER COMPLETE (OS/390): generate errormessage
while read n1 DSN DEST_FILE
do
    if [ "$n1" = "put" -o "$n1" = "append" ]
    then
        if [ $CHECK_VALUE -eq 0 ]
        then
            logmsg $0 "$SAVE_DSN has been sent to $2 as $SAVE_DEST" 3
        else
            if [ $CHECK_VALUE -ne 1 ]
            then
                logmsg $0 "$SAVE_DSN has NOT been sent to $2" 3
                RETC=1
            fi
        fi
        SAVE_DSN=""
    fi
    if [ -z "$DEST_FILE" ]
    then
        SAVE_DESTFILE=""
    else
        SAVE_DESTFILE="named $DEST_FILE"
    fi
```

CHECK_VALUE=226
else if [ "$n1" = "$CHECK_VALUE" -o "$n1 $DSN" = "250 TRANSFER" ]
then CHECK_VALUE=Ø
fi
fi
done < $1

# Check if there are files sent:
if [ "$SAVE_DSN" = "" ]
then logmsg $Ø "No files were sent" $3
exit 1
fi

# Generate a message for the last file that was or wasn't sent:
if [ $CHECK_VALUE -eq Ø ]
then logmsg $Ø "$SAVE_DSN has been sent to $2 $SAVE_DESTFILE" $3
else if [ $CHECK_VALUE -ne 1 ]
then logmsg $Ø "\Ø$SAVE_DSN has NOT been sent to $2" $3
RETC=1
fi
fi
exit $RETC

example script:
ftp -v afts1.aft.schuitema 1>/tmp/ftp_to_afts1 2>&1
# check the result of the ftp
testftp /tmp/ftp_to_afts1 afts1 project_ftp
RETC=$?
if [ "$RETC" -ne Ø ]
then echo "project_ftp: FTP to afts1 went wrong"
   echo "\nInformation follows: "
cat /tmp/ftp_to_afts1
else echo "\nFTP to afts1 was OK"
fi
rm /tmp/ftp_to_afts1

EXAMPLE OUTPUT

All FTPs OK

28/12/98 01:27:30 ftp99ØØ  testftp - project_ftp - /ftp/send/
B1Ø7Ø7.dmp.Z has been sent to afts1.aft.schuitema as /ftp/receive/
site7.dmp.Z
28/12/98 01:27:30 ftp99ØØ  testftp - project_ftp - /ftp/send/
B1Ø7Ø7.explog has been sent to afts1.aft.schuitema as /ftp/receive/
site7.explog
28/12/98 01:27:30 ftp99ØØ  testftp - project_ftp - /ftp/send/
bai_afts3_Ø7.cmd has been sent to afts1.aft.schuitema as /ftp/receive/

46 © 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
FTP to afts1.aft.schuitema was OK

One FTP went wrong

28/12/98 01:27:30 ftp9900 testftp - ftp - /ftp/send/
B10707.explog has been sent to afts1.aft.schuitema as /ftp/receive/
site7.explog
28/12/98 01:27:30 ftp9900 testftp - ftp - /ftp/send/
B10707.dmp.Z has NOT been sent to afts1.aft.schuitema

Information follows:
Connected to afts1.aft.schuitema.
331 Password required for root.
230 User root logged in.
type binary
200 Type set to I.
put /ftp/send/B10707.explog /ftp/receive/site7.explog
200 PORT command successful.
150 Opening data connection for /ftp/receive/site7.explog.
226 Transfer complete.
266 bytes sent in 0.01819 seconds (14.28 Kbytes/s)
local: /ftp/send/B10701.explog remote: /ftp/receive/site7.explog
200 PORT command successful.
150 Opening data connection for /ftp/receive/site7.dmp.
netout: Invalid argument
452 Error writing file: No space left on device
quit

All FTPs went wrong

27/12/98 07:32:53 29312 root testftp - project_ftp - No files were sent
project_ftp: FTP to afts1 went wrong

Information follows:
Name (afts1.aft.schuitema:root): 331 Password required for root.
530 Login incorrect.
Login failed.
221 Goodbye.

Teun Post
Unix Specialist
Schuitema NV (The Netherlands)
© Xephon 1999
A mailbox system for SMTP under MVS TCP/IP – 5

In this issue we continue the code for the implementation of a mailbox system for SMTP, based on ISPF functions.

```plaintext
SET &LENNAME = &LENGTH(&STR(&SYSNSUB(1,&NAME)))
SET &LENRE = &LENGTH(&STR(RE:))
SET &LENREB = &LENGTH(&STR(RE: ))
IF &LENNAME > &LENREB THEN DO
  IF &SYSCAPS(&SUBSTR(1:&LENREB,&STR(&SYSNSUB(1,&NAME)))) = + &STR(RE: ) THEN DO
    SET &NAME = &SUBSTR(&LENREB+1:&LENNAME,&STR(&SYSNSUB(1,&NAME)))
    SET &LENNAME = &LENGTH(&STR(&SYSNSUB(1,&NAME)))
  END
END
IF &LENNAME > &LENRE THEN DO
  IF &SYSCAPS(&SUBSTR(1:&LENRE,&STR(&SYSNSUB(1,&NAME)))) = + &STR(RE:) THEN DO
    SET &NAME = &SUBSTR(&LENRE+1:&LENNAME,&STR(&SYSNSUB(1,&NAME)))
    SET &LENNAME = &LENGTH(&STR(&SYSNSUB(1,&NAME)))
  END
END
/* MAKE DOUBLE ASSIGNMENTS OF MEMBER NAME TO CATER FOR INVALID CHAR*/
/* AND AVOID USING &STR AND &NRSTR IN THESE ASSIGNMENTS TO GET */
/* CORRECT TRUNCATION, BUT USE &SYSNSUB. */
/* A RETCODE OF 9ØØ AND OTHERS CAN OCCUR BUT IS ACCEPTABLE. */
SET &EDMEM = &SYSNSUB(1,&NAME)
SET &EDMEM = &EDMEM
SET &EDMEM = &EDMEM
/* REMOVE NUMERIC MEMBER PREFIX */
SET &LENMEM = &LENGTH(&STR(&SYSNSUB(1,&EDMEM)))
SET &MEMPREF = &SUBSTR(1:1,&STR(&SYSNSUB(1,&EDMEM)))
SET &Q = 1 /* ALLOW AT LEAST ONE IN LENGTH OF MEMBER NAME */
DO WHILE &Q < &LENMEM AND + &DATATYPE(&STR(&SYSNSUB(1,&MEMPREF))) = NUM
  SET &Q = &Q + 1
  IF &DATATYPE(&STR(&SYSNSUB(1,&MEMPREF))) = NUM THEN DO
    SET &EDMEM = &SUBSTR(2:&LENMEM,&STR(&SYSNSUB(1,&EDMEM)))
    SET &LENMEM = &LENGTH(&STR(&SYSNSUB(1,&EDMEM)))
    SET &MEMPREF = &SUBSTR(1:1,&STR(&SYSNSUB(1,&EDMEM)))
    SET &NAME = &SUBSTR(1:&LENMEM,&STR(&SYSNSUB(1,&EDMEM)))
    SET &LENNAME = &LENGTH(&STR(&SYSNSUB(1,&NAME)))
END
END
IF &DATATYPE(&SUBSTR(1:1,&STR(&SYSNSUB(1,&MEMPREF)))) = NUM + THEN DO
  ISPEXEC CONTROL DISPLAY LINE START(14)
  SLEEP 1
  WRITE =====> Numeric Member Name &STR(&SYSNSUB(1,&EDMEM)) +
```

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
substituted (&SYSICMD).
IF &LENNAME <= 1 THEN DO
  SET &NAME = SUBJECT
END
ELSE DO
  SET &NAME = &STR(S)&SUBSTR(2:&LENMEM,&STR(&SYSNSUB(1,&EDMEM)))
END
SET &LENNAME = &LENGTH(&STR(&SYSNSUB(1,&NAME)))
SET &N = &LENMEM
END
SET &MEMRET = Ø
SET &N = Ø
DO WHILE &N < &LENMEM
  SET &N = &N + 1
  SET &MEMCHAR = &SUBSTR(&N:&N,&STR(&SYSNSUB(1,&EDMEM)))
  SET &CHARHIT = +
  &SYSINDEX(&STR(&SYSNSUB(1,&MEMCHAR)),+
  ABCDEFGHIJKLMNOPQRSTUVWXYZ$#@Ø123456789)
  IF &CHARHIT = Ø THEN DO
    SET &SUBNAME = &SUBSTR(1:&N-1,&STR(&SYSNSUB(1,&EDMEM)))
    SET &NAME = &SUBSTR(1:&N-1,&STR(&SYSNSUB(1,&EDMEM)))
    SET &NAME = &STR(&SYSNSUB(1,&SUBNAME))
    &EDMEM = &STR(&SYSNSUB(1,&NAME))
    SET &LENNAME = &LENGTH(&STR(&SYSNSUB(1,&NAME)))
    SET &EDMEM = &STR(&SYSNSUB(1,&NAME))
    SET &LENMEM = &LENGTH(&STR(&SYSNSUB(1,&EDMEM)))
  IF &N < &LENMEM THEN DO
    SET &N = &N - 1
  END
  SET &MEMPREF = &SUBSTR(1:1,&STR(&SYSNSUB(1,&EDMEM)))
  SET &Q = 1 /* ALLOW AT LEAST ONE IN LENGTH OF MEMBER NAME */
  DO WHILE &Q < &LENMEM AND +
    &DATATYPE(&STR(&SYSNSUB(1,&MEMPREF))) = NUM
    SET &Q = &Q + 1
  IF &DATATYPE(&STR(&SYSNSUB(1,&MEMPREF))) = NUM THEN DO
    SET &EDMEM = &SUBSTR(2:&LENMEM,&STR(&SYSNSUB(1,&EDMEM)))
    SET &LENMEM = &LENGTH(&STR(&SYSNSUB(1,&EDMEM)))
    SET &MEMPREF = &SUBSTR(1:1,&STR(&SYSNSUB(1,&EDMEM)))
    SET &NAME = &SUBSTR(1:&LENMEM,&STR(&SYSNSUB(1,&EDMEM)))
    SET &LENNAME = &LENGTH(&STR(&SYSNSUB(1,&NAME)))
  END
END
IF &DATATYPE(&SUBSTR(1:1,&STR(&SYSNSUB(1,&MEMPREF)))) = NUM + THEN DO
  ISPSEXEC CONTROL DISPLAY LINE START(14)
  SLEEP 1
  WRITE ======> Numeric Member Name &STR(&SYSNSUB(1,&EDMEM)) + substituted (&SYSICMD).
  IF &LENNAME <= 1 THEN DO
    SET &NAME = SUBJECT
  END
ELSE DO
    SET &NAME = &STR(S)&SUBSTR(2:&LENMEM,&STR(&SYSNSUB(1,&EDMEM)))
END
SET &LENNAME = &LENGTH(&STR(&SYSNSUB(1,&NAME)))
SET &N = &LENMEM
END
END
END
IF &LENGTH(&STR(&SYSNSUB(1,&NAME))) > 6 THEN DO
    SET &NAME = &SUBSTR(1:6,&STR(&SYSNSUB(1,&NAME)))
    SET &LENNAME = &LENGTH(&STR(&SYSNSUB(1,&NAME)))
END
SET &SHNAME = &STR(&NAME)
SET &SUF1 = Ø
SET &SUF2 = Ø
SET &NAME = &STR(&SHNAME)&SUF1&SUF2
ISREDIT CURSOR = &EVAL(&JOBBEGIN + 1) 1
ISREDIT (JOBEND) = LINENUM .ZLAST
ISREDIT LABEL &JOBEND = .E Ø
SET &NOMORE = NO
SET &ENDJOB = NO
SET &MAXCNT5 = 512 /* SET LOOP CONTROL TO MAX 512 SCANS */
SET &CNT5 = Ø
DO WHILE &NOMORE NE YES AND &ENDJOB NE YES AND &CNT5 < &MAXCNT5
    SET &CNT5 = &CNT5 + 1
    IF &CNT5 = &MAXCNT5 THEN DO
        WRITE ====> Error, loop on CNT5 terminated (&SYSICMD).
    END
SET &MSGRET = 4
ISREDIT (MSGMSGL,MSGMSGR) = CURSOR
SET &MSGMSGLM = 2**24-1
SET &RET = Ø
ISREDIT SEEK ' ** MESSAGE ** '
SET &MSGRETM = &RET
IF &MSGRETM = Ø THEN DO
    ISREDIT (MSGMSGLM,MSGMSGRM) = CURSOR
END
SET &MSGMSGLA = 2**24-1
ISREDIT CURSOR = &MSGMSGL Ø
SET &RET = Ø
ISREDIT SEEK ' ** ACKNOWLEDGMENT ** '
SET &MSGRETA = &RET
IF &MSGRETA = Ø THEN DO
    ISREDIT (MSGMSGLA,MSGMGR) = CURSOR
END
SET &MSGMSGLR = 2**24-1
ISREDIT CURSOR = &MSGMSGL Ø
IF &MSGRETM > Ø AND &MSGRETA > Ø THEN DO
    ISREDIT CURSOR = &MSGMSGL Ø
END
SET &RET = Ø
ISREDIT SEEK 'RECEIVE' 1
SET &MSGRETR = &RET
IF &MSGRETR = Ø THEN DO
  ISREDIT (MSGMSGLR, MSGMSGRR) = CURSOR
END
IF &MSGRETM = Ø OR &MSGRETA = Ø OR &MSGRETR = Ø THEN DO
  SET &MSGRET = Ø
  SET &MSGMSGL = &MSGMSGLM
  SET &MSGMSGR = &MSGMSGRM
  IF &MSGMSGL > &MSGMSGLA THEN DO
    SET &MSGMSGL = &MSGMSGLA
    SET &MSGMSGR = &MSGMSGRA
  END
  IF &MSGMSGL > &MSGMSGLR THEN DO
    SET &MSGMSGL = &MSGMSGLR
    SET &MSGMSGR = 4
  END
  ISREDIT CURSOR = &MSGMSGL &MSGMSGR
END
IF &MSGRET > Ø THEN DO
  SET &NOMORE = YES
  SET &CNT1 = Ø
  SET &MAXCNT1 = &EVAL(39*39)
  SET &CRERET = 4
  DO WHILE &CRERET > Ø AND &CNT1 < &MAXCNT1
    SET &CNT1 = &CNT1 + 1
    SET &RET = Ø
    ISREDIT CREATE &STR(&NAME) .B .E
    SET &CRERET = &RET
  IF &CRERET = Ø THEN DO
    IF &FUNCTION NE MAIL THEN DO
      ISPEXEC CONTROL DISPLAY LINE START(14)
      SLEEP 1
      WRITE ===> Created &STR(&TOPREF..&LOGDS(&NAME)) from + &STR(&ORGNAME)
      IF &FUNCTION NE MAIL AND &SYSENV NE FORE THEN DO
        SE 'Mail &STR(&ORGNAME) received to + &STR(&TOPREF..&LOGDS(&NAME))' LOGON USER(&TOPREF)
      END
    ELSE DO
      SET &ZEDLMSG = + &STR(====> Mail saved to &TOPREF..&LOGDS(&NAME) <====)
      WRITE &STR(&ZEDLMSG)
    END
  ELSE DO
    SET &ZEDLMSG = + &STR(-----> Mail saved to &TOPREF..&LOGDS(&NAME) <-----)
    WRITE &STR(&ZEDLMSG)
  END
  ISPEXEC LMINIT DATAID(DID) + DATASET(&STR('&TOPREF..&LOGDS')) ENQ(SHRW)
  ISPEXEC LMMSTATS DATAID(&DID) MEMBER(&NRSTR(&NAME)) + USER(&USER)
  ISPEXEC LMFREE DATAID(&DID)
END
IF &FUNCTION NE MAIL THEN DO
IF &SYSENV = FORE THEN DO
SLEEP 1
TSOLINE1 /* Clear screen */
%EDITRECV /* take edit recovery */
IF &SYSCAPS(&STR(&DEBUG)) = DEBUG THEN DO
WRITE ======> Reentering &SYSICMD <======
END
ISPEXEC EDIT DATASET('&TOPREF..&LOGDS(&NAME)') MACRO(%ZSMTPI)
IF &SYSCAPS(&STR(&DEBUG)) = DEBUG THEN DO
WRITE ======> Reentering &SYSICMD <======
END
END
ELSE DO
SET &STARTSPC = 1
SET &RC = &STR(Ø123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ$#@)
SET &LOCSPC = &SYSINDEX(&STR(&SUF1),&STR(&RC),&STARTSPC)
IF &LOCSPC = Ø THEN DO
ISPEXEC CONTROL DISPLAY LINE START(14)
SLEEP 1
WRITE ===== ErrorØ1 creating &STR(&NAME) +
from &STR(&ORGNAME) rc &CRERET (&SYSICMD).
SET &ZSMTPR = &STR(ERROR)
SET &CNT1 = &MAXCNT1
END
SET &RET = Ø
SET &SUF1 = &SUBSTR(&LOCSPC+1:&LOCSPC+1,&STR(&RC))
IF &RET = Ø THEN DO
SET &NAME = &STR(&SHNAME)&SUF1&SUF2
END
ELSE DO
SET &SUF1 = Ø
SET &STARTSPC = 1
SET &LOCSPC = &SYSINDEX(&STR(&SUF2),&STR(&RC),&STARTSPC)
IF &LOCSPC = Ø THEN DO
ISPEXEC CONTROL DISPLAY LINE START(14)
SLEEP 1
WRITE =====> ErrorØ2 creating &STR(&NAME) +
from &STR(&ORGNAME) rc &CRERET (&SYSICMD).
SET &ZSMTPR = &STR(ERROR)
SET &CNT1 = &MAXCNT1
END
SET &RET = Ø
SET &SUF2 = &SUBSTR(&LOCSPC+1:&LOCSPC+1,&STR(&RC))
IF &RET = Ø THEN DO
SET &NAME = &STR(&SHNAME)&SUF1&SUF2
END
ELSE DO
ISPEXEC CONTROL DISPLAY LINE START(14)
SLEEP 1
WRITE =====> ErrorØ3 creating &STR(&NAME) + from &STR(&ORGNAME) rc &CRERET (&SYSICMD).
SET &ZSMTPR = &STR(ERROR)
SET &CNT1 = &MAXCNT1
END
END
ISREDIT (JOBEND) = LINENUM .ZLAST
END
ELSE DO
ISREDIT (JOBEND,JOBEROW) = CURSOR
ISREDIT (JOBREC) = LINE &STR(&SYSNSUB(1,&JOBEND))
SET &RET = Ø
ISREDIT LABEL &JOBEND = .ASX 1
SET &LABRET = &RET
ISREDIT CURSOR = &JOBEND Ø
SET &RET = Ø
ISREDIT SEEK RECEIVE WORD .ASX .ASX
SET &SEEKRET = &RET
IF (&SEEKRET = Ø OR &LABRET > 8 ) AND &FUNCTION NE MAIL THEN DO
IF &JOBEROW > 3 THEN DO
SET &ENDJOB = YES
SET &RET = Ø
ISREDIT LABEL &EVAL(&JOBEND - 1) = .E Ø
IF &RET > 8 THEN DO
ISPEXEC CONTROL DISPLAY LINE START(14)
SLEEP 1
WRITE =====> Error12 setting label for Jobend (&SYSICMD).
SET &ZSMTPR = &STR(SEVERROR)
SET &RET = Ø
SYSCALL FREERSC &FUNCTION &ZSMTPR DEBUG(&STR(&DEBUG))
/* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
SET &RET = &LASTCC
EXIT CODE(1)
END
SET &CNT1 = Ø
SET &MAXCNT1 = &EVAL(39*39)
SET &CRERET = 4
DO WHILE &CRERET > Ø AND &CNT1 < &MAXCNT1
SET &CNT1 = &CNT1 + 1
SET &RET = Ø
ISREDIT CREATE &STR(&NAME) .B .E
SET &CRERET = &RET
IF &CRERET = Ø THEN DO
IF &FUNCTION NE MAIL THEN DO
ISPEXEC CONTROL DISPLAY LINE START(14)
SLEEP 1
WRITE =====> Created &STR(&TOPREF..&LOGDS(&NAME)) from + &STR(&ORGNAME)
IF &FUNCTION NE MAIL AND &SYSENV NE FORE THEN DO
SE 'Mail &STR(&ORGNAME) received to +
&STR(&TOPREF..&LOGDS(&NAME)): LOGON USER(&TOPREF)
END
END
ELSE DO
SET &ZEDLMSG = +
&STR(====> Mail saved to &TOPREF..&LOGDS(&NAME) <====)
WRITE &STR(&ZEDLMSG)
END
ISPEXEC LMINIT DATAID(DID) +
DATASET(&STR('&TOPREF..&LOGDS')) ENQ(SHRW)
ISPEXEC LMMSTATS DATAID(&DID) MEMBER(&NRSTR(&NAME)) +
USER(&USER)
ISPEXEC LMFREE DATAID(&DID)
IF &FUNCTION NE MAIL THEN DO
IF &SYSENV = FORE THEN DO
SLEEP 1
TSOLINE1 /* Clear screen */
%EDITRECV /* take edit recovery */
IF &SYSCAPS(&STR(&DEBUG)) = DEBUG THEN DO
WRITE ======> Reentering &SYSICMD <======
END
ISPEXEC EDIT DATASET('&TOPREF..&LOGDS(&NAME)') MACRO(%ZSMTPI)
IF &SYSCAPS(&STR(&DEBUG)) = DEBUG THEN DO
WRITE ======> Reentering &SYSICMD <======
END
END
ELSE DO
SET &STARTSPC = 1
SET &RC = &STR(Ø123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ$#@)
SET &LOCSPC = &SYSINDEX(&STR(&SUF1),&STR(&RC),&STARTSPC)
IF &LOCSPC = Ø THEN DO
ISPEXEC CONTROL DISPLAY LINE START(14)
SLEEP 1
WRITE =====> ErrorØ5 creating &STR(&NAME) +
from &STR(&ORGNAME) rc &CRERET (&SYSICMD).
SET &ZSMTPR = &STR(ERROR)
SET &CNT1 = &MAXCNT1
END
SET &RET = Ø
SET &SUF1 = &SUBSTR(&LOCSPC+1:&LOCSPC+1,&STR(&RC))
IF &RET = Ø THEN DO
SET &NAME = &STR(&SHNAME)&SUF1&SUF2
END
ELSE DO
SET &SUF1 = Ø
SET &STARTSPC = 1
SET &LOCSPC = &SYSINDEX(&STR(&SUF2),&STR(&RC),&STARTSPC)
IF &LOCSPC = Ø THEN DO
ISPEXEC CONTROL DISPLAY LINE START(14)
SLEEP 1
WRITE =====> ErrorØ6 creating &STR(&NAME) +
from &STR(&ORGNAME) rc &CRERET (&SYSICMD).
SET &ZSMTPR = &STR(ERROR)
SET &CNT1 = &MAXCNT1
END
SET &RET = Ø
SET &SUF2 = &SUBSTR(&LOCSPC+1:&LOCSPC+1,&STR(&RC))
IF &RET = Ø THEN DO
    SET &NAME = &STR(&SHNAME)&SUF1&SUF2
END
ELSE DO
    ISPEXEC CONTROL DISPLAY LINE START(14)
    SLEEP 1
    WRITE =====> ErrorØ7 creating &STR(&NAME) +
    from &STR(&ORGNAME) rc &CRERET (&SYSICMD).
    SET &ZSMTPR = &STR(ERROR)
    SET &CNT1 = &MAXCNT1
END
END
SET &BEGJOB = NO
ISREDIT CURSOR = &JOBEND 1
SET &JOBBEGIN = &JOBEND
END
ELSE DO
    ISREDIT CURSOR = &EVAL(&JOBEND + 1) 1
END
END
ELSE DO
END
ELSE DO
END
ELSE DO
    ISREDIT CURSOR = &EVAL(&JOBBEGIN + 1) 1
END
END
SET &RET = Ø
SYSCALL FREERSC &FUNCTION &ZSMTPR DEBUG(&STR(&DEBUG))
/* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
SET &RET = &LASTCC
EXIT CODE(1)
/*
*/
/*
*/
/* INLINE SUBROUTINES
*/
/*
*/
FREERSC: +
PROC 2 FUNCTION ZSMTPR DEBUG(NEBUG)
/* */
/* INLINE RETURN ROUTINE TO FREE EVT ALLOCATED RESOURCES */
/* */
CONTROL NOMSG NOFLUSH NOCONLIST NOSYMLIST NOCAPS
ATTN DO
SET &FLUSH = FLUSH           /* NEXT STATEMENT MUST BE NULL LINE */
END
ERROR DO
SET &RET = &LASTCC
RETURN
END
IF &SYSCAPS(&STR(&DEBUG)) NE DEBUG THEN DO
ISPEXEC VGET (DEBUG)
END
IF &STR(&DEBUG) = &STR() THEN DO
SET &DEBUG = NEBUG
END
IF &SYSCAPS(&STR(&DEBUG)) = DEBUG THEN DO
CONTROL MSG NOFLUSH LIST CONLIST SYMLIST
END
IF &FLUSH = FLUSH THEN DO
SET &ZEDSMSG = &str(Function interrupted)
ISPEXEC SETMSG MSG(ISRZØØ1)
RETURN CODE(Ø)
END
ISPEXEC CONTROL ERRORS RETURN
ISREDIT RESET
ISREDIT RESET LABEL
ISPEXEC VPUT (ZSMTPR)
IF &FUNCTION = MAIL THEN DO
ISREDIT END
END
IF &FUNCTION NE MAIL AND &ZSMTPR NE &STR(OK) THEN DO
SET &DATA = +
&STR('All incoming mail is shown here but will also be saved as')
ISREDIT LINE_BEFORE .ZF = MSGLINE MASKLINE + &DATA
SET &DATA = +
&STR('separate members with names derived from subject.')
ISREDIT LINE_BEFORE .ZF = MSGLINE MASKLINE + &DATA
SET &DATA = +
&STR()
ISREDIT LINE_BEFORE .ZF = MSGLINE MASKLINE + &DATA
ISREDIT UP MAX
END
IF &FUNCTION NE MAIL AND &ZSMTPR = &STR(OK) THEN DO
ISREDIT END
END
RETURN CODE(Ø)
END
/* */
The CLIST (edit macro) ZSMTPN sets the ISPF note lines in the constructed mail informing the sender of the possible options before sending the mail and automatically setting the reply to receivers when the mail being constructed is a reply.

```/*
/* ZSMTPN
/* Edit macro to set edit messages in smtp-note and set receivers for
/* Reply.
/* Called from MAILSENS
/*
/* SUBROUTINES/EDIT MACROES:
/* %ZCC
/*
/* Utilities used:
/* TSOLINE1
/* */
PROC Ø DEBUG(nDEBUG)
CONTROL NOMSG NOFLUSH NOLIST NOCONLIST NOSYMLIST NOCAPS
ERROR DO
SET &RET = &LASTCC
RETURN
END
IF &SYSCAPS(&STR(&DEBUG)) = DEBUG THEN DO
CONTROL MSG NOFLUSH LIST CONLIST SYMLIST
WRITE ======> Entering &SYSICMD <======
END
IF &SYSISPF = &STR(NOT ACTIVE) THEN DO
WRITE ====> Sorry only executable under ISPF (&SYSICMD).
EXIT CODE(16)
END
IF &SYSNEST = NO THEN DO
ISREDIT MACRO PROCESS
END
ISPEXEC CONTROL ERRORS RETURN
ISPEXEC VGET (ZSCREEN)
ISREDIT (SAVE) = USER_STATE
ISREDIT (LRECL) = LRECL
ISPEXEC VGET (REPLY)
SET &ATSIGN = &STR(@)
SET &NETDLM = &STR(%)
IF &STR(&REPLY) = YES THEN DO
SET &DATA = +
&STR('                       ------- Reply mode -------')
ISREDIT LINE_BEFORE .ZF =  MSGLINE MASKLINE + &DATA
END
ELSE DO
SET &DATA = +
&STR('                      -------- Send mode --------')
ISREDIT LINE_BEFORE .ZF =  MSGLINE MASKLINE + &DATA
```
END
SET &DATA = +
&STR('To send mail when editing finished: hit END (PF3/PF15).')
ISREDIT LINE_BEFORE .ZF = MSGLINE MASKLINE + &DATA
SET &DATA = +
&STR('To send mail but stay in edit mode: enter ZREMAIL and hit + Enter.')
ISREDIT LINE_BEFORE .ZF = MSGLINE MASKLINE + &DATA
SET &DATA = +
&STR('To suppress and save or not-save mail: use ZNOMAIL or + ZNOMAIL CAN.')
ISREDIT LINE_BEFORE .ZF = MSGLINE MASKLINE + &DATA
SET &DATA = +
&STR('To enter CC-address use command ZCC evt followed by Receiver and + domain.')
ISREDIT LINE_BEFORE .ZF = MSGLINE MASKLINE + &DATA
ISREDIT CURSOR = 1 Ø
SET &RET = Ø
ISREDIT SEEK &STR('______ Reply Separator ')
IF &RET = Ø THEN DO
ISREDIT (SUBJR,SUBJC) = CURSOR
SET &SUBJR = &SUBJR - 1
SET &RSAFT = &SUBJR + 2
ISREDIT LABEL &SUBJR = .RS 1
ISREDIT LABEL &RSAFT = .RA 1
SET &RS = .RS
SET &RA = .RA
END
ELSE DO
SET &RS = .ZL
SET &RA = .ZL
END
ISPEXEC VPUT (RS) /* Info to ZCC */
IF &STR(&REPLY) = YES THEN DO
ISREDIT CURSOR = 1 Ø
SET &RET = Ø
ISREDIT SEEK &STR('______ Reply Separator ') &RA .ZL
IF &RET = Ø THEN DO
ISREDIT (NEXJR,NEXJC) = CURSOR
ISREDIT LABEL &NEXJR = .RR 1
SET &RR = .RR
END
ELSE DO
SET &RR = .ZL
END
SET &MAXRP = 256
ISPEXEC VPUT (MAXRP) /* Info to inline subroutine */
SET &RV = RV
SET &RP = Ø
DO WHILE &RP < &MAXRP
SET &RP = &RP + 1
ISPEXEC VGET (RV&RP)
SET &C = &STR(&SYSNSUB(2,&&RV&RP))
IF &STR(&SYSNSUB(1,&C)) NE &STR() THEN DO
  SET &&RV&RP = &STR()     /* clear variables */
  ISPEXEC VPUT (RV&RP)
END
ELSE DO
  SET &RP = &MAXRP
END

ELSE DO
  SET &&RV&RP = &STR()
  ISPEXEC VPUT (RV&RP)
END

ELSE DO
  SET &RP = &MAXRP
END

/*                                                                  */
SET &SEARCH = &STR(Reply-to:)
SET &RET = Ø
SYSCALL SCANFR &SEARCH &RA &RR &NETDLM &ATSIGN DEBUG(&STR(&DEBUG))
  /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
SET &RET = &LASTCC
SET &REPLYTOCC = &RET

/*                                                                  */
SET &SEARCH = &STR(Resent-reply-to:)
SET &RET = Ø
SYSCALL SCANFR &SEARCH &RA &RR &NETDLM &ATSIGN DEBUG(&STR(&DEBUG))
  /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
SET &RET = &LASTCC
SET &RESENTREPLYTOCC = &RET

/*                                                                  */
SET &FROMCC = Ø
IF &REPLYTOCC > Ø THEN DO
  SET &SEARCH = &STR(From:)
  SET &RET = Ø
  SYSCALL SCANFR &SEARCH &RA &RR &NETDLM &ATSIGN DEBUG(&STR(&DEBUG))
    /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
  SET &RET = &LASTCC
  SET &FROMCC = &RET
END

/*                                                                  */
SET &SEARCH = &STR(Resent-from:)
SET &RET = Ø
SYSCALL SCANFR &SEARCH &RA &RR &NETDLM &ATSIGN DEBUG(&STR(&DEBUG))
  /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
SET &RET = &LASTCC
SET &RESENTFROMCC = &RET

/*                                                                  */
IF &REPLYTOCC > Ø AND &FROMCC > Ø THEN DO
  SET &SEARCH = &STR(Sender:)
  SET &RET = Ø
  SYSCALL SCANFR &SEARCH &RA &RR &NETDLM &ATSIGN DEBUG(&STR(&DEBUG))
    /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
  SET &RET = &LASTCC
  SET &SENDERCC = &RET
END

/*                                                                  */
IF &REPLYTOCC > Ø AND &FROMCC > Ø THEN DO
  SET &SEARCH = &STR(Resent-sender:)
  SET &RET = Ø
  SYSCALL SCANFR &SEARCH &RA &RR &NETDLM &ATSIGN DEBUG(&STR(&DEBUG))
  /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
  SET &RET = &LASTCC
  SET &RESENTSENDERCC = &RET
END
/* */

SET &SEARCH = &STR(To:)
SET &RET = Ø
SYSCALL SCANFR &SEARCH &RA &RR &NETDLM &ATSIGN DEBUG(&STR(&DEBUG))
/* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
SET &RET = &LASTCC
SET &TOCC = &RET
/* */

SET &SEARCH = &STR(Cc:)
SET &RET = Ø
SYSCALL SCANFR &SEARCH &RA &RR &NETDLM &ATSIGN DEBUG(&STR(&DEBUG))
/* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
SET &RET = &LASTCC
SET &CCCC = &RET
/* */

SET &SEARCH = &STR(Bcc:)
SET &RET = Ø
SYSCALL SCANFR &SEARCH &RA &RR &NETDLM &ATSIGN DEBUG(&STR(&DEBUG))
/* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
SET &RET = &LASTCC
SET &BCCCC = &RET
/* */

ISREDIT CURSOR = 1 Ø
SET &RET = Ø
ISREDIT SEEK ' ** MESSAGE ** ' &RS .ZL
IF &RET = Ø THEN DO
  ISREDIT (JOBMSGL,JOBMSGR) = CURSOR
  ISREDIT DELETE &JOBMSGL /* REMOVE HEADER */
  ISREDIT DELETE &JOBMSGL /* REMOVE HEADER */
END
TSOLINE1 /* Clear screen */
END

ISREDIT CURSOR = 1 Ø
SET &RCPTTX = &STR('To:')
SET &RET = Ø
ISREDIT SEEK &RCPTTX 1 .ZF &RS
IF &RET = Ø THEN DO
  ISREDIT (TOBEGIN,TOFROW) = CURSOR
  SET &TOBEGIN = &TOBEGIN - 1
  ISREDIT LABEL &TOBEGIN = .TB 1
  ISREDIT EXCLUDE .ZF .TB P"" ALL
END
SET &DATA = +
&STR('Insert your data here')
ISREDIT LINE_Before &RS = MSGLINE MASKLINE + &DATA
ISREDIT (L) = LINENUM &RS
SET &COL = 1
ISREDIT INSERT &RS 10
ISREDIT CURSOR = (L, COL)
ISREDIT USER_STATE = (SAVE)
EXIT CODE(Ø) /* keep curser inline for insert, don't set ccl */
/* */
/* INLINE SUBROUTINES */
/* */
SCANFR: +
PROC 5 SEARCH RA RR NETDLM ATSIGN DEBUG(NEBUG)
/* */
/* INLINE ROUTINE TO LOCATE CC-RECEIVERS */
/* */
CONTROL NOMSG NOFLUSH NOLIST NOCONLIST NOSYMLIST NOCAPS
ATTN DO
SET &FLUSH = FLUSH /* NEXT STATEMENT MUST BE NULL LINE */
END
ERROR DO
SET &RET = &LASTCC
RETURN
END
IF &SYSCAPS(&STR(&DEBUG)) NE DEBUG THEN DO
ISPEXEC VGET (DEBUG)
END
IF &STR(&DEBUG) = &STR() THEN DO
SET &DEBUG = NEBUG
END
IF &SYSCAPS(&STR(&DEBUG)) = DEBUG THEN DO
CONTROL MSG NOFLUSH LIST CONLIST SYMLIST
END
IF &FLUSH = FLUSH THEN DO
SET &ZEDSMSG = &str(Function interrupted)
ISPEXEC SETMSG MSG(ISRZØØ1)
RETURN CODE(Ø)
END
ISPEXEC CONTROL ERRORS RETURN
/* set up a variable of length at least the same as maxlrecl of mail*/
SET &COMMABLANKS = +
&STR(. - - - )
SET &RETCODE = 4
SET &FROMNO = Ø
ISPEXEC VGET (HOSTNAME,SMTPNODE,OWNNODE,NICKOWN,PRIMRECV)
SET &OWNNET = &STR(&OWNNODE..&SMTPNODE)
SET &OWNALT = &STR(&OWNNODE..&NICKOWN)
ISREDIT CURSOR = 1 Ø
SET &RCPTTX = &STR('&SEARCH')
SET &LRCPTTX = &LENGTH(&STR(&SEARCH))
SET &MAXCNT5 = 512 /* SET LOOP CONTROL TO MAX 512 SCANS */
SET &CNT5 = Ø
SET &TORET = Ø
DO WHILE &TORET = Ø AND &CNT5 < &MAXCNT5
    SET &DM = &STR()
    SET &RET = Ø
    ISREDIT SEEK &RCPTTX 1 &RA &RR
    SET &TORET = &RET
    IF &TORET = Ø THEN DO
        ISREDIT (RCPBEGIN,RCPFROW) = CURSOR
        ISREDIT (RCPFREC) = LINE &STR(&SYSNSUB(1,&RCPBEGIN))
        SET &LENRCP = &LENGTH(&STR(&SYSNSUB(1,&RCPFREC)))
        SET &SYSDVAL = +
        &SUBSTR(&RCPFROW+&LRCPTTX:&LENRCP,&STR(&SYSNSUB(1,&RCPFREC)))
        SET &SYSDVAL = &STR(&SYSNSUB(1,&SYSDVAL))
        SET &SRCHSPC = &STR(,) /* look for last comma */
        SET &STARTSPC = 1
        SET &LOCSPC = 1
        SET &LASTSPC = Ø
        SET &MAXT = &LENRCP
        SET &R = Ø
        DO WHILE &LOCSPC > Ø AND &STARTSPC <= &LENRCP AND &R < &MAXT
            SET &R = &R + 1
            SET &LOCSPC = +
            &SYSINDEX(&STR(&SRCHSPC),&STR(&SYSNSUB(1,&RCPFREC)),&STARTSPC)
            IF &LOCSPC > Ø THEN DO
                SET &STARTSPC = &LOCSPC + 1
                SET &LASTSPC = &LOCSPC
            END
        IF &LASTSPC > Ø THEN DO /* is comma followed by all blanks */
            IF &SUBSTR(&LASTSPC:&LENRCP,&STR(&SYSNSUB(1,&RCPFREC))) = +
                &SUBSTR(1:&LENRCP-&LASTSPC-1,&STR(&COMMA BLANKS)) THEN DO
                    ISREDIT CURSOR = &EVAL(&RCPBEGIN + 1) Ø /* continue search */
                    SET &RCPTTX = &STR(P'='') /* for any character */
                    SET &LRCPTTX = Ø
                END
            ELSE DO
                SET &RCPTTX = &STR('&SEARCH') /* else look for */
                SET &LRCPTTX = &LENGTH(&STR(&SEARCH)) /* next &SEARCH */
            END
        END
    ELSE DO
        SET &RCPTTX = &STR('&SEARCH') /* else look for */
        SET &LRCPTTX = &LENGTH(&STR(&SEARCH)) /* next &SEARCH */
    END
END
ELSE DO

ISREDIT CURSOR = &EVAL(&RCPBEGIN + 1) Ø /* continue search */
SET &RCPTTX = &STR('&SEARCH')
SET &LRCPTTX = &LENGTH(&STR(&SEARCH))
END
SET &MAXCNT = 16
SET &N = Ø
SET &A = A
DO WHILE &N < &MAXCNT
    SET &N = &N + 1
    IF &N = &MAXCNT THEN DO
        WRITE ===== Error, loop on N terminated (&SYSICMD).
    END
SET &LENRC = &LENGTH(&STR(&SYSNSUB(2,&&A&N))))
IF &SUBSTR(1:1,&STR(&SYSNSUB(2,&&A&N))) = &STR(<) THEN DO
    SET &&A&N = &SUBSTR(2:&LENRC,&STR(&SYSNSUB(2,&&A&N))))
END
SET &LENRC = &LENGTH(&STR(&SYSNSUB(2,&&A&N))))
IF &SUBSTR(&LENRC:&LENRC,&STR(&SYSNSUB(2,&&A&N)))) = &STR(>) THEN DO
    SET &&A&N = &SUBSTR(1:&LENRC-1,&STR(&SYSNSUB(2,&&A&N))))
END
SET &LENRC = &LENGTH(&STR(&SYSNSUB(2,&&A&N))))
SET &CC = &STR(&SYSNSUB(2,&&A&N)))
SET &SRCHSPC = &STR(&ATSIGN) /* look for at sign */
SET &STARTSPC = 1
SET &LOCSPC = +
&SYSINDEX(&STR(&SRCHSPC),&STR(&SYSNSUB(1,&CC))),&STARTSPC)
IF &LOCSPC > Ø THEN DO
    SET &CC = &SUBSTR(1:&LOCSPC-1,&STR(&SYSNSUB(2,&&A&N))))
    SET &DM = &SUBSTR(&LOCSPC+1:&LENCC,&STR(&SYSNSUB(2,&&A&N))))
    SET &LENCC = &LENGTH(&STR(&SYSNSUB(1,&CC))))
    SET &SRCHSPC = &STR(&NETDLM) /* look for %-sign */
    SET &STARTSPC = 1
    SET &LOCSPC = +
&SYSINDEX(&STR(&SRCHSPC),&STR(&SYSNSUB(1,&CC))),&STARTSPC)
IF &LOCSPC > Ø THEN DO /* nje-network address */
    SET &US = &SUBSTR(1:&LOCSPC-1,&STR(&SYSNSUB(1,&CC))))
    SET &NET = &STR()
    SET &NET = &SUBSTR(&LOCSPC+1:&LENCC,&STR(&SYSNSUB(1,&CC))))
    IF &STR(&SYSLC(&OWNNET)) = &STR(&SYSLC(&SYSNSUB(1,&NET)))) THEN DO
        SET &CC = &STR(&SYSNSUB(1,&US)))
        SET &DM = &STR(&SYSNSUB(1,&OWNNODE))
    END
    IF &STR(&SYSLC(&OWNALT)) = &STR(&SYSLC(&SYSNSUB(1,&NET)))) THEN DO
        SET &CC = &STR(&SYSNSUB(1,&US)))
        SET &DM = &STR(&SYSNSUB(1,&OWNNODE))
    END
END

Editor’s note: this article will be continued in the next issue.

Nils Plum
Systems Programmer (Denmark) © Xephon 1999
NetManage has upgraded its OnNet Host and OnNet Host Suite terminal emulation and TCP/IP suite software, addressing Y2K compliance, supporting Windows 98, and optimizing products for TN3270 and TN5250. New features include support for the Interdrive Version 5.0 client and a Network File System utility that includes improved NIS automount and YP commands.

The company also announced shipments of PC/TCP Network Software Version 5.0 and PC/TCP Kernel Version 5.0, the year 2000-compliant versions of its PC-based TCP/IP networking solutions.

PC/TCP Kernel provides TSR-based TCP/IP stack technology and supports users who need to network in both DOS and Windows. It also includes diagnostic utilities that help optimize, monitor, and troubleshoot network connections.

For further information contact:
NetManage, 10725 N De Anza Blvd, Cupertino, CA 95014, USA.
Tel: (408) 973 7171.
NetManage UK, Chameleon House, 22 Frederick Sanger Road, Guildford, Surrey, GU2 5YD, UK.
Tel: (01483) 302333.

Progress Software has launched IPQoS, the first in the IPPatrol Suite range of quality of service products used to measure and monitor the overall availability and performance of devices and applications as they relate to service level agreements. It monitors and collects statistics on Windows NT, NetWare file systems, and NetWare services, and then aggregates statistics from either single or multiple perspective on a network, and puts it all into reports.

For further information contact:
Progress Software, 14 Oak Park, Bedford, MA 01730-9960, USA.
Tel: (781) 280 4000.
Progress Software, The Square, Basingview, Basingstoke, Hants, RG21 2EQ, UK.
Tel: (01256) 816668.

OpenConnect Systems has begun shipping its SNA Print Server for CIP, a centralized print server adding new print functions to host systems utilizing a Cisco Channel Interface Processor (CIP) or Channel Port Adapter (CPA). It provides centralized management for distributed mainframe printing throughout the enterprise.

The standards-based SNA Print Server for CIP distributes mainframe information to any LAN-attached printer. It interfaces with Cisco’s TN3270 server software for central print management.

For further information contact:
OpenConnect Systems, 2711 LBJ Frwy, Suite 800, Dallas, TX 75234-7324, USA.
Tel: (972) 484 5200.