60 TCP/SNA update

December 2005

In this issue

A itc PUBLICATION

TCP/SNA Update

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Colin Smith
E-mail: info@xephon.com

Subscriptions and back-issues

A year's subscription to *TCP/SNA Update*, comprising four quarterly issues, costs $190.00 in the USA and Canada; £130.00 in the UK; £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 2001 issue, are available separately to subscribers for $49.50 (£33.00) each including postage.

TCP/SNA Update on-line

Code from *TCP/SNA Update*, and complete issues in Acrobat PDF format, can be downloaded from http://www.xephon.com/tcpsna; you will need to supply a word from the printed issue.

Contributions

When Xephon is given copyright, articles published in *TCP/SNA Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

Sending e-mail with XMITIP

XMITIP is a mainframe mail application that is capable of sending electronic mail to any Internet address and can also send mainframe files in one of several different file attachment formats. The Simple Mail Transport Protocol (SMTP) is used for sending the mail with datasets attached using the appropriate SMTP statements.

XMITIP is like TSO TRANSMIT using the Internet Protocol.

XMITIP can be executed as a step within a batch job, under TSO as a command, or under ISPF using an ISPF interface. It can also be used within an automated operations tool to generate messages related to system events.

XMITIP can be used as a step within a batch job to send a report generated by a job via e-mail rather than printing the report. With this approach, the report will arrive in the intended user's electronic mail inbox within a few minutes of the job completing.

Some of the features of XMITIP are:

- Send electronic mail to one or more addresses.
- Send a quick message as a page.
- Send one or more datasets as file attachments in one of the following formats: plain text, HTML, Rich Text Format (RTF), Portable Document Format (PDF), Comma Separated Value (CSV), TSO Transmit (XMIT), and binary.
- Supports address lists.
- Supports CC and BCC.
- Supports priority, sensitivity, and importance.

The ISPF interface provides:

- Field-level help for all entry fields.

- Validation of all entered fields.
- Address table for look-up and selection.
- Dataset table for multiple dataset selection and formatting.
- Reports showing the complete XMITIP command syntax.
- An option to create a file with a complete batch job, which can be submitted, browsed, edited, or copied.

XMITIP is written almost completely in z/OS REXX, with one z/OS Assembler program that performs the MIME (Multipurpose Internet Mail Encoding) used for PDF and binary attachments. The ISPF interface is also written in z/OS REXX using the z/OS ISPF APIs.

INSTALLATION

XMITIP is a tool that can be downloaded from the Web site http://www.lbdsoftware.com/tcpip.html.

There is a zip file that must be expanded into a PC folder. To install the product read the *XMITIP Installation Guide* in Adobe Acrobat (PDF) or Microsoft Word (DOC) format.

This document will tell you how to upload to the mainframe the xmitip.xmit file and how to convert it into the installation partitioned dataset. It will also tell you how to customize the package.

USAGE

The XMITIP parameters are detailed in the XMITIP manual.

XMITIP can be used in two ways:

- Batch mode
- Interactive mode (ISPF).

Batch mode

These examples show XMITIP batch mode utilization. Some

things to be aware of are:

- The dataset referenced by the //SYSEXEC statement is the location where the XMITIP application has been installed.
- When the command exceeds one statement it must be continued with either a + or a – on the statement immediately before the continuation statement.
- The case of the command is not relevant. It can be all upper case, all lower case, or mixed case. Note that the subject is the only information in which you may be concerned about case.
- Not all examples include a FROM keyword; however, it is always good practice to include this keyword with your primary e-mail address coded.
- Using '**–**' for continuation will yield extra spacing which may not be desirable in a long subject while using '**+**' for continuation will suppress the extra spacing.

Send a PDS member with no message

This example will send a member of a partitioned dataset with no message text and a short subject to two users:

```
//JOB EXEC PGM=IKJEFT1B
//SYSEXEC  DD DISP=SHR,DSN=user.lib.exec
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
%XMITIP (user1@domain.com user2@domain.com) nomsg file -
 'user.data.set(member)' -
 from myaddress@domain.com -
 subject 'send a file but no message'
```

Send a message to one address

```
//JOB EXEC PGM=IKJEFT1B
//SYSEXEC  DD DISP=SHR,DSN=user.lib.exec
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
```

```
%XMITIP destaddr@domain.com msgds my.message.text -
 from myaddress@domain.com -
 subject 'test message'
```

Send a message with CC and FROM specified

This is an example of sending an e-mail with two CCs with a FROM word specified:

```
//JOB EXEC PGM=IKJEFT1B
//SYSEXEC  DD DISP=SHR,DSN=user.lib.exec
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
%XMITIP destaddr@domain.com msgds my.message.text subject -
 'Test message' -
 cc (user2@domain.com user3@domain.com) -
 from myaddress@domain.com
/*
```

Send a dataset in landscape with an 8-point font

```
//JOB EXEC PGM=IKJEFT1B
//SYSEXEC DD DISP=SHR,DSN=user.lib.exec
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//REPORT DD DISP=(OLD,DELETE),DSN=&&RPT
//SYSTSIN DD *
%XMITIP destaddr@domain.com nomsg filedd report subject -
 'passed report' -
 from myaddress@domain.com -
 format rtf/land/8
/*
```

Send a dataset to a list using blind copies

```
//STEP1 EXEC PGM=program
//SYSPRINT DD SYSOUT=*
//INPUT DD DISP=SHR,DSN=userid.input.dataset
//OUTPUT DD DISP=(,PASS),UNIT=SYSDA,DSN=&&RPT
//*
//STEP2 EXEC PGM=IKJEFT1B
//SYSEXEC DD DISP=SHR,DSN=user.lib.exec
//SYSPRINT DD SYSOUT=*
//ADDRLIST DD DSN=user.address.list
//SYSTSPRT DD SYSOUT=*
//REPORT DD DISP=(OLD,DELETE),DSN=&&RPT
//SYSTSIN DD *
%XMITIP * bcc user@domain.com nomsg filedd report -
```

```
subject 'report distribution' -
from myaddr@domain.com addressfiledd addrlist
/*
```

This job is composed of two steps. The first one writes the report in the DDname OUTPUT, and the second one sends the file to the DDname ADDRLIST(filename user.address.list). This file contains a list of destination addresses.

Send a dataset using an addressfile dataset

```
//JOB EXEC PGM=IKJEFT1B
//SYSEXEC DD DISP=SHR,DSN=user.lib.exec
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
%XMITIP * nomsg file ‘user.data.set(member)’ -
 subject ‘pds member’ -
 from myaddress@domain.com -
 Addressfile 'user.data.set(address)'
/*
```

where user.data.set(address) contains the following rows:

```
To user1@domain.com
CC user2@domain.org
Cc user3@domain.ca
Bcc user4@domain.uk
```

Using MSGQ

This example demonstrates a REXX EXEC that calls XMITIP with the MSGQ option:

```
/* rexx */
queue “this is a row #1”
queue “this is a row #2”
queue “this is a row #3”
“%xmitip destaddr@domain.com subject ‘test of msgq’” ,
“msgq from myaddr@domain.com”
```

Send a dataset in RTF format using ZIP to save space

```
//JOB EXEC PGM=IKJEFT1B
//SYSEXEC DD DISP=SHR,DSN=user.lib.exec
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
```

```
%xmitip destaddr@domain.com nomsg file ‘user.weekly.report’ -
from myaddr@domain.com -
subject ‘weekly report’ format ziprtf/weekly.rtf/port/9/let
/*
```

Sample using MSGT

```
//JOB EXEC PGM=IKJEFT1B
//SYSEXEC DD DISP=SHR,DSN=user.lib.exec
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
xmitip myaddr@domain.com -
subject ‘Test message text and symbolics’ -
from myaddr@domain.com -
msgt ‘This is a test +
of the message text keyword +
\and testing the symbolics +
\date = &date and date-2 = &date-2 +
\sdate = &sdate and sdate-2 = &sdate-2 +
\udate = &udate and udate-2 = &udate-2 +
\day = &day and day-2 = &day-2 +
\sysid = &sysid \userid = &userid ‘
/*
```

Notice how the continuation lines are coded. The continuation character is the **–** symbol or the **+** symbol. The quotes around the message text are found only before the first character of the text and after the last character – not around each line of text. The \ starts a new line. Leading blanks will be included, for example, after the ‘This is a test’ and before‘of the message text’. To avoid the extra blanks, do not indent the message text.

Sending a message to a pager if a job abnormally ends

```
.....................
.....................
//TEST1 IF (RC > 4 | ABEND = TRUE) THEN
//MAIL EXEC PGM=IKJEFT1B
//SYSEXEC DD DISP=SHR,DSN=user.lib.exec
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
xmitip 2227772424@message.paging.com +
page “job &job(&jobnum) has ended abnormally +
on &date at &time”
```

```
/*
// ENDIF
```

In this example, if the highest return code (RC) for the previous steps is greater than 4 or the job abends, the message will be generated using the PAGE option and an e-mail address of a text pager.

INTERACTIVE MODE

All the documentation of this article refers to XMITIP 4.22.

To start the program type **XMITIP** from TSO option 6, or **TSO XMITIP** from any command line ISPF panel and the screen in Figure 1 will be shown.

To get field information move the cursor to the right of the arrow (===>) and press F1. For instance, the information about the Subject field is shown in Figure 2.

The XMITIP ISPF interface is a quick way to send an SMTP

Figure 1: XMITIP start screen

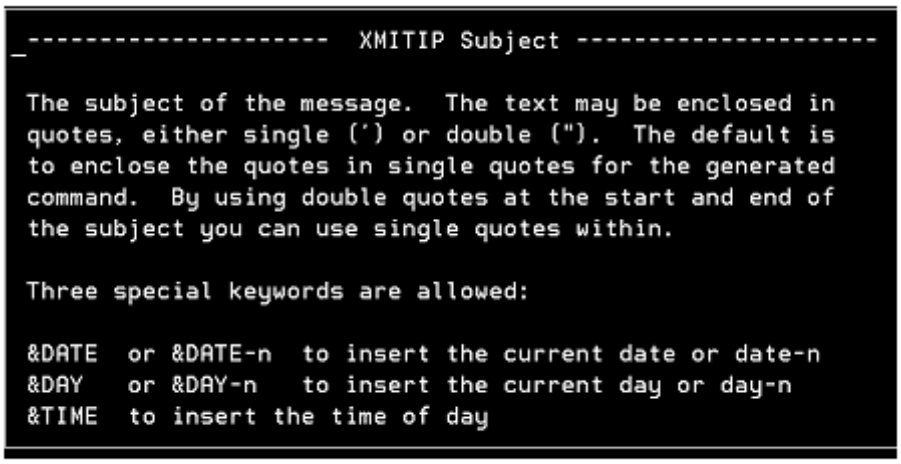

Figure 2: Subject information

message. For instance, to send a simple message you need to fill four fields:

- *Recipient address*, which is the mail address to send the e-mail to.
- *Subject*, which is the subject field.
- *From Address*, which is my e-mail address.
- *Message DSN*.

There are two ways to fill the *Message DSN* field:

- Fill the name of the dataset where, before starting XMITIP, we wrote the message text.
- Put * in the field. Write the message in the black screen.

Press *ENTER* and the F3 key. The message will be sent automatically.

Magni Mauro
Systems Engineer (Italy)

A multi-tasking TCP/IP socket listener for OS/390 or z/OS

TCP/IP socket listener programs are key server entities in many of today's client/server applications. On server systems where request volume is high, being able to service multiple requests simultaneously will generally provide better overall throughput and increased client satisfaction. It's in this type of scenario that a multi-tasking server can be extremely beneficial.

Single-threaded listeners can be effective when server requests are well spaced and no individual request is long running. If there are times when server requests spike or if requests processed by the server can be long running, using a multi-tasking server is a far better option. However, a multi-tasking server does not come without issues of its own. To effectively use a multi-tasking server, the multi-tasking environment must be managed by a main driver program and givesocket()/ takesocket() function calls must be introduced into the environment. A single-threaded listener can operate with a base set of TCP/IP-related functions that include: socket(), bind(), listen(), and accept(). A multi-tasking listener should replace accept() with a timed select(), and will also need to use givesocket() and takesocket() to transfer ownership of an active socket between the main driver task and the processing subtask.

Presumably, the merits of a multi-tasking listener are understood so this article will focus on an example. This article presents a C/C++ multi-tasking listener for OS/390 or z/OS. There are idiosyncrasies in using C/C++ in a multi-tasking environment – primarily, which multi-tasking technique offers the best option. The C Multi-Tasking Facility (MTF) can be used, but it has some restrictions:

- A fixed maximum number of subtasks is set when the multi-tasking environment is initialized.

- The subtasks are all permanently active.
- An abend in any one of the active subtasks renders the entire MTF environment unusable.

Another option is the pthread_ suite of C/C++ functions, although I make this claim cautiously because I have never been able to get socket functions working properly in this environment (this does not mean it doesn't work, but only that I have not been able to get it to work).

A third option is to use native MVS ATTACH/DETACH capabilities. This can be done by creating C/C++ callable functions, written in Assembler, that can perform the creation and removal of subtasks as required. It is this latter option that will be used in the program example in this article.

HOW THINGS WORK

For the purpose of discussion, the process we are examining and the main driver program of the example will be referred to as MTGSL (Multi-Tasking Generic Socket Listener). The MTGSL is a collection of four programs:

1 MTGSL – a C/C++ program that acts as the main driver program for the multi-tasking listener.

2 MTTSKCC – a C/C++ program that acts as the subtask functional component. This is where the logic for the listener functionality resides and where you would most likely make code changes if you wanted to tailor this listener to your specific requirements.

3 ATTACH – an Assembler program designed to be called as a C/C++ function. It ATTACHes the specified program as a subtask and formats the optional program parameters to be usable by a C/C++ main() subtask program. In this specific case, the optional program parameters are the address of a subtask information structure created in the main driver program.

4 DETACH – an Assembler program designed to be called as a C/C++ function. It is used to DETACH subtasks created by the ATTACH() function.

The MTGSL main program establishes the active listener environment. The listener port to be used is passed to the main program as a parameter at program start-up. The main program opens a socket connection (using the socket() function call), acquires a name for the socket (with a bind() function call), and establishes its intent to be a server (using the listen() function call). When the listener is notified of an incoming request (an active select() function), the main program must:

- Acquire a block of storage to be used to track the status of the corresponding subtask (the SUBTASK_INFO structure).
- Prepare to transfer socket ownership with a givesocket() function call.
- ATTACH the subtask processing program (MTTSKCC).

The main program also 'wakes up' periodically if no new work is identified. When this happens, the information regarding outstanding subtasks is examined to determine whether socket ownership has been transferred (in which case the main program's associated socket is closed) and to see whether any of the subtasks have completed. If a subtask has completed, the associated information storage block is released. For this specific example, a check is also made to see whether the corresponding subtask acknowledged a request for listener termination and, if so, the listener is terminated (upon completion of any and all existing subtasks).

PROGRAM COMPILATION AND LINKAGE

The ATTACH and DETACH programs should be assembled with a standard assembly job that includes SYS1.MACLIB and SYS1.MODGEN in the SYSLIB dataset concatenation. The resulting object modules should be directed to an object

code PDS that is the same one that will be used for the C/C++ programs' compile and prelink. Before compiling the MTGSL and MTTSKCC programs, be sure to convert all occurrences of '[' to X'AD' and all occurrences of ']' to X'BD'. The following compile job can be used:

```
//STEP1   EXEC EDCC,CPARM='LIST',
//          CPARM2='RENT,NOSEARCH,NOMAR,NOSEQ,NOOPT',
//          CPARM3='LANGLVL(EXTENDED),SOURCE,LONGNAME,SSCOMM',
//          INFILE=c.source.code.pds(MTGSL),
//          OUTFILE='object.code.pds(MTGSL),DISP=SHR',
//          SYSLBLK=8000
//COMPILE.SYSLIB  DD DSN=TCPIP.SEZACMAC,DISP=SHR
//          DD DSN=CEE.SCEEH.H,DISP=SHR
//          DD DSN=CEE.SCEEH.SYS.H,DISP=SHR)
//STEP2   EXEC EDCC,CPARM='LIST',
//          CPARM2='RENT,NOSEARCH,NOMAR,NOSEQ,NOOPT',
//          CPARM3='LANGLVL(EXTENDED),SOURCE,LONGNAME,SSCOMM',
//          INFILE=c.source.code.pds(MTTSKCC),
//          OUTFILE='object.code.pds(MTTSKCC),DISP=SHR',
//          SYSLBLK=8000
//COMPILE.SYSLIB  DD DSN=TCPIP.SEZACMAC,DISP=SHR
//          DD DSN=CEE.SCEEH.H,DISP=SHR
//          DD DSN=CEE.SCEEH.SYS.H,DISP=SHR
```

Note, the above JCL assumes a pre-existing object code PDS with a blocksize of 8000. The resulting object modules should be prelinked using a job similar to the following:

```
//PLKED1  EXEC PGM=EDCPRLK,PARM='UPCASE,OMVS',REGION=2048K
//SYSMSGS  DD  DSNAME=CEE.SCEEMSGP(EDCPMSGE),DISP=SHR
//SYSLIB   DD  DSN=TCPIP.SEZARNT1,DISP=SHR
//SYSOBJ   DD  DSN=object.code.pds,DISP=SHR
//SYSMOD   DD  DSN=object.code.pds(MTGSLO),DISP=SHR
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  INCLUDE SYSOBJ(MTGSL)
//PLKED2  EXEC PGM=EDCPRLK,PARM='UPCASE,OMVS',REGION=2048K
//SYSMSGS  DD  DSNAME=CEE.SCEEMSGP(EDCPMSGE),DISP=SHR
//SYSLIB   DD  DSN=TCPIP.SEZARNT1,DISP=SHR
//SYSOBJ   DD  DSN=object.code.pds,DISP=SHR
//SYSMOD   DD  DSN=object.code.pds(MTTSKCCO),DISP=SHR
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  INCLUDE SYSOBJ(MTTSKCC)
```

When the ATTACH and DETACH programs have been assembled and the MTGSL and MTTSKCC programs have been compiled and prelinked, the following linkedit job can be used to create the MTGSL and MTTSKCC load modules:

```
//IEWL      EXEC  PGM=HEWLHØ96,PARM='XREF,LIST,MAP,RENT'
//SYSPRINT DD    SYSOUT=*
//SYSUT1    DD    UNIT=SYSDA,SPACE=(CYL,(2,1))
//OBJECT    DD    DSN=object.code.pds,DISP=SHR
//SYSLIB    DD    DSN=TCPIP.SEZACMTX,DISP=SHR
//          DD    DSN=CEE.SCEELKEX,DISP=SHR
//          DD    DSN=CEE.SCEELKED,DISP=SHR
//          DD    DSN=SYS1.CSSLIB,DISP=SHR
//SYSLMOD   DD    DSN=load.library,DISP=SHR
//SYSLIN    DD    *
   INCLUDE OBJECT(MTGSLO)
   INCLUDE OBJECT(ATTACH)
   INCLUDE OBJECT(DETACH)
   ENTRY   CEESTART
   NAME    MTGSL(R)
   INCLUDE OBJECT(MTTSKCCO)
   ENTRY   CEESTART
   NAME    MTTSKCC(R)
```

ACTIVATING THE MTGSL LISTENER

The MTGSL listener can run as a batch job or started task on your OS/390 or z/OS system. Be sure that the userid the batch job or started task will be running under has an OMVS security segment with the requisite information defined to allow for communication with the TCP/IP stack. To start the multi-tasking listener, use a job similar to the following:

```
//MTGSL     EXEC PGM=MTGSL,PARM='9Ø1Ø',REGION=3M
//STEPLIB   DD   DSN=load.library,DISP=SHR
```

This example starts a multi-tasking listener that is listening on port 9010. Ensure that whatever port number is provided in the JCL PARM is available for use by your listener application – you will have to ensure that it is not in use by another TCP/IP application and that it is not an otherwise reserved port in TCP/IP.

Things to note:

- If you want the printf() output from the main task and all subtasks to be directed to the same output DD, include a //SYSPRINT DD SYSOUT=* statement in the above JCL. If no SYSPRINT DD is included in the MTGSL start-up JCL, each subtask will allocate a new output DD with a name prefixed with 'SYS' and a unique 5-digit suffix as in SYS00013.
- The subtask program MTTSKCC is POSIX(OFF) by design. Any C/C++ functions that require POSIX(ON) cannot be used in MTTSKCC.
- To demonstrate the full potential of the multi-tasking server, an artificial delay may need to be programmed into the MTTSKCC program.

SETTING MTGSL INTO ACTION

The MTTSKCC subtask program is designed to process a small subset of HTML formatted requests. If we assume that MTGSL is running on a system with IP address 10.0.2.2 and listening on port 9010, you can trigger a response to your workstation's Web browser by issuing the following on your browser's address line:

```
http://10.0.2.2:9010/httptest
```

If MTGSL is properly contacted, it will return a response to the browser and a browser window will be displayed containing this message:

```
Server test request
```

MTGSL will also respond to:

```
http://10.0.2.2:9010/quit
```

In this case, this will cause the listener to terminate on the corresponding OS/390 or z/OS system and will cause a window on your browser to display:

```
Server termination request
```

Any other message sent by the browser to MTGSL will cause a response of:

```
Unknown request type
```

CONCLUSION

The extra effort required to set up a multi-tasking listener is well worth the benefits that are gained over a single-threaded listener. Being able to isolate the working components (the subtasks) from the network infrastructure (the main listener task) can provide significant advantages. Build into the MTTSKCC program some of your own custom logic that delivers a functional enhancement to your environment – who knows what you might come up with.

MTGSL.C

```
/*
 *  This program provides a multi-tasking TCP/IP socket listener for
 *  OS/390 or z/OS systems.
 *  Once a connection has been accepted on the socket, this program
 *  will attach a subtask to process the incoming request.  This
 *  program keeps track of the status of subtasks using a data
 *  structure to maintain subtask-specific information.
 *  Once you have compiled, prelinked, and linked this program, you
 *  can activate the listener on your OS/390 or z/OS system.  The
 *  TCP/IP port that the server will listen on is specified in the
 *  PARM value passed to the program at start-up as in:
 *  //MTGSL    EXEC PGM=MTGSL,PARM='9010'
 *  In the above case, this listener program will be listening on
 *  port 9010.  From a Web browser, you can direct a request to this
 *  listener as follows:
 *  http://ipaddr:port#/requesttype
 *  where 'ipaddr' is the IP address of the system the HTTP server
 *  program is running on, 'port#' is the port number the server is
 *  listening on, and 'requesttype' is either httptest or quit.  If
 *  this listener program is running on a system with an IP address
 *  of 10.0.2.2 and was using port 9010, a browser request would
 *  look like:
 *  http://10.0.2.2:9010/httptest
 *  If you wanted to terminate the server from the browser, enter
 *  the following (from the browser):
 *  http://10.0.2.2:9010/quit
 *  This sample program uses HTML for feedback to the browser, but
```

```
 *  you could add whatever you desire based on command input and
 *  feedback technique of your preference.
 *  Don't forget - the userid that this program is running under will
 *  require an OMVS security segment for successful operation.
 */
#pragma runopts("POSIX(ON)")
/*  Indicate to the compiler that standard OS linkage will be used.  */
#ifdef __cplusplus
  extern "OS" int ATTACH(char*, unsigned char**, unsigned int*,
                         unsigned int*, unsigned int*, ...);
  extern "OS" int DETACH(unsigned int*, char*);
#else
  #pragma linkage (ATTACH, OS)
  #pragma linkage (DETACH, OS)
#endif
#define  MVS
#include <manifest.h>
#include <bsdtypes.h>
#include <socket.h>
#include <in.h>
#include <netdb.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <tcperrno.h>
#include <stdio.h>
#include <unistd.h>
typedef struct subtask_info {   //Structure for maintaining subtask info
  char attach_workarea[260];
  int socket;
  int take_socket;
  unsigned int ECB;
  unsigned int TCB;
  char myname[8];
  char mysname[8];
  int shutdown;
  struct subtask_info *next;
} SUBTASK_INFO;
#define ERROR_SOCKET_CREATE      -1000
#define ERROR_SOCKET_CONNECT     -1001
#define ERROR_SOCKET_PORT_USED   -1002
#define ERROR_SOCKET_BIND_DENIED -1003
#define ERROR_SOCKET_BIND        -1004
#define ERROR_SOCKET_LISTEN      -1005
#define ERROR_SOCKET_INUSE       -1006
#define ERROR_ACCEPT             -1007
//#define SELECT_WAIT 5
#define SELECT_WAIT 2
#define BUFSIZE 1023
#define SOCKET_BACKLOG   5
```

```
/*   doaccept(int *socket#)                                          */
/*   doaccept() waits for a client connection to occur.  A select()  */
/*   is performed waiting for socket connection to become active.    */
/*   The select() has a wait time option that causes the function    */
/*   to wake up after a period of non-activity.  Modify the wait     */
/*   time value specified in time.tv_sec as appropriate for your     */
/*   environment.                                                    */
int doaccept(int *s)
{
  char msg_buff[256];
  int temps;
  int clsocket;
  struct sockaddr clientaddress;
  int addrlen;
  int maxfdpl;
  struct fd_set readmask;
  struct fd_set writmask;
  struct fd_set excpmask;
  int rc;
  struct timeval time;
  temps = *s;
  time.tv_sec = SELECT_WAIT;             // SELECT_WAIT is 5 seconds
  time.tv_usec = 0;
  maxfdpl = temps + 1;
  FD_ZERO(&readmask);
  FD_ZERO(&writmask);
  FD_ZERO(&excpmask);
  FD_SET(temps, &readmask);
  rc = select(maxfdpl, &readmask, &writmask, &excpmask, &time);
  if ( rc < 0 )
  {
    sprintf(msg_buff, "Error from select\n");
    tcperror(msg_buff);
    printf("doaccept() select() errno: %d...%d\n",
           errno,temps);
    return(rc);
  }
  else if ( rc == 0 )                    // Time limit has expired
  {
    return(rc);
  }
  else
  {
    addrlen = sizeof(clientaddress);
    clsocket = accept(temps, &clientaddress, &addrlen);
    if ( clsocket < 0 )
    {
      printf(" doaccept() accept() errno: %d...%d\n",
             errno,temps);
    }
```

```
    return(clsocket);
  }
}
/*   dogive(int *socket#, char *name)                                    */
/*   dogive() "gives" a socket pointed to by socket# and owned by        */
/*   jobname name.                                                       */
int dogive(int *clsocket, char *myname)
{
  int rc;
  struct clientid cid;
  int temps;
  temps = *clsocket;
  memset(&cid, Ø, sizeof(cid));
  cid.domain = AF_INET;
  memcpy(cid.name, myname, 8);
  memcpy(cid.subtaskname, "        ", 8);
  rc = givesocket(temps, &cid);
  return(rc);
}
/*   Subtask_Complete_Check(SUBTASK_INFO *subtask_info_first,            */
/*                          unsigned int *shutdown_flag)                 */
/*   Subtask_Complete_Check() examines the subtask_info structures       */
/*   starting with subtask_info_first.  If there are any structures      */
/*   on the chain, it determines whether the corresponding subtask       */
/*   has appropriately taken the assigned socket and if it has, it       */
/*   closes the parent task's socket.  If the subtask has completed,     */
/*   this routine checks to see whether the associated subtask           */
/*   acknowledged a browser termination request and if it did, it        */
/*   sets the SHUTDOWN flag.  As well, if the subtask has completed      */
/*   but the parent task's socket was not flagged as taken, this         */
/*   routine will do socket clean up.                                    */
SUBTASK_INFO* Subtask_Complete_Check(SUBTASK_INFO *subtask_info_first,
                                     unsigned int *SHUTDOWN)
{
  SUBTASK_INFO *subtask_info_temp;
  SUBTASK_INFO *subtask_info_prev;
  SUBTASK_INFO *subtask_info;
  int i;
  unsigned int e;
  subtask_info_temp = subtask_info_first;
  subtask_info_prev = subtask_info_first;
  while (subtask_info_temp != NULL)
  {
    subtask_info = subtask_info_temp;
    printf("take_socket indicator: %d for socket %d\n",
           subtask_info->take_socket, subtask_info->socket);
    if (subtask_info->take_socket == 1)
    {
      shutdown(subtask_info->socket, 2);
      close(subtask_info->socket);
```

```
      subtask_info->take_socket = 99;
    }
    e = subtask_info->ECB;
    e = e >> 24;
    e = e & 0x00000040;
    if (e != 0)
    {
      printf("Subtask complete\n");
      i = DETACH(&subtask_info->TCB,"NOSTAE");
      if (i != 0)
      {
        printf("subtask failed to DETACH successfully; rc=%d\n",i);
      }
      if (subtask_info->shutdown == 1)
      {
        *SHUTDOWN = 1;
      }
      if (subtask_info->take_socket != 99)
      {
        shutdown(subtask_info->socket, 2);
        close(subtask_info->socket);
      }
      if (subtask_info == subtask_info_first)
      {
        subtask_info_first = subtask_info->next;
        subtask_info_temp = subtask_info_first;
        subtask_info_prev = subtask_info_first;
      }
      else
      {
        subtask_info_temp = subtask_info->next;
        subtask_info_prev->next = subtask_info_temp;
      }
      free(subtask_info);
    }
    else
    {
      subtask_info_prev = subtask_info_temp;
      subtask_info_temp = subtask_info_temp->next;
    }
  }
  return(subtask_info_first);
}
/*   Listener(int portNo, int backLog)                                  */
/*   Generic socket listener:                                           */
/*   This subroutine provides a basic, generic socket listener.         */
/*   The port number to be used is passed by the calling routine        */
/*   and this routine acquires a socket, applies a name to the          */
/*   socket using bind(), and readies the socket to accept client       */
/*   connection requests using listen().  At that point, requests       */
```

```
/*   sent by clients can be processed.                                  */
int Listener(int portNo, int backLog)
{
  int listener, caller;
  int portArg;
  struct sockaddr_in address;
  int rc;
  struct clientid cid;
  char myname[8];
  char mysname[8];
  int clsocket;
  SUBTASK_INFO *subtask_info_first;
  SUBTASK_INFO *subtask_info_temp;
  SUBTASK_INFO *subtask_info_prev;
  SUBTASK_INFO *subtask_info;
  unsigned int SHUTDOWN;
  unsigned int workarea_addr;
  int ret_code;
  char msg_buff[256];
  int option_value;
  int option_len;
  int accept_fail_count;
  char line[32768] = {0};
  char out_line[8092] = {0};
  int i, n, good_request;
  i = 256;
  ret_code = maxdesc(&i, &i);
  ret_code = getdtablesize();
  SHUTDOWN = 0;
  subtask_info_first = NULL;
  accept_fail_count = 0;
  portArg = htons( portNo );
  memset( &address, 0, sizeof(address) );
  address.sin_family       = AF_INET;
  address.sin_port         = portArg;
  address.sin_addr.s_addr  = htonl(INADDR_ANY);
  memset(&cid, 0, sizeof(cid));
  rc = getclientid(AF_INET, &cid);
  memcpy(myname, cid.name, 8);
  memcpy(mysname, cid.subtaskname, 8);
/*  Acquire a socket. */
  listener = socket( AF_INET, SOCK_STREAM, 0 );
  if( listener < 0 )
  {
    printf("socket() failed rc %d errno %d\n",listener,errno);
    return ERROR_SOCKET_CREATE;
  }
/*  Set the socket option to allow reuse of the specified port if
 *  it's for the same application. */
  option_value = 1;
```

```
  option_len = sizeof(option_value);
  rc = setsockopt(listener, SOL_SOCKET, SO_REUSEADDR,
                  (char *) &option_value, option_len);
/*  Apply a unique local name to the socket. */
  rc = bind(listener, (struct sockaddr *)(&address), sizeof(address));
  if ( rc < 0 )
  {
    if ( errno != EADDRINUSE )
    {
      i = errno;
      printf("bind() failed rc %d errno %d\n",rc,errno);
      close(listener);
      if( i==EINVAL )
        return ERROR_SOCKET_PORT_USED;
      else if( errno==EACCES )
        return ERROR_SOCKET_BIND_DENIED;
      else
        return ERROR_SOCKET_BIND;
    }
/*  Loop for up to a minute if EADDRINUSE is being returned by the
 *  bind() request. */
    for ( i=0; i<30; i++)
    {
      sleep(2);                              //  Wait for 2 seconds
      rc=bind(listener, (struct sockaddr *)(&address), sizeof(address));
      if ( rc == 0 )
      {
        break;
      }
    }
    if ( rc < 0 )
    {
      close(listener);
      return ERROR_SOCKET_INUSE;
    }
  }
/*  Ready the socket to accept client connection requests. */
  if( listen(listener, backLog) < 0 )
  {
    i = errno;
    close(listener);
    return ERROR_SOCKET_LISTEN;
  }
  for(;;)
  {
/*   The doaccept() function performs a select() which will allow    */
/*   this program to "wake up" periodically.  Using doaccept() can   */
/*   provide for more sophisticated operator command processing and  */
/*   is useful for managing a multi-tasking environment as is        */
/*   demonstrated in this example.                                   */
```

```
    caller = doaccept(&listener);
    if( caller<Ø )
    {
      if( errno==EINTR ]] errno==EMFILE ]] errno==ENFILE ]]
          errno==24 ]] errno==23 )
      {
        if(errno==EINTR)
        {
          printf("System call ACCEPT interrupted.  Trying again\n");
        }
        if(errno==EMFILE ]] errno==ENFILE ]] errno==24 ]] errno==23)
        {
          accept_fail_count = accept_fail_count + 1;
          if(accept_fail_count >= 15)
          {
            printf("accept() fail limit reached.  Trying again\n");
            accept_fail_count = Ø;
          }
        }
        continue;
      }
      else
      {
        break;
      }
    }
    accept_fail_count = Ø;
/*   Determine whether we have a real accept condition or if we just */
/*   did a timed select() wake up.                                   */
    if ( caller == Ø )
    {
      if (subtask_info_first != NULL)
      {
        subtask_info_first = Subtask_Complete_Check(subtask_info_first,
                             &SHUTDOWN);
      }
      if (subtask_info_first == NULL && SHUTDOWN == 1)
      {
        return(Ø);
      }
    }
    else
    {
      subtask_info = (SUBTASK_INFO*)calloc(1,sizeof(SUBTASK_INFO));
      memset(subtask_info, Ø, sizeof(SUBTASK_INFO));
      memcpy(subtask_info->myname,myname,8);
      memcpy(subtask_info->mysname,mysname,8);
      subtask_info->socket = caller;
      subtask_info->take_socket = Ø;
      subtask_info->shutdown = Ø;
```

```
      subtask_info->next = NULL;
      if (subtask_info_first == NULL)
      {
        subtask_info_first = subtask_info;
      }
      else
      {
        subtask_info_temp = subtask_info_first;
        while (subtask_info_temp->next != NULL)
        {
          subtask_info_temp = subtask_info_temp->next;
        }
        subtask_info_temp->next = subtask_info;
      }
      printf("dogive() for socket %d and %s\n", caller, myname);
      i = dogive(&caller, myname);
      i = ATTACH("MTTSKCC ", &subtask_info,
                 &subtask_info->ECB, &subtask_info->TCB,
                 NULL, subtask_info);
      subtask_info_first = Subtask_Complete_Check(subtask_info_first,
                           &SHUTDOWN);
    }
  }
  sprintf(msg_buff, "accept failed\n");
  perror(msg_buff);
  return ERROR_ACCEPT;
  }
/*  The main routine extracts the program PARM representing the
 *  port number to be used by this listener.  It then calls the
 *  Listener() function to perform the listener dialogue
 *  processing. */
main(int argc, char *argv[])
{
  int err;
  int listener_port;
  if (argc != 2)
  {
    return(8);
  }
  listener_port = atoi(argv[1]);
  printf("Listen port is %d\n",listener_port);
  err = Listener(listener_port, SOCKET_BACKLOG);
  return err;
  }
```

MTTSKCC.C

```
/*  The MTTSKCC program is ATTACHed as a subtask program from the
 *  main listener program.  MTTSKCC takes over a socket given to
```

```
 *  a subtask by the main listener program, flags the socket as
 *  being taken, and then completes the dialogue with the initiating
 *  requestor.
 *  The comments in the main listener program indicate the HTTP browser
 *  requests that are addressed by this program's code. */
#pragma runopts (NOEXECOPS,NOARGPARSE)
#define  MVS
#include <manifest.h>
#include <bsdtypes.h>
#include <socket.h>
#include <in.h>
#include <netdb.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <tcperrno.h>
#include <stdio.h>
#include <unistd.h>
typedef struct subtask_info {  // Structure for maintaining subtask info
  char attach_workarea[260];
  int socket;
  int take_socket;
  unsigned int ECB;
  unsigned int TCB;
  char myname[8];
  char mysname[8];
  int shutdown;
  struct subtask_info *next;
} SUBTASK_INFO;
struct PARM_STRUCT {
  struct PARMS *PARMS;
  };
struct PARMS {
  struct SUBTASK_INFO *PARM1;
  };
#define BUFSIZE 1023
/*   doget() will take over ownership for a specified socket from    */
/*   the primary listener task.                                      */
int doget(int *clsocket, char *xtskname, char *xtsksname)
{
  int rc;
  int temps;
  struct clientid cid;
  memset(&cid, 0, sizeof(cid));
  temps = *clsocket;
  memcpy(cid.name, xtskname, 8);
  memcpy(cid.subtaskname, xtsksname, 8);
  cid.domain = AF_INET;
  printf("doget() jobname: %s; give socket; %d\n", cid.name, temps);
  rc = takesocket(&cid, temps);
```

```
  printf("takesocket rc=%d; errno=%d\n", rc, errno);
  *clsocket = temps;
  return (rc);
}
/*   MTTSKCC(int argc, char *argv[])                                    */
/*   Generic socket listener subtask:                                   */
/*   This routines runs as a socket listener subtask.  It takes         */
/*   the main task's socket and then performs the requested             */
/*   function as determined from the incoming HTML request.  If a       */
/*   shutdown is requested, the appropriate SUBTASK_INFO structure      */
/*   field is set accordingly.                                          */
main(int argc, char *argv[])
{
  char xtskname[8];
  char xtsksname[8];
  int rc;
  int clsocket;
  int caller;
  char line[32768] = {0};
  char out_line[8092] = {0};
  int i, n, good_request;
  SUBTASK_INFO *subtask_info;
   subtask_info=(SUBTASK_INFO*)((((struct PARM_STRUCT*)argv[1])->PARMS)-
>PARM1);
  clsocket = subtask_info->socket;
  memcpy(&xtskname, subtask_info->myname, 8);
  memcpy(&xtsksname, subtask_info->mysname, 8);
  printf("myname %s\n",subtask_info->myname);
  printf("main task socket: %d\n",subtask_info->socket);
  rc = doget(&clsocket, xtskname, xtsksname);
  printf("doget() rc=%d\n",rc);
  if (rc >= 0)
  {
    caller = rc;
    printf("subtask local socket is %d\n",caller);
    printf("  main task socket: %d\n",subtask_info->socket);
    subtask_info->take_socket = 1;
    good_request = 1;
    n = read( caller, line, BUFSIZE );
    line[n] = 0;
    __atoe(line);
//  printf("Inbound request: %s\n",line);
    if (strncmp(line+5,"quit",4) == 0)
    {
      printf("termination request\n");
/*  Build the html response string to indicate that the termination
 *  request has been received. */
      strcpy(out_line,
         "<html><head><title>Term request acknowledged</title></head>");
      strcat(out_line,
```

```
              "<b><font face=\"Verdana\" size=\"2\">");
      strcat(out_line,
              "<p align=\"left\">Server termination request</b></font>");
/*  Convert the response to ascii and send it back to the browser. */
      __etoa(out_line);
      n = write( caller, out_line, strlen(out_line) );
      shutdown(caller,2);
      close(caller);
      subtask_info->shutdown = 1;
    }
    else if (strncmp(line+5,"httptest",8) == 0)
    {
      printf("request type is httptest\n");
/*  Build the html response string to indicate that the test
 *  request has been received. */
      strcpy(out_line,
         "<html><head><title>Test request acknowledged</title></head>");
      strcat(out_line,
              "<b><font face=\"Verdana\" size=\"2\">");
      strcat(out_line,
              "<p align=\"left\">Server test request</b></font>");
/*  Convert the response to ascii and send it back to the browser. */
      __etoa(out_line);
      n = write( caller, out_line, strlen(out_line) );
      shutdown(caller,2);
      close(caller);
    }
    else
    {
      good_request = 0;
    }
    if (good_request == 0)
    {
      printf("Unknown request\n");
//    printf("Unknown request %s\n",line);
/*  Build the html response string to indicate that an invalid
 *  request has been received. */
      strcpy(out_line,
              "<html><head><title>Unknown request type</title></head>");
      strcat(out_line,
              "<b><font face=\"Verdana\" size=\"2\">");
      strcat(out_line,
              "<p align=\"left\">Unknown request type</b></font>");
/*  Convert the response to ascii and send it back to the browser. */
      __etoa(out_line);
      n = write( caller, out_line, strlen(out_line) );
      shutdown(caller,2);
      close(caller);
    }
  }
```

```
  return 0;
}
```

ATTACH.ASM

```
*--------------------------------------------------------------------*
*   This program provides support for a C/C++ ATTACH() function.     *
*   It is designed to function similarly to the ATTACH macro for     *
*   Assembler programs.  The ATTACH() function supports a basic      *
*   program ATTACH with parameter passing support.  This version     *
*   of the function does not support the more esoteric ATTACH macro  *
*   parameters, but does provide for specifying the address of a DCB *
*   for an open TASKLIB DD.  As well, for tasks that need to be      *
*   waited on for completion, the ATTACH() function supports the     *
*   passing of an ECB area address and a return area for the TCB     *
*   address.                                                         *
*   This specific ATTACH() function expects to attach a C/C++        *
*   main() program.  It passes one parameter to the C/C++ program -  *
*   the address of a vector of parameter addresses built from the    *
*   optional parameters passed on the ATTACH() function call.        *
*   For this program the following register usage is in effect:      *
*   RØ - R1   :  work registers, but generally available for use     *
*                by calls to system functions                        *
*   R2        :  used to save the incoming parameter address         *
*   R3 - R7   :  work registers                                      *
*   R8        :  used as base register for the required incoming     *
*                workarea                                            *
*   R9        :  work register                                       *
*   RlØ - R11 :  reserved (future base register expansion)           *
*   R12       :  base register                                       *
*   R13       :  DSA/workarea address                                *
*   R14 - R15 :  work registers, return address and return code, but *
*                generally available for use by calls to system      *
*                functions                                           *
*--------------------------------------------------------------------*
*   Routine:     ATTACH                                              *
*   Function:    To provide MVS ATTACH capabilities from an IBM      *
*                C/C++ program.                                      *
*   Arguments:   ATTACH program name address (right pad with blanks) *
*                ATTACH() function workarea address.  This workarea  *
*                   must be a minimum of 256 bytes plus four bytes   *
*                   for each optional parm that is passed.  It       *
*                   should not be modified by the calling program or *
*                   used for any other ATTACH() calls while this     *
*                   task is active.                                  *
*                ECB area address (or NULL)                          *
*                TCB area address (or NULL)                          *
*                TASKLIB DCB area address (or NULL)                  *
*                Optional parms to be passed to the attached program. *
```

```
*                   The last parm address will have the x'80' flag    *
*                   set.  You can pass up to 256 optional parms.      *
*   Return:      0 if the ATTACH is successful                         *
*               -7 ATTACH failed.  If an ECB area address has been    *
*                  provided, the ECB area contains the ATTACH         *
*                  return code.                                       *
*               -8 incorrect minimum number of parms.  The ATTACH()   *
*                  function call requires a minimum of five parms.    *
*               -9 no parms were detected on entry to ATTACH()        *
*   C usage:    i = ATTACH(&pgm_name, &attach_workarea_addr,          *
*                          &ecb, &tcb, &tasklib_dcb,                  *
*                          &opt_parm1, &opt_parm2, ... , &opt_parmn); *
*-------------------------------------------------------------------*
ATTACH   CSECT
ATTACH   AMODE 31
ATTACH   RMODE ANY
         EDCPRLG BASEREG=R12,DSALEN=WORKLEN
         USING ATTAWORK,R13             Addressability to temp storage
         LTR   R1,R1                    Parms ok?
         BZ    RETNEGØ9                 No - return -9
         LR    R9,R1                    Copy parm address
         L     R2,Ø(,R9)                Get buffer address
         N     R2,=X'8ØØØØØØØ'          Turn off address value
         C     R2,=X'8ØØØØØØØ'          Is this the last parm?
         BE    RETNEGØ8                 Yes - return -8
         L     R2,4(,R9)                Get buffer address
         N     R2,=X'8ØØØØØØØ'          Turn off address value
         C     R2,=X'8ØØØØØØØ'          Is this the last parm?
         BO    RETNEGØ8                 Yes - return -8
         L     R2,4(,R9)                Get buffer address
         L     R8,Ø(,R2)                Get WORKAREA address
         USING WORKAREA,R8              Set WORKAREA addressability
         LA    R6,PARMS                 Get parm address area address
*   R1 contains the address of the incoming parms.  Check to          *
*   make sure that a valid, minimum number of parameters have been    *
*   passed.                                                           *
         ST    R1,PARMØ                 Save incoming parm address
         LTR   R1,R1                    Parms ok?
         BZ    RETNEGØ9                 No - return -9
         LR    R9,R1                    Copy parm address
         L     R2,Ø(,R9)                Get buffer address
         ST    R2,ATTAPGM               Save pgm name address
         TM    ATTAPGM,X'8Ø'            Is this the last parm?
         BO    RETNEGØ8                 Yes - return -8
         L     R2,4(,R9)                Get buffer address
         ST    R2,ATTAWRK               Save work area address
         TM    ATTAWRK,X'8Ø'            Is this the last parm?
         BO    RETNEGØ8                 Yes - return -8
         L     R2,8(,R9)                Get buffer address
         ST    R2,ATTAECB               Save ECB address
```

```
         TM    ATTAECB,X'8Ø'           Is this the last parm?
         BO    RETNEGØ8                Yes - return -8
         L     R2,12(,R9)              Get buffer address
         ST    R2,ATTATCB              Save TCB address
         TM    ATTATCB,X'8Ø'           Is this the last parm?
         BO    RETNEGØ8                Yes - return -8
         L     R2,16(,R9)              Get buffer address
         ST    R2,ATTATSKL             Save TASKLIB DCB address
         LA    R14,256                 Set parm base number
         LA    R15,256                 Set parm base number
         TM    ATTATSKL,X'8Ø'          Is this the last parm?
         BNO   MOREPRMS                No - capture additional parms
         OI    PARMS,X'8Ø'             Set last parm flag
         B     PASTPRMS                Bypass parm capture
MOREPRMS DS    ØH
         NI    ATTATSKL,X'7F'          Turn off the x'8Ø' flag
         OI    FLAG1,PPARMS            Set parm flag
         LA    R1,PARMS                Get parm addr save area addr
         LA    R15,256                 Set loop count
         LA    R9,2Ø(,R9)              Point to first parm addr
PARMLP   DS    ØH
         MVC   Ø(4,R1),Ø(R9)           Copy parm address
         TM    Ø(R1),X'8Ø'             Last parm?
         BNO   PARMNEXT                No - set up for next parm
         BCTR  R15,Ø                   Reduce loop count by one
         B     PASTPRMS                Done with parms
PARMNEXT DS    ØH
         LA    R1,4(,R1)               Point to next target area
         LA    R9,4(,R9)               Point to next source area
         BCT   R15,PARMLP              Check for more
         OI    PARMS+255*4,X'8Ø'       Set last parm flag
PASTPRMS DS    ØH
         MVC   PARMLN(2),=H'Ø'         Set default parm length of Ø
         SR    R14,R15                 Calculate number of parms
         LTR   R14,R14                 Any parms?
         BZ    NOPRMS                  No - bypass initialization
         LA    R6,PARMS                Get parm area address
         ST    R6,PARMADDR             Save it
         OI    PARMADDR,X'8Ø'          Set last parm flag
         MVC   PARMLN(2),=H'4'         Set length
NOPRMS   DS    ØH
         L     R3,ATTAPGM              Get pgm name address
         L     R4,ATTAECB              Get ECB area address
         L     R5,ATTATCB              Get TCB area address
         L     R7,ATTATSKL             Get TASKLIB DCB area address
         LTR   R4,R4                   An ECB address?
         BZ    NODETACH                No - DETACH isn't required
         LTR   R5,R5                   A TCB address?
         BZ    NODETACH                No - DETACH isn't required
         XC    Ø(4,R4),Ø(R4)           Clear the ECB
```

```
*   ATTACH the requested program.                                       *
         LA    R6,PARMLN                Get parm addr
         MVC   ATTACHWK(ATTACHLN),ATTACHLS Copy the model
         LTR   R7,R7                    A TASKLIB DCB?
         BNZ   TASKLIB1                 Yes - issue ATTACH with TASKLIB
         ATTACHX EPLOC=(R3),         ** Specified program            **X
               ECB=(R4),             ** Target ECB                   **X
               PARAM=((R6)),         ** Specify parameters           **X
               MF=(E,ATTAPARM),      ** Indicate dynamic parm area   **X
               VL=1,                 ** Set X'80' bit on last parm   **X
               SF=(E,ATTACHWK)       ** Indicate execute form        **
         LTR   R15,R15                  All's well?
         BNZ   RETNEG07                 No - save RC in ECB area
         ST    R1,0(,R5)                Save TCB address
         B     RETURNOK                 Return
TASKLIB1 DS    0H
         L     R7,0(,R7)                Get TASKLIB DCB address
         ATTACHX EPLOC=(R3),         ** Specified program            **X
               ECB=(R4),             ** Target ECB                   **X
               PARAM=((R6)),         ** Specify parameters           **X
               TASKLIB=(R7),         ** TASKLIB DCB                  **X
               MF=(E,ATTAPARM),      ** Indicate dynamic parm area   **X
               VL=1,                 ** Set X'80' bit on last parm   **X
               SF=(E,ATTACHWK)       ** Indicate execute form        **
         LTR   R15,R15                  All's well?
         BNZ   RETNEG07                 No - save RC in ECB area
         ST    R1,0(,R5)                Save TCB address
         B     RETURNOK                 Return
NODETACH DS    0H
*   ATTACH the requested program.                                       *
         LA    R6,PARMLN                Get parm addr
         MVC   ATTACHWK(ATTACHLN),ATTACHLS Copy the model
         LTR   R7,R7                    A TASKLIB DCB?
         BNZ   TASKLIB2                 Yes - issue ATTACH with TASKLIB
         ATTACHX EPLOC=(R3),         ** Specified program            **X
               PARAM=((R6)),         ** Specify parameters           **X
               MF=(E,ATTAPARM),      ** Indicate dynamic parm area   **X
               VL=1,                 ** Set X'80' bit on last parm   **X
               SF=(E,ATTACHWK)       ** Indicate execute form        **
         LTR   R15,R15                  All's well?
         BNZ   RETNEG07                 No - save RC in ECB area
         B     RETURNOK                 Return
TASKLIB2 DS    0H
         L     R7,0(,R7)                Get TASKLIB DCB address
         ATTACHX EPLOC=(R3),         ** Specified program            **X
               PARAM=((R6)),         ** Specify parameters           **X
               TASKLIB=(R7),         ** TASKLIB DCB                  **X
               MF=(E,ATTAPARM),      ** Indicate dynamic parm area   **X
               VL=1,                 ** Set X'80' bit on last parm   **X
               SF=(E,ATTACHWK)       ** Indicate execute form        **
```

```
         LTR   R15,R15                 All's well?
         BNZ   RETNEGØ7                No - save RC in ECB area
         B     RETURNOK                Return
RETURNOK EQU   *
         MVC   RETCODE(4),=F'Ø'        Set return code to Ø
         B     RETURN                  Return
RETNEGØ7 EQU   *
         ST    R15,Ø(,R4)              Save RC in ECB area
         MVC   RETCODE(4),=F'-7'       Set return code to -7
         B     RETURN                  Return
RETNEGØ8 EQU   *
         MVC   RETCODE(4),=F'-8'       Set return code to -8
         B     RETURN                  Return
RETNEGØ9 EQU   *
         MVC   RETCODE(4),=F'-9'       Set return code to -9
         B     RETURN                  Return
RETURN   EQU   *
         L     R15,RETCODE             Load return code
         EDCEPIL
ATTACHLS ATTACHX SF=L
ATTACHLN EQU   *-ATTACHLS
         LTORG
ATTAWORK EDCDSAD
WORKLEN  EQU   *-ATTAWORK
WORKAREA DSECT
ATTAWRKL DS    F                       Length of this WORKAREA
PARMØ    DS    F                       Address of incoming parms
ATTAPGM  DS    F                       Address of ATTACH pgm name
ATTAWRK  DS    F                       Address of this WORKAREA
ATTAECB  DS    F                       Address of the ATTACH ECB
ATTATCB  DS    F                       Address of TCB addr return area
ATTATSKL DS    F                       Address of TASKLIB DCB addr
RETCODE  DS    F                       Return code
DBL1     DS    2D                      Dbl work work area
DBL2     DS    2D                      Dbl work work area
FLAGS    DS    ØF
FLAG1    DS    XL1
PPARMS   EQU   X'8Ø'
FLAG2    DS    XL1
FLAG3    DS    XL1
FLAG4    DS    XL1
ATTAPARM DS    F
PARMLIST DS    ØF,H
PARMLN   DS    H
PARMADDR DS    CL4,F
ATTACHWK DS    ØD,CL(ATTACHLN)
PARMS    DS    256F                    Incoming parm addresses
WORKLEN2 EQU   *-PARMØ
RØ       EQU   Ø
R1       EQU   1
```

```
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         END
```

DETACH.ASM

```
*--------------------------------------------------------------------*
*   This file contains the Assembler support code for a DETACH()     *
*   function call.  This routine is intended to be called from       *
*   IBM C/C++ programs and is used to remove a previously created    *
*   subtask.                                                         *
*   Register Usage Conventions:                                      *
*   RØ - R1   :  work registers, but generally available for use     *
*                by calls to system functions                        *
*   R2        :  used to save the incoming parameter address         *
*   R3 - R9   :  work registers                                      *
*   R1Ø - R11 :  reserved (future base register expansion)           *
*   R12       :  base register                                       *
*   R13       :  DSA/workarea address                                *
*   R14 - R15 :  work registers, return address and return code, but *
*                generally available for use by calls to system      *
*                functions                                           *
*--------------------------------------------------------------------*
*   Routine:     DETACH                                              *
*   Function:    To provide MVS DETACH capabilities from an IBM      *
*                C/C++ program.                                      *
*   Arguments:   TCB area address                                    *
*                STAE option indicator address (STAE/NOSTAE)         *
*   Return:      Ø if the DETACH is successful                       *
*               -1 task was DETACHed while active                    *
*               -8 incorrect number of parms.  the DETACH() function *
*                  call requires a tcb address parm.                 *
*               -9 no parms were detected on entry to DETACH()       *
*   C Usage:    i = DETACH(&tcb, &stae_opt);                        *
*--------------------------------------------------------------------*
DETACH   CSECT
DETACH   AMODE 31
```

```
DETACH   RMODE ANY
         EDCPRLG BASEREG=R12,DSALEN=WORKLEN
         LR    R2,R1                    Save incoming parm addr
         USING DETAWORK,R13             Addressability to temp storage
         ST    R2,PARMØ                 Save incoming parm address
         LTR   R2,R2                    Parms ok?
         BZ    RETNEGØ9                 No - return -9
         L     R9,Ø(,R2)                Get buffer address
         ST    R9,DETATCB               Save TCB address
         TM    DETATCB,X'8Ø'            Is this the last parm?
         BO    RETNEGØ8                 Yes - return -8
         L     R9,4(,R2)                Get buffer address
         ST    R9,DETASTAE              Save STAE option indicator addr
         TM    DETASTAE,X'8Ø'           Is this the last parm?
         BNO   RETNEGØ8                 No - return -8
         L     R1,DETASTAE              Get STAE option indicator addr
         CLC   Ø(4,R1),=C'STAE'         STAE=YES?
         BE    STAEYES                  Yes - DETACH with STAE=YES
         B     STAENO                   No - DETACH with STAE=NO
*   DETACH the requested TCB (STAE=YES).                               *
STAEYES  DS    ØH
         L     R5,DETATCB               Get TCB area addr
         DETACH (R5),STAE=YES           DETACH
         LTR   R15,R15                  All's well?
         BNZ   RETNEGØ1                 No - return -1
         MVC   RETCODE(4),=F'Ø'         Set return code
         B     RETURN                   Return
*   DETACH the requested TCB (STAE=NO).                                *
STAENO   DS    ØH
         L     R5,DETATCB               Get TCB area addr
         DETACH (R5),STAE=NO            DETACH
         LTR   R15,R15                  All's well?
         BNZ   RETNEGØ1                 No - return -1
         MVC   RETCODE(4),=F'Ø'         Set return code
         B     RETURN                   Return
RETNEGØ1 DS    ØH
         MVC   RETCODE(4),=F'-1'        Set return code to -1
         B     RETURN                   Return
RETNEGØ8 DS    ØH
         MVC   RETCODE(4),=F'-8'        Set return code to -8
         B     RETURN                   Return
RETNEGØ9 DS    ØH
         MVC   RETCODE(4),=F'-9'        Set return code to -9
         B     RETURN                   Return
RETURN   DS    ØH
         L     R5,RETCODE               Copy the return code
         LR    R15,R5                   Load return code
         EDCEPIL
         LTORG
DETAWORK EDCDSAD
```

```
PARMØ     DS    F                    Address of incoming parms
DETATCB   DS    F                    Address of TCB
DETASTAE  DS    F                    STAE option indicator address
RETCODE   DS    F                    Return code
DBL1      DS    2D                   Dbl word work area
DBL2      DS    2D                   Dbl word work area
WORKLEN   EQU   *-DETAWORK
RØ        EQU   Ø
R1        EQU   1
R2        EQU   2
R3        EQU   3
R4        EQU   4
R5        EQU   5
R6        EQU   6
R7        EQU   7
R8        EQU   8
R9        EQU   9
R1Ø       EQU   1Ø
R11       EQU   11
R12       EQU   12
R13       EQU   13
R14       EQU   14
R15       EQU   15
          END
```

Rudy Douglas
System Programmer (Canada)

VTAM tuning statistics

As is commonly known, tuning is the process of balancing the network load among resources so as to eliminate congestion on any one resource. One needs to tune VTAM in order to get optimal service while using the least amount of resources. One of the objectives of tuning VTAM is to use storage in the host processor and in the communication controller more efficiently and to lessen the load on the host processor. The picture may get clearer if you regard the elements of a network as a series of related storage spaces. A host processor provides storage for VTAM and application programs. Communication controllers contain storage for NCPs and

related programs. Cluster controllers and programmable peripheral nodes also contain storage space. In the case of VTAM we know that it must use storage to build control blocks to keep track of sessions, blocks of data, and other information. If your system needs exceed your storage capacity, you might experience degraded response times. Thus, in order to use the storage more efficiently, you should avoid allocating too many VTAM buffers in the host and choose a more appropriate buffer size in the communication controller. In a similar fashion, an NCP can exceed its storage capacity if it receives more data from other parts of the network than it can send out.

In general, to tune your environment, you need to be able to collect information about your network. You need to know how to gather and analyse tuning statistics, determine the amount of coattailing (when more than one piece of data is being transferred in or out of the host with a single I/O operation) taking place in your network, analyse slowdown conditions, and monitor your common storage areas. VTAM provides several operator commands and other facilities to help you gather this information. Some statistics pertinent to VTAM internal performance activity and resource utilization are collected by VTAM application programs known as monitors. After a monitor obtains the data, it can report information to its end users. VTAM can also record tuning statistics about some of its activities. You can use these statistics to adjust VTAM and NCP variables to improve performance. You can use tuning statistics to gather information on communications between VTAM and any of the following channel-attached nodes:

- Communication controllers
- Adjacent host processors
- SNA cluster controllers.

GATHERING TUNING STATISTICS

By using the TNSTAT start option in the VTAM start-up parameters (member ATCSTR00 on the dataset specified by

the VTAMLST DD card in the VTAM start-up procedure) or the MODIFY TNSTAT command, you can collect data that will help you set the proper values on resource definition operands that control VTAM I/O operations in your system. TNSTAT need not be specified in the VTAM start list to later activate tuning statistics. You may use the display tuning statistics command (**D NET,TNSTAT**) to determine several things: global setting for collecting tuning statistics; whether tuning statistics data is directed to the system console; the time interval between tuning statistics summaries; whether TRLE-controlled devices are collecting statistics and if so, the TRLE names; and whether devices subsequently activated will collect tuning statistics.

You can use VTAM tuning statistics to gather information on the following connections:

- SNA controllers
- Channel-to-channel
- Multipath channel
- TCP.

Tuning statistics can be activated or inactivated for all devices simultaneously (global tuning statistics), and tuning statistics can also be activated or inactivated based on a Transport Resource List Entry name (TRLE tuning statistics). When a TRLE is first activated, the tuning statistics state for that TRLE is set to the global tuning statistics state. For instance, if global tuning statistics are active, TRLE tuning statistics are active for that TRLE. The System Management Facility (SMF) is required to record tuning statistics. Tuning statistics can optionally be displayed at the system console using the CNSL operand, and statistics are always recorded in the appropriate tuning statistics file. This file is an SMF dataset. The tuning statistics record is SMF record type 50. The format depends on the resource for which the tuning I/O operation is collected. The tables 41, 42, 43, 44 and 45 in section 10 of *z/OS V1R4.0 CS: SNA Network Implementation Guide* (SC31-8777-02)

show the formats that can be present in a tuning statistics record. A single set of VTAM tuning statistics can be enough to indicate how a network is operating. However, these statistics become more valuable as you compare sets of values over time to see trends or the effects caused by changing buffer pool specifications and operands. You should analyse tuning statistics before and after making any change that might affect system performance.

CODE

In order to extract VTAM tuning statistics information from SMF data, I have constructed a three-part job stream. In the first part (DUMP50), SMF records 50 are extracted from the SMF weekly dataset to a file, which can be used as a base of archived records.

Please note that the sorting of SMF data may issue an error message (ICE204A), set a return code of 16 and terminate if it detects an incomplete spanned record. In order to overcome this potential obstacle, DFSORT's SPANINC=RC4 option was used to remove the incomplete spanned records. It should be noted that SPANINC=RC0 tells DFSORT (Release 14) to issue a warning message, set a return code of 0, and eliminate all incomplete spanned records it detects. Valid records (that is, complete spanned records) are recovered and written to the output dataset, while SPANINC=RC4 does the same thing as SPANINC=RC0, but with a return code of 4 instead of 0. The shipped default is SPANINC=RC16.

In the second step (COPY50) previously extracted records (selection being defined by INCLUDE's condition) are sorted and copied to a file, which is the input to analysis and reporting VTAMSTAT EXEC invoked in the last step (VTAMST).

There are six reports generated by this REXX EXEC. The first one is *Tuning statistics report for SNA controllers*, which provides information about the state of data-transfer operations between VTAM and one channel-attached SNA controller

(communication or cluster controller). Please remember that VTAM uses channel programs to send data to SNA controllers. The amount of data that VTAM can read in one operation depends on the number of buffers used by a read channel program and on the size of each buffer. Each record of this report contains statistics that cover the time period since the last tuning statistics record was written for that controller or channel-to-channel connection. There are several things to take into account when reviewing this report:

- *Cntl* is the name of the user-defined channel-attached SNA cluster controller or the name of the channel link that attaches the communication controller for which the statistics are gathered. For a VTAM-generated channel-link name, this field contains the channel unit address followed by -L.
- The maximum number of dump-load-restart requests (DLRMAX) number refers to the entire domain, not to the SNA controller named in the report. This value can be used to determine the proper setting for the DLRTCB start option, which determines how many dump-load-restart requests can be processed concurrently. If DLRMAX consistently exceeds DLRTCB, it indicates that VTAM is serializing requests on the available TCBs and that performance might be affected.
- The total number of read channel programs issued (CHRD) to read data does not include the read that informs the cluster controller to clear its buffers.
- The total number of attention interrupts received from a controller (ATTN) includes the total number of read attentions (RDATN).
- Several performance notes are added, a calculation of which was based on recommendations one may find described in chapter 10.1.2.5, 'Analyzing Tuning Statistics', of the manual mentioned above.

The second report, *Tuning statistics report for channel-to-*

channel adapters, provides information about VTAM performing I/O operations across a channel-to-channel link. In this case, VTAM uses channel programs to send data to other hosts.

VTAM channel-to-channel I/O operations, which are similar to SNA controller operations, are impacted by factors such as VTAM I/O buffer size, the channel delay, and the maximum number of buffer pages on the CTC definitions. The amount of data that VTAM can read in one operation depends on the number of buffers used by a read channel program and on the size of each buffer. Each record of this report contains information about the state of data transfer operations between two VTAMs (using channel-to-channel adapters). By analysing statistics provided by this report, you can select I/O buffer sizes, data transfer delay, and other options that can improve performance of these I/O operations. During periods of the day when it is most likely that you will experience performance problems, turn on the tuning statistics. For channel-to-channel attachments, turn on tuning statistics at each processor by using the following operator command:

```
F NET,TNSTAT,CNSL,TIME=1.
```

Again, there are several things to take into account when reviewing this report:

- The number of channel programs issued (CHNRM) that VTAM used to send data to the node on the other side of the adapter will be greater than or equal to the sum of write triggers. The difference between CHNRM and the sum of the write triggers represents the following: the number of channel programs with write data that are initiated by an attention (ATTN) from the other host when data was queued, but a channel program with write data could not be triggered. As you increase the value of the DELAY operand for the channel-to-channel adapter, the difference between CHNRM and the sum of the write triggers may be greater.
- ATTN is the number of times a channel program is initiated because the other host has data to send. This

statistic cannot be correlated with any of the other statistics that are provided; it is simply a value indicating the number of attention interrupts. When compared over an interval of time, ATTN usually does not equal the sum of triggers at the other host. VTAM counts only the first event that initiates an I/O operation, and when both hosts try to write at once, one of the hosts receives an attention that is not counted in its tuning statistics.

- There are several triggers one should review. Timers trigger represents the number of times a channel program with write data is started because the period specified for queueing channel-to-channel PIUs has expired (if session traffic is heavy, the desirable value is 0; if session traffic is light, a low value rather than 0 is desirable). Increasing the DELAY operand on the LINE definition statement or using transmission priority 2 may decrease the value of timers. The desirable TIMERS value is 0, but an occasional non-zero value is acceptable. For channel-to-channel attachments**,** if the TIMERS value in any tuning statistics record is too large, deactivate the channel-attachment major node at each host processor, and activate a previously-defined major node in which DELAY=0 has been specified on the LINE definition statement.

  The queue depth limit trigger represents the number of times a channel program is initiated because the queue limit has been reached. This number should be higher than that of the timers trigger. Please note: if DELAY=0 is specified for the channel-to-channel adapter, the TIMERS and QDPTH tuning statistics may be misleading (ie if DELAY=0, *qdpth* indicates the number of channel programs that wrote data to the channel-to-channel adapter. VTAM determines the QDPTH limit based on usage except in the case of DELAY=0; if DELAY=0, *timers* does not increment).

  The capacity limit trigger (BUFCAP) shows the number of times a channel program with write data is initiated because there is enough data to fill the read buffers of the host on

the other end of the channel. The value will also be incremented if a channel program with write data is initiated because of residual PIUs left on the data queue after a channel program with write data containing a full write buffer of data has completed. If BUFCAP is always 0, the other VTAM host has too many read buffers.

The high priority request trigger (PRI) shows the number of times a channel program with write data is started because a high priority PIU is on the outbound channel queue; that is, the PIU is running under transmission priority 2 or is a virtual route pacing response. If this number is high and there is very little transmission priority 2 traffic over this channel, the minimum virtual route window sizes are probably too small. The higher this number is in relation to the sum of TIMERS + QDPTH + BUFCAP, the less outbound coattailing occurs, and the more CPU time is used for each PIU. You can also use VTAM tuning statistics to analyse a virtual route impact on the channel-to-channel connection. A virtual route that uses transmission priority 2 and traverses the channel-to-channel connection causes VTAM immediately to schedule the data transfer operation. Another virtual route pacing response is high-priority traffic, which has the same effect. The PRI tuning statistic in VTAM indicates the number of times that a VTAM channel program is started to transfer this high priority data (TP2 or VR pacing response) to the other VTAM host. If this number is high and the channel-to-channel connection is not used extensively for TP2 traffic, the minimum virtual route pacing window size is probably too small. Also, the higher this number is in relation to the sum of TIMERS, QDPTH, and BUFCAP, the less outbound coattailing occurs.

- You can also analyse the average number of bytes transferred per I/O operation by dividing the total number of bytes transferred (RDBUF) by the number of READ channel programs (CHNRM).

The next report, *Tuning statistics report for TCP connections*, provides information about VTAM performing I/O operations across TCP connections. By analysing data provided by VTAM and calculated statistics, you can improve performance of I/O operations.

The next three reports pertain to multipath channel (MPC) connections using XCF or channel connectivity and it is generated from SMF 50 records of Version 02. This situation presented a bit of a problem, since IBM did not consider them to be three distinct types of records. Therefore, I have investigated the record extension length in the second step (COPY50) so as to get the extension length pertaining to multipath channel connections using XCF records. Please take a note of the COPY50 step output and change the xcfoff value of the VTAMSTAT EXEC accordingly.

The first report from this set is *Tuning statistics report for multipath channel connections using XCF.* As we know, with XCF multipath channel attachment, VTAM uses the MVS XCF signalling facility to send data to other hosts. Each tuning statistics record contains information about the state of data transfer operations between two VTAMs (using multipath channel connections). By analysing statistics provided by this report, you can select I/O buffer sizes and other options that can improve performance of these I/O operations. There are two distinct types of tuning statistics – the first record contains statistics for the entire MPC group while the second record contains statistics for the READ/WRITE subchannel.

The second report from this set is *Tuning statistics report for multipath channel connections*. This is a case of MPC channel connectivity attachment when VTAM performs I/O operations across multiple single-direction channel links. To perform I/O operations VTAM uses channel programs or Direct Memory Access (DMA) when transmitting and receiving data. By analysing statistics provided by VTAM, you can select I/O buffer sizes, data transfer delay, and other options that can improve performance of these I/O operations. There are three

distinct types of tuning statistics: the first record contains statistics for the entire MPC group while the subsequent records contain statistics for each write subchannel, and each read subchannel. There are several things to take into account when reviewing this report:

- VTAM provides the following levels of MPC capability: High-Performance Data Transfer (HPDT) and non-HPDT.
- HPDT MPC connections provide more efficient transfer of data than non-HPDT MPC connections. They do this by using HPDT services to provide the following functions:
  - data packing without data movement: this process decreases consumption of CPU cycles by reducing the internal movement of data, thus increasing the availability of MIPS (million instructions per second) for user processing.
  - chain scheduling of channel programs: this process reduces operating system I/O invocations and CPU overhead.
- HPDT services are available over connections to other nodes that implement HPDT MPC. Applications that use high-performance data transfer services rely on HPDT MPC connections for path length reductions and performance enhancements when sending and receiving data.
- Non-HPDT MPC connections do not use HPDT services. Non-HPDT MPC connections can be considered synonymous with APPN Host-to-Host Channel (AHHC) connections. Non-HPDT multipath channels use a special data block called a sweep that is exchanged with the adjacent host to verify that data has not been lost. A host initiating a sweep request holds all outbound multipath channel transmissions until it receives a sweep reply from the adjacent host. A sweep is initiated when either of the following occurs:

  - a timer expires in the host with the higher subarea number
  - the receive queue depth in either host is excessive.

  The host initiating the sweep sends the sequence number of the last output transmit block. The adjacent host compares this number with its last input transmit block sequence number. The adjacent host then sends a response to the initiating host that includes the adjacent host last output transmit block sequence number. The initiating host makes the same comparison. If the numbers do not match, or the sweep does not complete within a time limit, the multipath channel group will be inactivated. Otherwise, normal flow continues. The tuning statistics contain a count of how many sweeps are initiated by an expired timer and how many are initiated by excessive receive queue depth.

- For HPDT MPC connections, *tsweep* and *qsweep* are always 0.

The last report from MPC connections pertains to OSA-Express connections. Again, there are basically two different types of tuning statistics: the first record contains statistics for the entire MPC group while the subsequent records contain statistics for OSA-Express datapath queues (READ queue as well as WR/x queue). Please note that many of the statistics in this group contain both a count and an overflow. Both the count and overflow are maintained in unsigned 32-bit variables (unless otherwise indicated). Since an unsigned 32-bit variable can only contain a value up to and including 4294967295 ('FFFFFFFF'X), the variable will wrap through 0 if an increment results in this value being exceeded. When such a wrap occurs, the overflow is incremented by 1. Therefore the total count is determined as follows: Total = (overflow * 4294967296) + count.

Sample JCL to execute SMF type 50 data extract and VTAM statistics reporting:

```
//* UNLOAD SMF5Ø RECORDS FROM VSAM OR VBS TO VB                    *
//* Note: change the DUMPIN DSN=your.smfdata to be the name of     *
//* the dataset where you currently have SMF data being            *
//* recorded. It may be either SMF weekly dataset or an active     *
//* dump dataset. If you chose the latter, then prior to           *
//* executing this job, you need to terminate SMF recording        *
//* of the currently active dump dataset for allow the             *
//* unload of SMF records.                                         *
//* Also, change the DCB reference to match the name of your       *
//* weekly SMF dump dataset.                                       *
//DUMP5Ø   EXEC  PGM=IFASMFDP
//DUMPIN   DD    DISP=SHR,DSN=your.smfdata
//DUMPOUT  DD    DISP=(NEW,PASS),DSN=&&SMF5ØOUT,UNIT=SYSDA,
//         SPACE=(CYL,(5,5)),DCB=(your.smfweekly.dataset)
//SYSPRINT DD    SYSOUT=*
//SYSIN    DD    *
    INDD(DUMPIN,OPTIONS(DUMP))
    OUTDD(DUMPOUT,TYPE(5Ø))
/*
//* COPY VBS TO VB, DROP HEADER/TRAILER RECORDS, SORT ON DATE/TIME *
//* Note: change the SMF5Ø DSN=hlq.SMF5Ø.DATA to be the name of    *
//* the dataset you'll use in the last step.                       *
//COPY5Ø·©ŠEXEC PGM=ICETOOL
//TOOLMSG  DD SYSOUT=*
//DFSMSG   DD SYSOUT=*
//RAWSMF   DD DSN=&&SMF5ØOUT,DISP=SHR
//SMF5Ø    DD DSN=hlq.SMF5Ø.DATA,SPACE=(CYL,(x,y)),UNIT=SYSDA,
//         DISP=(NEW,CATLG,KEEP),
//         DCB=(RECFM=VB,LRECL=32756,BLKSIZE=3276Ø)
//REPORT   DD SYSOUT=*
//TOOLIN   DD *
   SORT FROM(RAWSMF) TO(SMF5Ø) USING(SMFI)
   DISPLAY FROM(SMF5Ø) LIST(REPORT)                -
   TITLE('VTAM statistics records collected')      -
   HEADER('SMF Date')  ON(11,4,DT1,E'9999/99/99')  -
   HEADER('SMF time')  ON(Ø7,4,TM1,E'99:99:99')    -
   HEADER('ID')        ON(19,8,CH)                 -
   HEADER('Version')   ON(65,2,CH)                 -
   HEADER('Extension') ON(63,1,BI)
//SMFICNTL DD *
   OPTION SPANINC=RC4,VLSHRT
   INCLUDE COND=(6,1,BI,EQ,5Ø)
   SORT FIELDS=(11,4,PD,A,7,4,BI,A)
/*
//* FORMAT VTAM Statistics TYPE 5Ø records                         *
//* Note: change the SYSEXEC DSN=your.rexx.library to be the name  *
//* of the dataset where you have placed the VTAMSTAT REXX EXEC.   *
//* Also, change the SMF5Ø DSN=hlq.SMF5Ø.DATA to the name of       *
//* the dataset you have created in the previous step.             *
//VTAMST   EXEC PGM=IKJEFTØ1,REGION=ØM
```

```
//SYSEXEC  DD DISP=SHR,DSN=your.rexx.library
//SMF5Ø    DD DISP=SHR,DSN=hlq.SMF5Ø.DATA
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
prof nopref
%VTAMSTAT
/*
```

VTAMSTAT reporting EXEC:

```
/* REXX VTAM Tuning Statistics                                  */
/* Valid subtypes:                                              */
/* Ø1.Channel-to-Channel Adapters                               */
/* Ø2.Multipath Channel Connections Using XCF                   */
/* Ø2.Multipath Channel Connections (MPC and OSA-E)             */
/* Ø3.TCP Connections                                           */
/* Ø4.SNA Controllers                                           */
xcfoff    = ???? /* Extension length pertaining to XCF records:*/
                 /* change ???? with Extension value of version*/
                 /* Ø2 records (see output of step 2 of JCL job*/
interval  = 3ØØ  /* Interval duration in seconds: change to    */
                 /* value you have specified in MODIFY TNSTAT  */
                 /* command                                    */
MAXBFRU   =  1Ø  /* Needs customization                        */
Numeric digits 32
userid=SYSVAR(SYSUID)
/* Part 1: Handle report files allocation & datasets         */
/*         existence                                         */
r5Øsna = userid||'.sna.xrep'
If SYSDSN(r5Øsna) = 'OK'
Then "DELETE "r5Øsna" PURGE"
  "ALLOC FILE(SNARPT) DA("r5Øsna")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(1,1) CATALOG",
  "REUSE LRECL(115) RECFM(F B) "
r5Øtcp = userid||'.tcp.xrep'
If SYSDSN(r5Øtcp) = 'OK'
Then "DELETE "r5Øtcp" PURGE"
  "ALLOC FILE(TCPRPT) DA("r5Øtcp")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(1,1) CATALOG",
  "REUSE LRECL(15Ø) RECFM(F B) "
r5Øctc = userid||'.ctc.xrep'
If SYSDSN(r5Øctc) = 'OK'
Then "DELETE "r5Øctc" PURGE"
  "ALLOC FILE(CTCRPT) DA("r5Øctc")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(1,1) CATALOG",
  "REUSE LRECL(14Ø) RECFM(F B) "
r5Øxcf = userid||'.xcf.xrep'
If SYSDSN(r5Øxcf) = 'OK'
Then "DELETE "r5Øxcf" PURGE"
```

```
  "ALLOC FILE(XCFRPT) DA("r50xcf")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(1,1) CATALOG",
  "REUSE LRECL(120) RECFM(F B) "
r50mpc = userid||'.mpc.xrep'
If SYSDSN(r50mpc) = 'OK'
Then "DELETE "r50mpc" PURGE"
  "ALLOC FILE(MPCRPT) DA("r50mpc")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(1,1) CATALOG",
  "REUSE LRECL(120) RECFM(F B) "
r50osa = userid||'.osa.xrep'
If SYSDSN(r50osa) = 'OK'
Then "DELETE "r50osa" PURGE"
  "ALLOC FILE(OSARPT) DA("r50osa")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(1,1) CATALOG",
  "REUSE LRECL(120) RECFM(F B) "
time   = 0;  tims   = 0;  timc   = 0; timx   = 0
Totsna = 0;  Totctc = 0;  Tottcp = 0; Totxcf = 0
Totmpc = 0;  Totosa = 0;  inrec = 0
/*  Print reports header                                    */
/*  Channel-to-Channel Connections report header            */
cdr.1 = left('Tuning I/O Operations for Channel-to-Channel
Connections',80)
cdr.2 = left(' ',1,' ')
cdr.3  =left('Report produced on',18),
        ||left(' ',1,' ')||left(date(),12),
        ||left('at',3,' ')||left(time(),10)
cdr.4 = left(' ',1,' ')
cdr.5 = left('Reporting interval (sec.):',26)||left(interval,4)
cdr.6 = left(' ',1,' ')
cdr.7 = left('Date        Time',21)  left('ID',8),
        left('CHNRM',8)  left('CHMAX',8)    left('RDBUF',8),
        left('ATTN',6)   left('TIMERS',6)   left('QDPTH',6),
        left('BUFCAP',6) left('PRI',6)      left('SLODN',6),
        left('IPIU',8)   left('OPIU',8)     left('DRLMAX',6),
        left('WRBUF',6)
cdr.8 =left('-',130,'-')
/*  Tuning Statistics Report for MPC using XCF              */
xdr.1 = left('Tuning Statistics File Report for MPC using XCF',99)
xdr.2 = left(' ',1,' ')
xdr.3  =left('Report produced on',18),
        ||left(' ',1,' ')||left(date(),12),
        ||left('at',3,' ')||left(time(),10)
xdr.4 = left(' ',1,' ')
xdr.5 = left('Reporting interval (sec.):',26)||left(interval,4)
xdr.6 = left(' ',1,' ')
xdr.7 = left('Date        Time',21)        left('TRLE',8)   ,
        left('READ',8)   left('WRITE',8)   left("R/W: ",6)  ,
        left('TSWEEP',6) left('QSWEEP',6)  left('TOKEN',6)  ,
        left('INLP',6)   left('ONLP',6)    left('NLPBYC',7) ,
        left('NLPBYCO',7)
```

```
xdr.8 =left('-',1Ø6,'-')
/*  SNA report header                                          */
hdr.1 = left('Tuning Statistics File Report for SNA Controllers',9Ø)
hdr.2 = left(' ',1,' ')
hdr.3  =left('Report produced on',18),
        ||left(' ',1,' ')||left(date(),12),
        ||left('at',3,' ')||left(time(),1Ø)
hdr.4 = left(' ',1,' ')
hdr.5 = left('Reporting interval (sec.):',26)||left(interval,4)
hdr.6 = left(' ',114,' ')
hdr.7 = left('Date        Time',21)||left('Cntl',9),
        left('DLRMAX',7)  left('CHWR',6)   left('CHRD',6),
        left('ATTN',6)    left('RDATN',6)  left('IPDU',6),
        left('OPDU',6)    left('RDBUF',6)  left('SLODN',6),
        left('INLP',6)    left('ONLP',6)   left('BFNLP',5)
hdr.8 =left('-',115,'-')
/*  TCP/IP report header                                       */
hdd.1 = left('Tuning Input/Output (I/O) Operations for TCP
Connections',99)
hdd.2 = left(' ',1,' ')
hdd.3  =left('Report produced on',18),
        ||left(' ',1,' ')||left(date(),12),
        ||left('at',3,' ')||left(time(),1Ø)
hdd.4 = left(' ',1,' ')
hdd.5 = left('Reporting interval (sec.):',26)||left(interval,4)
hdd.6 = left(' ',1,' ')
hdd.7 = left('Date        Time',21)            left('ID',8)      ,
        left('TYPE',7)     left('ATTN',7)      left('RSIO',8)    ,
        left('WSIO',6)     left('INPACKET',8)  left('OTPACKET',8),
        left('ARPACKET',8) left('AWPACKET',8)  left('MAXRCVD',8) ,
        left('MAXSENT',8)  left('INBYTE',7)    left('OUTBYTE',9) ,
        left('ARBYTE',8)   left('AWBYTE',7)
hdd.8 =left('-',15Ø,'-')
 /*                 Main processing loop                      */
 DO FOREVER
   "EXECIO 1 DISKR SMF5Ø"
    inrec = inrec + 1
    IF RC ¬= Ø THEN call End_of_file
 else do
    PARSE PULL record
    PARSE VAR record header 15 rest
    smf5Ørty  = c2d(substr(header,2,1))      /* Record type */
 call SMF5Ø_header
  end
 End
 /* Part 3: End of input file                                  */
End_of_file:
ml.1 =" "
ml.2 ="Multipath Channel Connections legend                                "
ml.3 =" "
```

```
ml.4 ="MPC LINE - the name of the link for which tuning statistics are "
ml.5 ="           being recorded.                                       "
ml.6 ="IPDU     - the total no. of inbound PDUs received.               "
ml.7 ="OPDU     - the total no. of outbound PDUs transmitted.           "
ml.8 ="TSWEEP   - the no. of sweeps initiated as a result of a          "
ml.9 ="           time-out.                                             "
ml.10="QSWEEP   - the no. of sweeps initiated as a result of            "
ml.11="           excessive receive queue depth.                        "
ml.12="DEV      - the hexadecimal subchannel address of the device      "
ml.13="           for which tuning statistics are being recorded. It    "
ml.14="          corresponds to one of the addresses coded on the READ,"
ml.15="           WRITE, or DATAPATH statement on the LINE or TRLE      "
ml.16="           definition statement.                                 "
ml.17="DIR      - the direction of this resource: READ, WRITE, or       "
ml.18="           WR/x (where x is the QDIO write priority level).      "
ml.19="BSIZE    - the maximum buffer size supported by this device.     "
ml.20="MAXBYTES - the no. of bytes used in the largest channel program."
ml.21="           This field provides information about the utilization"
ml.22="           or packing of data into the transmit or receive       "
ml.23="           buffer. As this no. approaches bsize, this indicates  "
ml.24="          that maximum instantaneous utilization of the device's"
ml.25="           buffer has occurred.                                  "
ml.26="SIO      - the no. of start I/O operations counted for the       "
ml.27="           subchannel. This no. is reset each time VTAM reports  "
ml.28="           tuning statistics.                                    "
ml.29="SLOWDOWN - the no. of times slowdown mode has been entered. If  "
ml.30="   slowdown is incrementing, this indicates a lack of available  "
ml.31="  I/O buffers. If DIR = READ, slowdown is incremented every time"
ml.32="   the channel program cannot be reinitiated immediately because"
ml.33="        of lack of I/O buffers to unpack the inbound data.       "
ml.34="BYTECNTO - PDU byte count overflow.                              "
ml.35="BYTECNT  - byte count is the accumulated no. of bytes of PDUs    "
ml.36="    transmitted on the WRITE subchannel or received on the READ  "
ml.37="           subchannel.                                           "
ml.38="INLP     - the total no.of inbound NLPs received.                "
ml.39="ONLP     - the total no.of outbound NLPs transmitted.            "
ml.40="NBYTECTO - NLP byte count overflow.                              "
ml.41="NBYTECT  - the NLP byte count. This value represents the         "
ml.42="           accumulated no. of bytes of NLPs transmitted on the   "
ml.43="           WRITE subchannel or received on the READ subchannel   "
Select
 when Totmpc > 0  then do
   "EXECIO * DISKW MPCRPT (STEM ml.)"
   end
 otherwise do
 mpf.1 = 'Total number of MPC records is 0.'
 mpf.2 = 'No report is being produced.'
   "EXECIO * DISKW MPCRPT (STEM mpf.)"
   end
End
```

```
os.1 ="                                                                 "
os.2 ="OSA specific legend:                                             "
os.3 ="                                                                 "
os.4 ="PCIREAL - real PCI count. The higher the ratio of real PCI to    "
os.5 ="  virtual PCI, the less successful QDIO is at avoiding the    "
os.6 ="  overhead of the system interrupt handler.                    "
os.7 ="PCIREALO- real PCI overflow.                                     "
os.8 ="PCIVIRT - virtual PCI count. The higher the ratio of virtual PCI"
os.9 ="        to real PCI, the more successful QDIO is at avoiding the"
os.10="          overhead of the system interrupt handler.              "
os.11="PCIVIRTO- virtual PCI overflow.                                  "
os.12="SBALCNTI- Storage Block Address List (SBAL) count is accumulated"
os.13="   no. of SBALs used for I/O on the QDIO WRITE priority level or"
os.14="          the QDIO READ data transfer point.                     "
os.15="SBALCNTO- SBAL count overflow.                                   "
os.16="PACKCNT - packet count: represents the accumulated no of packets"
os.17="     transmitted on the QDIO WRITE priority level or received on"
os.18="    the QDIO READ data transfer point. A packet is a single unit"
os.19="          of data presented to the QDIO device driver.           "
os.20="PACKCNTO - packet count overflow.                                "
os.21="SIGACNT  - Signal Adapter (SIGA) count: the accumulated no. of   "
os.22="     SIGA instructions issued for the QDIO WRITE priority level."
os.23="   SIGA tells the QDIO adapter that data is ready to be written."
os.24="SIGACNTO - SIGA count overflow.                                  "
os.25="PCITHRSH - threshold PCI count. A count of zero indicates the    "
os.26="    QDIO device driver is providing sufficient resources to keep"
os.27="           pace with the inbound data stream from the adapter.  "
os.28="PCITHRSO - threshold PCI overflow.                               "
os.29="PCIUNPRD - unproductive PCI count. An unproductive PCI will      "
os.30="          occur when a virtual PCI causes the processing of read"
os.31=" completions for which a real PCI is pending. An unproductive    "
os.32="         PCI indicates the system interrupt handler overhead was"
os.33="         incurred unnecessarily.                                 "
os.34="PCIUNPRO - unproductive PCI overflow.                            "
os.35="rprocdef - read processing deferral count. A read processing     "
os.36="  deferral occurs when the QDIO PCI Exit must defer a read       "
os.37="  completion because a control block cannot be obtained to       "
os.38="  represent the inbound data.                                    "
os.39="RPROCDEO - read processing deferral overflow.                    "
os.40="RREPLDEF - read replenishment deferral count. A read             "
os.41="  replenishment deferral occurs when the QDIO PCI Exit does not  "
os.42="  have enough available read buffers to tack-in a new read.      "
os.43="RREPLDEO - read replenishment deferral overflow.                 "
os.44="NOREADS  - reads exhausted count. This value is incremented by 1"
os.45="  each time the QDIO PCI Exit is invoked and it detects that all"
os.46="  the read buffers are full. A zero total is preferred because   "
os.47="       lack of read buffers may result in the adapter discarding"
os.48="       inbound data. Examination of the read processing and read"
os.49="       replenishment deferral counts may indicate the reason the"
os.50="     QDIO device driver is not providing sufficient resources to"
```

```
os.51="           accept the inbound data from the adapter.            "
os.52="NOREADSO - reads exhausted overflow.                            "
os.53="SBALMAX  - the maximum no. of active SBALs at the completion of "
os.54=" the write initiation process for the QDIO WRITE priority level."
os.55="       The value range is: Ø-128 (Ø - the priority level had no "
os.56="    outbound activity in the interval; 128 - at one point in the"
os.57="     interval ALL the SBALs for the priority level were active)."
os.58="SBALAVG -  the average no. of active SBALs at the completion of "
os.59=" the write initiation process for the QDIO WRITE priority level."
os.6Ø="   This value will be in the range Ø-128 (Ø meaning the priority"
os.61="          level had no outbound activity in the interval and 128"
os.62="     meaning that every time the QDIO write initiator completed,"
os.63="           all 128 SBALs were active).                          "
os.64="QDPTHMAX-  the maximum no. of work elements left on the outbound"
os.65="work queue at the completion of the write initiation process for"
os.66="      the QDIO WRITE priority level. This value will be a no. in"
os.67="           the range Ø-54 or a character constant > 254.        "
os.68="QDPTHAVG-  the average no. of work elements left on the outbound"
os.69=" work queue at the completion of the write initiation process of"
os.7Ø="        the QDIO WRITE priority level. This value will be in the"
os.71="   range Ø-254 or a character constant >254. A nonzero value for"
os.72="    either of these fields suggests the adapter is not accepting"
os.73="    outbound data as fast as the device driver is presenting it."
Select
 when Totmpc > Ø  then do
   "EXECIO * DISKW OSARPT (STEM os.)"
   end
 otherwise do
 opf.1 = 'Total number of OSA-Express records is Ø.'
 opf.2 = 'No report is being produced.'
   "EXECIO * DISKW OSARPT (STEM opf.)"
   end
End
desc.1 = "                                                             "
desc.2 = "LEGEND:                                                      "
desc.3 = "                                                             "
desc.4 = "Cntl - the name of the user-defined channel-attached SNA     "
desc.5 = "cluster controller or the name of the channel link that      "
desc.6 = "attaches the communication controller for which the stats are"
desc.7 = "gathered. For a VTAM-generated channel-link name, this       "
desc.8 = "field contains the channel unit address followed by  -L.     "
desc.9 = "                                                             "
desc.1Ø= "DLRMAX - maximum number of dump-load-restart requests        "
desc.11= "that were awaiting processing or were being processed        "
desc.12= "at one time during the interval. This number refers          "
desc.13= "to the entire domain, not to the SNA controller named        "
desc.14= "in the report.                                               "
desc.15= "                                                             "
desc.16= "CHWR - the total number of write channel programs issued     "
desc.17= "during the interval.                                         "
```

```
desc.18= "                                                                "
desc.19= "CHRD - the total number of read channel programs issued        "
desc.2Ø= "to read data. It does not include the read that informs        "
desc.21= "the cluster controller to clear its buffers.                   "
desc.22= "                                                               "
desc.23= "ATTN - the total number of attention interrupts received       "
desc.24= "from a controller, including the total number of READ          "
desc.25= "ATTENTION (RDATN). The NCP raises an attention interrupt when"
desc.26= "data waiting to be sent to VTAM and no channel operation       "
desc.27= "is active.                                                     "
desc.28= "                                                               "
desc.29= "RDATN - the number of times that VTAM, after reading data,     "
desc.3Ø= "is requested with an attention to read more data.              "
desc.31= "This is a count of the  number of times a VTAM read            "
desc.32= "operation failed to empty the waiting NCP buffers completely."
desc.33= "                                                               "
desc.34= "IPDU - the total number of inbound (to VTAM) PDUs (messages)   "
desc.35= "received from this controller.                                 "
desc.36= "                                                               "
desc.37= "OPDU - the total number of outbound (from VTAM) PDUs sent      "
desc.38= "to this controller.                                            "
desc.39= "                                                               "
desc.4Ø= "RDBUF - the total number of read buffers used.                 "
desc.41= "ie the number of VTAM buffers used for NCP data transfer.      "
desc.42= "                                                               "
desc.43= "SLODN - the total number of times the controller               "
desc.44= "has entered a slowdown condition; for NCP, this is             "
desc.45= "the number of times the CWALL buffer threshold has             "
desc.46= "been reached. In other words, this state is entered when       "
desc.47= "the NCP buffer pool is nearly depleted and prevents            "
desc.48= "acceptance of any further messages, either from the network    "
desc.49= "or from VTAM.                                                  "
desc.5Ø= "                                                               "
desc.51= "INLP - the total number of inbound (to VTAM) NLPs              "
desc.52= "received from this controller.                                 "
desc.53= "                                                               "
desc.54= "ONLP - the total number of outbound (from VTAM) NLPs           "
desc.55= "sent to this controller.                                       "
desc.56= "                                                               "
desc.57= "BFNLP - the total number of read buffers used for NLPs.        "
Select
 when Totsna ¬= Ø  then do
 msn.1 = left(' ',1,' ')
 msn.2 = left('Total number of SNA records: ',29) left(Totsna,4)
   "EXECIO * DISKW SNARPT (STEM msn.)"
   "EXECIO * DISKW SNARPT (STEM desc.)"
   end
 otherwise do
 msn.1 = 'Total number of SNA records is Ø.'
 msn.2 = 'No report is being produced.'
```

```
   "EXECIO * DISKW SNARPT (STEM msn.)"
   end
End
lab.1 ="                                                                  "
lab.2 ="LEGEND:                                                           "
lab.3 ="                                                                  "
lab.4 ="ID  -  the name of the link for which tuning statistics are       "
lab.5 =" being recorded, and is the name specified on the LINE            "
lab.6 =" definition statement in the associated channel-attached major    "
lab.7 =" node.                                                            "
lab.8 ="TYPE - the TCP/IP resource type, which can be one of these:       "
lab.9 ="       CDLC - Channel data link control                           "
lab.1Ø ="       CLAW - Common link access to work stations                 "
lab.11 ="       CTC  - Channel to channel                                  "
lab.12 ="       LCS  - LAN channel station                                 "
lab.13 ="       HYP  - Hyper Channel                                       "
lab.14 ="                                                                 "
lab.15 ="RSIO     - the number of READ start I/Os issued.                 "
lab.16 ="WSIO     - the number of WRITE start I/Os issued.                "
lab.17 ="INPACKET - the number of inbound TCP/IP packets received.        "
lab.18 ="OTPACKET - the number of outbound TCP/IP packets sent.           "
lab.19 ="ARPACKET - the average number of TCP/IP packets received.        "
lab.2Ø ="AWPACKET - the average number of TCP/IP packets sent.            "
lab.21 ="MAXRCVD  - the largest TCP/IP packet received.                   "
lab.22 ="MAXSENT  - the largest TCP/IP packet sent.                       "
lab.23 ="INBYTE   - the number of inbound TCP/IP bytes received.          "
lab.24 ="OUTBYTE  - the number of outbound TCP/IP bytes sent.             "
lab.25 ="ARBYTE   - the average number of TCP/IP bytes received.          "
lab.26 ="AWBYTE   - the average number of TCP/IP bytes sent.              "
lab.27 ="                                                                 "
lab.28 ="For CLAW resources only:                                         "
lab.29 ="                                                                 "
lab.3Ø ="READCCW  - the number of READ CCWs used.                         "
lab.31 ="WRITECCW - the number of WRITE CCWs issued.                      "
lab.32 ="SIOs R/W - No. of READ/WRITE SIOs.                               "
lab.33 ="APPEND   - the number of WRITE appends done.                     "
lab.34 ="PCICNT   - the number of PCI interrupts received.                "
Select
 when Tottcp ¬= Ø  then do
 mtn.1 = left(' ',1,' ')
 mtn.2 = left('Total number of TCP records: ',29) left(Tottcp,4)
   "EXECIO * DISKW TCPRPT (STEM mtn.)"
   "EXECIO * DISKW TCPRPT (STEM lab.)"
   end
 otherwise do
 mtn.1 = 'Total number of SNA records is Ø.'
 mtn.2 = 'No report is being produced.'
   "EXECIO * DISKW TCPRPT (STEM mtn.)"
   end
End
```

```
ctc.1  ="                                                              "
ctc.2  ="LEGEND:                                                       "
ctc.3  ="                                                              "
ctc.4  ="ID -    provides the name of the link through which the tuning"
ctc.5  ="        statistics are taken.                                 "
ctc.6  ="CHNRM - the number of channel programs issued that VTAM used  "
ctc.7  ="    to send data to the node on the other side of the adapter."
ctc.8  ="CHMAX - Ø because all channel programs are the same size.     "
ctc.9  ="RDBUF - the total number of input bytes transferred           "
ctc.1Ø ="        during the measurement period.                        "
ctc.11 ="ATTN  - the number of times a channel program is initiated    "
ctc.12 ="        because the other host has data to send.              "
ctc.13 ="TIMERS- the number of times a channel program with write data "
ctc.14 ="        is started because the period specified for queuing   "
ctc.15 ="        channel-to-channel PIUs has expired.                  "
ctc.16 ="                                                              "
ctc.17 ="QDPTH - the number of times a channel program is initiated    "
ctc.18 ="        because the queue limit has been reached.             "
ctc.19 ="        This number should be higher than TIMERS.             "
ctc.2Ø ="BUFCAP- the number of times a channel program with write data "
ctc.21 ="        is initiated because there is enough data to fill the "
ctc.22 ="        read buffers of the host on the other end of the      "
ctc.23 ="        channel.                                              "
ctc.24 ="PRI - the number of times a channel program with write data is"
ctc.25 ="        started because a high priority PIU is on the outbound"
ctc.26 ="        channel queue; that is, the PIU is running under      "
ctc.27 =" transmission priority 2 or is a virtual route pacing response"
ctc.28 ="SLODN - the number of times that this VTAM had channel        "
ctc.29 ="  programs with write data blocked by a slowdown condition in "
ctc.3Ø ="  the other VTAM.                                             "
ctc.31 ="IPIU  - the number of inbound PIUs. The average number of PIUs"
ctc.32 ="        for each channel program can be calculated from the   "
ctc.33 ="        sending side as OPIU / (CHNRM + CHMAX).               "
ctc.34 ="OPIU  - the number of outbound PIUs. The average number of    "
ctc.35 =" output PIUs for each channel program with write data can be  "
ctc.36 ="        calculated as OPIU / (CHNRM + CHMAX).                 "
ctc.37 ="DLRMAX- indicates the maximum number of dump-load-restart     "
ctc.38 ="        requests that were awaiting processing or were being  "
ctc.39 ="        processed at one time during the interval. This number"
ctc.4Ø ="        refers to the entire domain, not to the SNA controller"
ctc.41 ="        named in the report.                                  "
ctc.42 ="WRBUF - total number of output bytes transferred              "
ctc.43 ="        during the measurement period.                        "
Select
 when Totctc > Ø  then do
   "EXECIO * DISKW CTCRPT (STEM ctc.)"
   end
 otherwise do
 mcg.1 = 'Total number of CTC records is Ø.'
 mcg.2 = 'No report is being produced.'
```

```
   "EXECIO * DISKW CTCRPT (STEM mcg.)"
   end
End
xcflab.1 ="                                                              "
xcflab.2 ="LEGEND:                                                       "
xcflab.3 ="                                                              "
xcflab.4 ="TRLE   - TRLE name                                            "
xcflab.5 ="READ   - Number of read records                               "
xcflab.6 ="WRITE  - Number of write records                              "
xcflab.7 ="R/W    - Number of READ/WRITE records                         "
xcflab.8 ="TSWEEP - The number of sweeps initiated during a time-out"
xcflab.9 ="QSWEEP - The number of sweeps initiated due to excessive "
xcflab.1Ø ="         receive queue depth.                                "
xcflab.11 ="TOKEN  - The XCF token MVS assigned to the adjacent VTAM"
xcflab.12 ="INLP   - Number of inbound NLPs                              "
xcflab.13 ="ONLP   - Number of outbound NLPs                             "
xcflab.14 ="NLPBYC - NLP byte count                                      "
xcflab.15 ="NLPBYCO- NLP byte count overflow                             "
xcflab.16 ="XCF id - XCF identifier                                      "
xcflab.17 ="RDWR   - READ/WRITE indicator                                "
xcflab.18 ="BSIZE  - The maximum buffer size supported by the device"
xcflab.19 ="MAXBYTES-number of bytes used in the largest channel
program"
xcflab.2Ø ="RCVD   - Number of received bytes                            "
xcflab.21 ="RCVDO  - Receive byte overflow count                         "
xcflab.22 ="SEND   - Sent byte count                                     "
xcflab.23 ="SENDO  - Send byte overflow count                            "
Select
 when Totxcf > Ø  then do
   "EXECIO * DISKW XCFRPT (STEM xcflab.)"
   end
 otherwise do
 mcf.1 = 'Total number of XCF records is Ø.'
 mcf.2 = 'No report is being produced.'
   "EXECIO * DISKW XCFRPT (STEM mcf.)"
   end
End
 /* Close & free all allocated files  */
 "EXECIO Ø DISKR SMF5Ø  (FINIS"
 "EXECIO Ø DISKR SNARPT (FINIS"
 "EXECIO Ø DISKR TCPRPT (FINIS"
 "EXECIO Ø DISKR CTCRPT (FINIS"
 "EXECIO Ø DISKR XCFRPT (FINIS"
 "EXECIO Ø DISKR MPCRPT (FINIS"
 "EXECIO Ø DISKR OSARPT (FINIS"
  say "Channel-to-Channel Adapters report ..........."  r5Øctc
  say "Multipath Channel Connections: XCF report ...."  r5Øxcf
  say "Multipath Channel Connections: MPC report ...."  r5Ømpc
  say "Multipath Channel Connections: OSA-E report .."  r5Øosa
  say "TCP Connections report ......................."  r5Øtcp
```

```
  say "SNA Controllers report......................."  r5Øsna
 "FREE FILE(SMF5Ø SNARPT TCPRPT CTCRPT XCFRPT MPCRPT OSARPT)"
 EXIT Ø
SMF5Ø_header:
 /* Part 2: Basic section of tuning statistics records      */
smf5Ø tme = smf(c2d(substr(header,Ø3,Ø4)))  /* Time record was written */
smf5Ødte = substr(c2x(substr(header,Ø7,Ø4)),3,5)
                                          /* Date record was written */
smf5Øsid = substr(header,11,Ø4)           /* System Identification   */
SMF_date = left(Date('N',smf5Ødte,'J'),11)
smf5Ørty = substr(rest,47,2)               /* Record subtype          */
    Select
      when smf5Ørty = "Ø1" then call VTAM_CTC
      when smf5Ørty = "Ø2" then call VTAM_MPC
      when smf5Ørty = "Ø3" then call VTAM_TCP
      when smf5Ørty = "Ø4" then call VTAM_SNA
      otherwise nop
    End                                       /* of select */
return
VTAM_CTC:
 /* Ø1: Tuning Statistics for Channel-to-Channel Adapters   */
Totctc = Totctc +1
 /*  Print CTCA report header                                */
Select
 when Totctc = 1 then do
   "EXECIO * DISKW CTCRPT (STEM cdr.)"
   end
 otherwise nop
End
Select
  when timc = smf5Ø tme then linc = left(' ',8,' ')
  otherwise                  linc = left(smf5Ø tme,8)
End
    PARSE var rest
ctcactc = substr(rest,1,8)         /* CTCA name (ID)        */
ctcadlr = c2d(substr(rest,9,4))    /* Dump-Load*Restart*request count */
ctcachn = c2d(substr(rest,13,4)) /*Normal-sized ch. pgm. count (CHNRM)*/
ctcachl = c2d(substr(rest,17,4)) /*Large-sized ch. pdm. count (CHMAX) */
ctcaatr = c2d(substr(rest,21,4))   /* Attentions (total) (ATTN)       */
ctcawbf = c2d(substr(rest,25,4))   /* No. of write buffers used(WRBUF)*/
ctcaipi = c2d(substr(rest,29,4))   /* No. of PIUs inbound (IPIU)      */
ctcaopi = c2d(substr(rest,33,4))   /* No. of PIUs outbound (OPIU)     */
ctcardb = c2d(substr(rest,37,4))   /* Total input bytes used (RDBUF)  */
ctcaslw = c2d(substr(rest,41,4))   /* No. of slowdowns (SLODN)        */
ctcaex1 = c2d(substr(rest,45,1))   /* CTCA extension length           */
ctcacat = c2d(substr(rest,46,1))   /* CTCA attachment type            */
ctcattg = c2d(substr(rest,49,4))
                               /*Ch.pgm starts - timer (TIMERS) trigger*/
ctcaqdp = c2d(substr(rest,53,4)) /*Ch.pgm starts - queue depth (QDPTH)*/
ctcabcy = c2d(substr(rest,57,4))
```

```
                              /* Ch.pgm starts - dest.capacity (BUFCAP)*/
ctcapri = c2d(substr(rest,61,4)) /*Ch.pgm starts - high priority (PRI)*/
ctca.1  = right(Date('N',smf5Ødte,'J'),11)  linc,
           left(ctcactc,8)      right(ctcachn,8) ,
          right(ctcachl,8)      right(ctcardb,8) ,
          right(ctcaatr,6)      right(ctcattg,6) ,
          right(ctcaqdp,6)      right(ctcabcy,6) ,
          right(ctcapri,6)      right(ctcaslw,6) ,
          right(ctcaipi,8)      right(ctcaopi,8) ,
          right(ctcadlr,6)      right(ctcawbf,6)
 "EXECIO * DISKW CTCRPT (STEM ctca.)"
  timc = smf5Ø tme
 /*  The average number of PIUs during the interval         */
Select
 when ctcaipi > Ø then  aipi =avg(ctcaipi)      /* Avg.no. IPIU */
 otherwise              aipi = ' '
End
Select
 when ctcaopi > Ø then  aopi =avg(ctcaopi)      /* Avg.no. OPIU */
 otherwise              aopi = ' '
End
 /* The average no. of bytes transferred per I/O operation  */
Select
 when ctcardb > Ø then avgioby = format((ctcardb/ctcachn),5,3)
 otherwise             avgioby = ' '
End
timesum = ctcattg + ctcaqdp + ctcabcy + ctcapri
timesum1= ctcattg + ctcaqdp + ctcabcy
Select
  when ctcachn > Ø then CØØ = format((ctcachn/timesum),5,3)
  otherwise             CØØ = ''
End
Select
  when ctcattg = Ø then CØ1 = 'Heavy session traffic'
  otherwise             CØ1 = 'Light session traffic'
End
Select
  when ctcapri > timesum1 then CØ2 = 'Outbound coattailing decreasing'
  otherwise                    CØ2 = ' '
End
/* Find the average number of output PIUs for each channel */
/* program with write data.                                */
CØ3 = ctcaopi/(ctcachn + ctcachl)
/* Print performance notes                                 */
perc.1 = left('Performance notes ',2Ø),
         left('Send data index:',16)  left(CØØ,6)
perc.2 = left(' ',2Ø,' ') left('Sess.traffic:',16)    left(CØ1,3Ø)
perc.3 = left(' ',2Ø,' ') left('Coattailing:',16)     left(CØ2,3Ø)
perc.4 = left(' ',2Ø,' ') left('Avg.I/O bytes :',16)  left(avgioby,9)
perc.5 = left(' ',2Ø,' ') left('Avg.I/O weight:',16)  left(CØ3,7)
```

```
perc.6 = left(' ',2Ø,' ') left('Avg. IPIU     :',16)  left(aipi,7)
perc.7 = left(' ',2Ø,' ') left('Avg. OPIU     :',16)  left(aopi,7)
perc.8 = left(' ',2Ø,' ')
Select
 when ctcachn > Ø then do
   "EXECIO * DISKW CTCRPT (STEM perc.)"
    end
  otherwise nop
End
return
VTAM_TCP:
 /* Ø3: Tuning Statistics for TCP Connections               */
  Tottcp = Tottcp +1
 /*  Print TCP/IP report header                            */
Select
 when Tottcp = 1 then do
  "EXECIO * DISKW TCPRPT (STEM hdd.)"
   end
 otherwise nop
End
Select
  when time = smf5Ø tme then line = left(' ',8,' ')
  otherwise                  line = left(smf5Ø tme,8)
End
    PARSE var rest
tcpiplne = substr(rest,1,8)              /* TCP line name             */
tcpipcwr = c2d(substr(rest,13,4))        /* No.of write channel pgm.  */
tcpipcrd = c2d(substr(rest,17,4))        /* No.of read channel pgm.   */
tcpipatn = c2d(substr(rest,21,4))        /* Attentions (total)        */
tcpipmsn = c2d(substr(rest,25,4))       /*Largest outbound packet sent*/
tcpipmrc = c2d(substr(rest,29,4))        /* Largest packet received   */
tcpipex3 = c2d(substr(rest,45,1))        /* Extension length          */
tcpippui = c2d(substr(rest,65,4))        /* Inbound packet count      */
tcpippuo = c2d(substr(rest,69,4))        /* Outbound packet count     */
tcpipbyi = c2d(substr(rest,73,4))        /* Inbound byte count        */
tcpipbyj = c2d(substr(rest,77,4))       /*Inbound byte count, overflow*/
tcpipbyo = c2d(substr(rest,81,4))        /* Outbound byte count       */
tcpipbyk = c2d(substr(rest,85,4))      /*Outbound byte count, overflow*/
tcpipleg = c2d(substr(rest,89,1))        /* TCP legacy type           */
 Select
  when tcpipleg = 16 then legacy = 'CTC'
  when tcpipleg = 32 then legacy = 'LCS'
  when tcpipleg = 48 then do
     legacy = 'CLAW'
     tcpipint = c2d(substr(rest,94,4))   /* No. of PCI interrupts     */
     tcpiprcc = c2d(substr(rest,98,4))   /* No. of READ CCWs completed*/
     tcpipwcc = c2d(substr(rest,1Ø2,4))  /*No. of WRITE CCWs completed*/
     tcpipwap = c2d(substr(rest,1Ø6,4))  /* No. of WRITE appends      */
     tcpiprws = c2d(substr(rest,11Ø,4))  /* No. of READ/WRITE SIOs    */
    end
```

```
  when tcpipleg = 64 then legacy = 'CDLC'
  when tcpipleg = 8Ø then legacy = 'HYPER'
  when tcpipleg = 96 then legacy = 'SameHost'
  otherwise nop
 End                                        /* of select */
 /* Calculate the average per second                      */
  arpacket = avg(tcpippui)      /* avg. no of TCP/IP packets received */
  awpacket = avg(tcpippuo)      /* avg. no.of TCP/IP packets sent     */
  arbyte   = avg(tcpipbyi)      /* avg. no.of TCP/IP bytes received   */
  awbyte   = avg(tcpipbyo)      /* avg. no.of TCP/IP bytes sent       */
tcp.1 = right(Date('N',smf5Ødte,'J'),11)  line,
        left(tcpiplne,8)     left(legacy,6)   ,
        right(tcpipatn,4)    right(tcpipcrd,8),
        right(tcpipcwr,8)    right(tcpippui,8),
        right(tcpippuo,8)    right(arpacket,8),
        right(awpacket,8)    right(tcpipmrc,8),
        right(tcpipmsn,8)    right(tcpipbyi,8),
        right(tcpipbyo,8)    right(arbyte,8)  ,
        right(awbyte,8)
 "EXECIO * DISKW TCPRPT (STEM tcp.)"
  time = smf5Ømtme
claw.1 = left('Additional info:',16),
         left('READCCW: ',8) right(tcpiprcc,8)
claw.2 = left(' ',16,' ') left('WRITECCW: ',1Ø) right(tcpipwcc,8)
claw.3 = left(' ',16,' ') left('SIOs R/W: ',1Ø) right(tcpiprws,8)
claw.4 = left(' ',16,' ') left('APPEND:   ',1Ø) right(tcpipwap,8)
claw.5 = left(' ',16,' ') left('PCIICNT:  ',1Ø) right(tcpipint,8)
 Select
  when tcpipleg = 48 then do
   "EXECIO * DISKW TCPRPT (STEM claw.)"
  end
  otherwise nop
 End
return
VTAM_SNA:
 /* Ø4: Tuning Statistics for SNA Controllers              */
  Totsna = Totsna +1
 /*  Print SNA report header                               */
Select
 when Totsna = 1 then do
   "EXECIO * DISKW SNARPT (STEM hdr.)"
   end
 otherwise nop
End
    PARSE var rest
sna04nme = substr(rest,1,8)  /* Locally attchd intelligent controller */
sna04dlr = c2d(substr(rest,9,4))   /* Dump-Load*Restart*request count */
sna04cwr = c2d(substr(rest,13,4))  /* No.of write channel programs    */
sna04crd = c2d(substr(rest,17,4))  /* No.of read channel programs     */
sna04atn = c2d(substr(rest,21,4))  /* Attentions (total)              */
```

```
sna04atr = c2d(substr(rest,25,4))  /* Attentions on read              */
sna04pui = c2d(substr(rest,29,4))  /* Pio's inbound                  */
sna04puo = c2d(substr(rest,33,4))  /* Pio's outbound                 */
sna04buf = c2d(substr(rest,37,4))  /* Read buffers used              */
sna04sld = c2d(substr(rest,41,4))  /* Times NCP entered slowdown     */
sna04ex4 = c2d(substr(rest,45,1))  /* Extension length               */
sna04inl = c2d(substr(rest,69,4))  /* No. of inbound NLPs (INLP)     */
sna04onl = c2d(substr(rest,73,4))  /* No. of outbound NLPs (ONLP)    */
sna04rnl = c2d(substr(rest,77,4))  /* No. of NLP read buffers        */
/* Compare the number of attention interrupts with         */
/* channel READs (A01) - inbound coattailing.              */
Select
  when sna04atn = 0        then A01 ='                  '
  when sna04atn > sna04crd then A01 ='A normal condition'
  when sna04atn = sna04crd then A01 ='CPU util.overhead '
  otherwise                     A01 ='CPU util.increased'
End
Select
  when sna04atn > 0 then Incoix = format((sna04atn/sna04crd),5,3)
  otherwise              Incoix = ' '
End
/* The inbound coattailing index (B01)                     */
Select
   when sna04pui > 0 then Icx = format((sna04pui/sna04crd),5,3)
   otherwise              Icx ='0'
End
/* Analyse IOBUF size (A02)                                */
Select
  when sna04buf = 0            then A02 = '               '
  when sna04buf > (2*sna04pui) then A02 = 'IOBUF too small'
  when sna04buf =  sna04pui    then A02 = 'IOBUF too large'
  otherwise                         A02 = '  '
End
/*  Analyse data transfer operations (A03) VTAM <----> SNA */
Inbdt =  MAXBFRU*sna04crd         /* inbound data transfer */
/* pkk is a ratio of Inbound data transfer and the number  */
/* of channel READs. It should be close to 1. Otherwise,   */
/* data is not being transferred for all the channel       */
/* command words (CCWs) in the read channel program.       */
Select
  when sna04buf > 0 then pkk = format((Inbdt/sna04buf),5,3)
  otherwise              pkk ='0'
End
Select
  when  sna04buf < Inbdt then A03 = 'MAXBFRU too large '
  otherwise                  A03 = '      '
End
/* Examine the read attention (RDATN) information (A04)    */
Select
  when sna04atr = 0              then A04 = 'MAXBFRU or IOBUF too large'
```

```
  when sna04atr > (sna04crd*0.1) then A04 = 'MAXBFRU or IOBUF too small'
  otherwise                          A04 = ' '
End
/* The read attention (RDATN) index                          */
K = sna04crd*0.1
Select
  when sna04atr = 0  then  readix = sna04atr
  when sna04atr > (sna04crd*0.1) then readix = format((sna04atr/k),5,3)
  otherwise               readix =' '
end
/* Analyse the outbound data transfer operation (A05) -      */
/* the outbound coattailing index                            */
Select
  when sna04puo > sna04cwr then A05 = 'VTAM using cpu cycles blockage'
  otherwise                     A05 = ' '
End
Select
   when sna04puo > 0 then Ocx = format((sna04puo/sna04cwr),5,3)
   otherwise              Ocx ='0'
End
/* NCP slowdown  (A06)                                       */
Select
   when sna04sld > 0 then a06 = 'NCP delay parameter is too high'
   otherwise              A06 = ' '
End
/*  I/O index   (read vs write: A07)                         */
Select
   when  sna04crd > 0 then A07 = format((sna04crd/sna04cwr),5,3)
   otherwise               A07 = ' '
End
  Select
    when tims = smf50tme then lins = left(' ',8,' ')
    otherwise                 lins = left(smf50tme,8)
  End
sna.1 = right(Date('N',smf50dte,'J'),11)   lins,
        left(sna04nme,9)      right(sna04dlr,4),
       right(sna04cwr,6)      right(sna04crd,6),
       right(sna04atn,6)      right(sna04atr,6),
       right(sna04pui,6)      right(sna04puo,6),
       right(sna04buf,6)      right(sna04sld,6),
       right(sna04inl,6)      right(sna04onl,6),
       right(sna04rnl,6)
 "EXECIO * DISKW SNARPT (STEM sna.)"
drop sna.
 tims = smf50tme
/* Print performance notes                                   */
perf.1 = left('Performance notes ',20),
         left('CPU impact:',11) left(Incoix,9) left(' - ',3),
         left(A01,18)
perf.2 = left(' ',20,' ') left('In coat.ix:',11)  left(Icx,9)
```

```
perf.3 = left(' ',2Ø,' ') left('VTAM block:',11)  left(Ocx,9),
         left(' - ',3)    left(AØ5,3Ø)
perf.4 = left(' ',2Ø,' ') left('I/O index:',11)   left(AØ7,9)
perf.5 = left(' ',2Ø,' ') left('IOBUF size:',12)  left(AØ2,4Ø)
perf.6 = left(' ',2Ø,' ') left('Transfer:',11)    left(pkk,9),
         left(' - ',3)    left(AØ3,3Ø)
perf.7 = left(' ',2Ø,' ') left('Read attn:',12)   left(AØ4,5Ø)
perf.8 = left(' ',2Ø,' ') left('NCP Slow:',12)    left(AØ6,5Ø)
perf.9 = left(' ',2Ø,' ')
Select
  when snaØ4buf ¬= Ø  then do
   "EXECIO * DISKW SNARPT (STEM perf.)"
    end
  otherwise nop
End
return
VTAM_MPC:
 /* Ø2: Multipath Channel Connections                        */
 /* Customization is needed here:                            */
 /* replace the 'xcfoff' by extension length pertaining to  */
 /* subtype Ø2. The value of extension length you have just */
 /* got displayed by ICETOOL in a previous step.             */
End
    PARSE var rest
    smf5Øex2 = c2d(substr(rest,45,1))  /* Extension length  */
    Select
      when smf5Øex2 = "xcfoff" then call MPC_XCF
      otherwise                     call MPC_CHC
    End                                        /* of select */
return
MPC_XCF:
 /* Ø2: Multipath Channel Connections using XCF              */
Totxcf = Totxcf + 1
 /*  Print MPC XCF report header                             */
Select
 when Totxcf = 1 then do
   "EXECIO * DISKW XCFRPT (STEM xdr.)"
   end
 otherwise nop
End
  Select
    when timx = smf5Ømme then linx = left(' ',8,' ')
    otherwise                linx = left(smf5Ømme,8)
  End
    PARSE var rest
xcf5Ømrl = substr(rest,1,8)        /* TRLE name               */
xcf5Ømvs = c2d(substr(rest,13,8))  /* MVS token               */
xcf5Øexx = c2d(substr(rest,45,1))  /* Extension length        */
xcf5Øctc = c2d(substr(rest,46,1))  /* CTCA attachment type    */
smf5Ørty = substr(rest,47,2)              /* Record subtype
```

```
/* ------------ common: first and next record   --------*/
xcf50xyi = c2d(substr(rest,77,4))  /* Number of inbound NLPs (INLP)  */
xcf50xyj = c2d(substr(rest,81,4))  /* Number of outbound NLPs (ONLP) */
xcf50xyo = c2d(substr(rest,85,4))  /* NLP byte count                 */
xcf50xyk = c2d(substr(rest,89,4))  /* NLP byte count overflow        */
  Select
    when xcf50ctc = 1 then call XCF_first
    when xcf50ctc = 3 then call XCF_next
    otherwise nop
  End
return
XCF_first:
 /* Ø2: Multipath Channel Connections using XCF - group summ */
    PARSE var rest
xcf50fts = c2d(substr(rest,21,4))  /* Number of timer sweeps     */
xcf50fqs = c2d(substr(rest,25,4))  /* Number of queue sweeps     */
xcf50fnw = c2d(substr(rest,29,4))  /* Number of write records    */
xcf50fnr = c2d(substr(rest,33,4))  /* Number of read records     */
xcf50frw = c2d(substr(rest,37,4))  /* Number of READ/WRITE rec.s */
 /* The 1st record contains stat for the entire XCF group        */
xcff.1 = right(Date('N',smf50dte,'J'),11)   linx,
         left(xcf50trl,8)  ,            /* TRLE name             */
         right(xcf50fnr,8) ,            /* Number of read rec.   */
         right(xcf50fnw,8) ,            /* Number of write rec.  */
         right(xcf50frw,8),             /* No.of READ/WRITE rec. */
         right(xcf50fts,6) ,            /* TSWEEP                */
         right(xcf50fqs,6) ,            /* QSWEEP                */
         right(xcf50mvs,1Ø),            /* TOKEN                 */
         right(xcf50xyi,8) ,            /* INLP                  */
         right(xcf50xyj,8) ,            /* ONLP                  */
         right(xcf50xyo,8) ,            /* NLP byte count        */
         right(xcf50xyk,8)              /* NLP byte count overflw*/
 "EXECIO * DISKW XCFRPT (STEM xcff.)"
 timx = smf50tme
 drop xcff.
return
XCF_next:
 /* Ø2: Multipath Channel Connections using XCF - next rec  */
    PARSE var rest
xcf50nrb = c2d(substr(rest,21,4))   /* Number of received bytes*/
xcf50nbo = c2d(substr(rest,25,4))   /* Receive byte overflow   */
xcf50nxc =     substr(rest,29,4)    /* XCF identifier          */
xcf50nbz = c2d(substr(rest,33,4))   /* Transmit block size     */
xcf50nrw =     substr(rest,37,4)    /* READ/WRITE indicator    */
xcf50nsb = c2d(substr(rest,41,4))   /* Sent byte count         */
xcf50nso = c2d(substr(rest,49,4))   /* Send byte overflow count*/
xcf50nxz = c2d(substr(rest,61,4))   /* Max transmit size       */
 /* The 2nd record contains stats for the READ/WRITE subchannel*/
xcfn.1 = left('Subchannel R/W data:',2Ø)     left('XCF id.',8),
         left(xcf50nxc,5) left('RDWR: ',8)   left(xcf50nrw,5)
```

```
xcfn.2 = left(' ',20,' ') left('BSIZE: ',8)  right(xcf50nbz,7)
xcfn.3 = left(' ',20,' ') left('MAXBYTES',8) right(xcf50nxz,7)
xcfn.4 = left(' ',20,' ') left('RCVD:',8)    right(xcf50nrb,7)
xcfn.5 = left(' ',20,' ') left('RCVDO:',8)   right(xcf50nbo,7)
xcfn.6 = left(' ',20,' ') left('SEND:',8)    right(xcf50nsb,7)
xcfn.7 = left(' ',20,' ') left('SENDO:',8)   right(xcf50nso,7)
xcfn.8 = left(' ',20,' ') left('INLP :',8)   right(xcf50xyi,7)
xcfn.9 = left(' ',20,' ') left('ONLP :',8)   right(xcf50xyj,7)
xcfn.10= left(' ',20,' ') left('NLPBC:',8)   right(xcf50xyo,7)
xcfn.11= left(' ',20,' ') left('NLPBCO:',8)  right(xcf50xyk,7)
xcfn.12= left(' ',20,' ')
         "EXECIO * DISKW XCFRPT (STEM xcfn.)"
drop xcfn.
return
MPC_CHC:
 /* 02.Multipath Channel Connections (MPC and OSA-E)        */
    PARSE var rest
ctcatyp  = c2d(substr(rest,46,1))   /* CTCA attachment type */
  Select
    when ctcatyp  = 1 then call MPC_group
    when ctcatyp  = 2 then call MPC_chan
    when ctcatyp  = 4 then call MPC_osae
    otherwise nop
  End
return
MPC_group:
 /* 02.Multipath Channel Connections (MPC and OSA-E) group  */
    PARSE var rest
mpc_line = substr(rest,1,8)        /* MPC line name                    */
dlrmax   = c2d(substr(rest,9,4))   /* Dump-Load*Restart*request count */
ipiu     = c2d(substr(rest,13,4))  /* No.of inbound PIUs               */
opiu     = c2d(substr(rest,17,4))  /* No.of outbound PIUs              */
tsweep   = c2d(substr(rest,21,4))  /* No.of timer sweeps               */
qsweep   = c2d(substr(rest,25,4))  /* No.of queue sweeps               */
wrrec    = c2d(substr(rest,29,4))  /* No.of write records              */
readrec  = c2d(substr(rest,33,4))  /* No.of read records               */
inlp     = c2d(substr(rest,77,4))  /* Number of inbound NLPs (INLP)    */
onpl     = c2d(substr(rest,81,4))  /* Number of outbound NLPs (ONLP)   */
nbytect  = c2d(substr(rest,85,4))  /* NLP byte count                   */
nbytecto = c2d(substr(rest,89,4))  /* NLP byte count overflow          */
nosasec  = c2d(substr(rest,93,4))  /* No. of OSA-E datapath queues     */
 /* The 1st record contains stat for the entire group             */
 /* for both types of connections (MPC Channel and MPC OSA-E)     */
mpcgr.1 = left('Statistics for the entire MPC group',50),
          right(Date('N',smf50dte,'J'),11)
mpcgr.2 = left('MPC line name: ',22)       left(mpc_line,8)
mpcgr.3 = left('Dump*Load*Restart: ',22)   right(dlrmax,7)
mpcgr.4 = left('Number of read rec.:',22)  right(readrec,8)
mpcgr.5 = left('Number of write rec.:',22) right(wrrec,8)
mpcgr.6 = left('TSWEEP:',22)               right(tsweep,6,)
```

```
mpcgr.7 = left('QSWEEP:',22)               right(qsweep,6)
mpcgr.8 = left('INLP  :',22)               right(ipiu,8)
mpcgr.9 = left('ONLP  :',22)               right(opiu,8)
mpcgr.10= left('NLP byte count: ',22)      right(nbytect,8)
mpcgr.11= left('NLP byte count over: ',22) right(nbytecto,8)
mpcgr.12= left(' ',2,' ')
return
MPC_chan:
 /* Ø2.Multipath Channel Connections - the subsequent rec.  */
 /* CTCA attachment type is X'Ø2'                           */
Totmpc = Totmpa + 1;
    PARSE var rest
mpc_line = substr(rest,1,8)        /* MPC line name                    */
dlrmax   = c2d(substr(rest,9,4))   /* Dump*Load*Restart request count */
devadr   =     substr(rest,29,4)   /* Device address                   */
bsize    = c2d(substr(rest,33,4))  /* Transmit block size              */
dir      =     substr(rest,37,4)   /* Subchannel polarity              */
bytecnt  = c2d(substr(rest,41,4))  /* Transmit byte count              */
bytecnto = c2d(substr(rest,49,4))  /* Overflow byte count              */
slowdown = c2d(substr(rest,53,4))  /* Slowdown frequency               */
sio      = c2d(substr(rest,57,4))  /* No.of SIO issued                 */
maxbytes = c2d(substr(rest,61,4))  /* Max transmit size                */
inlp     = c2d(substr(rest,77,4))  /* Number of inbound NLPs (INLP)    */
onpl     = c2d(substr(rest,81,4))  /* Number of outbound NLPs (ONLP)   */
nbytect  = c2d(substr(rest,85,4))  /* NLP byte count                   */
nbytecto = c2d(substr(rest,89,4))  /* NLP byte count overflow          */
 /* The subsequent records contain statistics for each write*/
 /* and read subchannel.                                    */
mpchn.1 = left('Statistics for each read and write subchannel',7Ø)
mpchn.2 = left('MPC line name: ',22)        left(mpc_line,8)
mpchn.3 = left('Dump*Load*Restart: ',22)    right(dlrmax,7)
mpchn.4 = left('Device address :',22)       right(devadr,8)
mpchn.5 = left('Subchannel polarity :',22)  right(dir,8)
mpchn.6 = left('Transmit block size : ',22) right(bsize,6)
mpchn.7 = left('Transmit byte count :',22)  right(bytecnt,8)
mpchn.8 = left('Overflow byte count :',22)  right(bytecnto,8)
mpchn.9 = left('Slowdown frequency  :',22)  right(slowdown,8)
mpchn.10= left('No.of SIO issued    :',22)  right(sio,8)
mpchn.11= left('Max transmit size   :',22)  right(maxbytes,8
mpchn.12= left('INLP  :',22)                right(ipiu,8)
mpchn.13= left('ONLP  :',22)                right(opiu,8)
mpchn.14= left('NLP byte count: ',22)       right(nbytect,8)
mpchn.15= left('NLP byte count over: ',22)  right(nbytecto,8)
 /*  Print MPC group stat first and then statistics         */
 /*  for each read and write of subchannel.                 */
         "EXECIO * DISKW MPCRPT (STEM mpcgr.)"
         "EXECIO * DISKW MPCRPT (STEM mpchn.)"
drop mpcgr.
return
MPC_osae:
```

```
 /* Ø2.Multipath Channel OSA-Express Connections.           */
 /* CTCA attachment type is X'Ø4'.                          */
 /* Statistics for each OSA-Express datapath queue.         */
 /* Print the 1st record which contains stat for the entire */
 /* group (connections type is: MPC OSA-E.                  */
 Totosa = Totosa + 1;
"EXECIO * DISKW OSARPT (STEM mpcgr.)"
drop mpcgr.
    PARSE var rest
dir      =     substr(rest,37,4)     /* Subchannel polarity */
  Select
    when dir  = "READ"  then call MPC_osaer
    otherwise               call MPC_osaew
  End
return
MPC_osaer:
 /* Statistics for OSA-Express READ datapath queue.         */
    PARSE var rest
mpc_line = substr(rest,1,8)        /* MPC line name                    */
dlrmax   = c2d(substr(rest,9,4))   /* Dump*Load*Restart request count */
dir      =     substr(rest,37,4)   /* Subchannel polarity              */
bytecnt  = c2d(substr(rest,41,4))  /* Transmit byte count              */
bytecnto = c2d(substr(rest,49,4))  /* Overflow byte count              */
inlp     = c2d(substr(rest,77,4))  /* Number of inbound NLPs (INLP)    */
onpl     = c2d(substr(rest,81,4))  /* Number of outbound NLPs (ONLP)   */
nbytect  = c2d(substr(rest,85,4))  /* NLP byte count                   */
nbytecto = c2d(substr(rest,89,4))  /* NLP byte count overflow          */
pcirealo = c2d(substr(rest,97,4))  /* Real PCI overflow                */
pcireal  = c2d(substr(rest,1Ø1,4)) /* Real PCI count                   */
pcivirto = c2d(substr(rest,1Ø5,4)) /* Virtual PCI overflow             */
pcivirt  = c2d(substr(rest,1Ø9,4)) /* Virtual PCI count                */
pcithrso = c2d(substr(rest,113,4)) /* Threshold PCI overflow           */
pcithrsh = c2d(substr(rest,117,4)) /* Threshold PCI count              */
pciunpro = c2d(substr(rest,121,4)) /* Unproductive PCI overflow        */
pciunprd = c2d(substr(rest,125,4)) /* Unproductive PCI count           */
rprocdeo = c2d(substr(rest,129,4)) /* Processing deferrals overflow    */
rprocdef = c2d(substr(rest,133,4)) /* Processing deferrals             */
rrepldeo = c2d(substr(rest,137,4)) /* Replenishment deferrals overflow*/
rrepldef = c2d(substr(rest,141,4)) /* Replenishment deferrals          */
noreadso = c2d(substr(rest,145,4)) /* Reads exhausted overflow         */
noreads  = c2d(substr(rest,149,4)) /* Reads exhausted                  */
sbalcnto_r c2d(substr(rest,143,4)) /* Storage Block Addr.list count
overflow */
sbalcnt_r= c2d(substr(rest,147,4)) /* Storage Block Addr.list count    */
packcnto = c2d(substr(rest,151,4)) /* Packet count overflow            */
packcnt  = c2d(substr(rest,155,4)) /* Packet count                     */
 /* Print statistics for OSA-Express READ datapath queue.   */
osarh.1 = left('Statistics for OSA-Express READ datapath queue',7Ø)
osarh.2 = left('MPC line name: ',22)           left(mpc_line,8)
osarh.3 = left('Dump*Load*Restart: ',22)       right(dlrmax,7)
```

```
osarh.4 = left('Subchannel polarity:',22)      right(dir,4)
osarh.5 = left('Transmit byte count: ',22)     right(bytecnt,7)
osarh.6 = left('Overflow byte count:',22)      right(bytecnto,7)
osarh.7 = left('No.of inbound NLPs: ',22)      right(inlp,7)
osarh.8 = left('No.of outbound NLPs:',22)      right(onpl,7)
osarh.9 = left('NLP byte count     :',22)      right(nbytect,7)  ,
          left('Overflow :',10)                right(nbytecto,7)
osarh.10= left('Real PCI count:',22)           right(pcireal,7)  ,
          left('Overflow: ',10)                right(pcirealo,7)
osarh.11= left('Virtual PCI count :',22)       right(pcivirt,7)  ,
          left('Overflow: ',10)                right(pcivirto,7)
osarh.12= left('Threshold PCI count: ',22)     right(pcithrsh,7) ,
          left('Overflow: ',10)                right(pcithrso,7)
osarh.13= left('Unproductive PCI count: ',22)  right(pciunprd,7) ,
          left('Overflow:',10)                 right(pciunpro,7)
osarh.14= left('Processing deferrals: ',22)    right(rprocdef,7) ,
          left('Overflow:',10)                 right(rprocdeo,7)
osarh.15= left('Replenishment deferrals: ',22) right(rrepldef,7) ,
          left('Overflow: ',10)                right(rrepldeo,7)
osarh.16= left('Reads exhausted: ',22)         right(noreads,7)  ,
          left('Overflow: ',10)                right(noreadso,7)
osarh.17= left('SBAL count: ',22)              right(sbalcnt_r,7),
          left('Overflow: ',10)                right(sbalcnto_r,7)
osarh.18= left('Packet count: ',22)            right(packcnt,7)  ,
          left('Overflow: ',10)                right(packcnto,7)
osarh.19= left(' ',1,' ')
"EXECIO * DISKW OSARPT (STEM osarh.)"
drop osarh.
return
MPC_osaew:
 /* Statistics for OSA-Express WR/x datapath queue.         */
    PARSE var rest
mpc_line = substr(rest,1,8)        /* MPC line name                    */
dlrmax   = c2d(substr(rest,9,4))   /* Dump*Load*Restart request count */
dir      =     substr(rest,37,4)   /* Subchannel polarity              */
bytecnt  = c2d(substr(rest,41,4))  /* Transmit byte count              */
bytecnto = c2d(substr(rest,49,4))  /* Overflow byte count              */
inlp     = c2d(substr(rest,77,4))  /* Number of inbound NLPs (INLP)    */
onpl     = c2d(substr(rest,81,4))  /* Number of outbound NLPs (ONLP)   */
nbytect  = c2d(substr(rest,85,4))  /* NLP byte count                   */
nbytecto = c2d(substr(rest,89,4))  /* NLP byte count overflow          */
sbalmax  = c2d(substr(rest,97,4))  /* Maximum Storage Block Addr.lists*/
sbalavg  = c2d(substr(rest,101,4)) /* Average Storage Block Addr.lists*/
qdpthmax = c2d(substr(rest,105,4)) /* Maximum queue depth              */
qdpthavg = c2d(substr(rest,109,4)) /* Average queue depth              */
sigacnto = c2d(substr(rest,113,4)) /* Signal Adapter overflow          */
sigacnt  = c2d(substr(rest,117,4)) /* Signal Adapter count             */
sbalcnto_w=c2d(substr(rest,121,4)) /* Storage Block Addr.list cnt.ovfl*/
sbalcnt_w= c2d(substr(rest,125,4)) /* Storage Block Addr.list count    */
packcnto = c2d(substr(rest,129,4)) /* Packet count overflow            */
```

```
packcnt  = c2d(substr(rest,133,4)) /* Packet count                    */
 /* Print statistics for OSA-Express WR/x datapath queue.   */
osawh.1 = left('Statistics for OSA-Express WR/x datapath queue',70)
osawh.2 = left('MPC line name: ',22)           left(mpc_line,8)
osawh.3 = left('Dump*Load*Restart: ',22)       right(dlrmax,7)
osawh.4 = left('Subchannel polarity:',22)      right(dir,4)
osawh.5 = left('Transmit byte count: ',22)     right(bytecnt,7)
osawh.6 = left('Overflow byte count:',22)      right(bytecnto,7)
osawh.7 = left('No.of inbound NLPs: ',22)      right(inlp,7)
osawh.8 = left('No.of outbound NLPs:',22)      right(onpl,7)
osawh.9 = left('NLP byte count     :',22)      right(nbytect,7)  ,
          left('Overflow :',10)                right(nbytecto,7)
osawh.10= left('Maximum SBAL :',22)            right(sbalmax,7)
osawh.11= left('Average SBAL :',22)            right(sbalavg,7)
osawh.12= left('Maximum queue depth: ',22)     right(qdpthmax,7)
osawh.13= left('Average queue depth: ',22)     right(dpthavg,7)
osawh.14= left('Signal Adapter count :',22)    right(sigacnt,7)  ,
          left('Overflow: ',10)                right(sigacnto,7)
osawh.15= left('Storage Block Addr.list :',22) right(sbalcnt_w,7),
          left('Overflow: ',10)                right(sbalcnto_w,7)
osawh.16= left('Packet count : ',22)           right(packcnt,7)  ,
          left('Overflow: ',10)                right(packcnto,7)
osawh.17= left(' ',1,' ')
"EXECIO * DISKW OSARPT (STEM osawh.)"
drop osawh.
return
SMF: procedure
 /* REXX - convert a SMF time to hh:mm:ss:hd format          */
arg time
time1 = time % 100
 hh = time1 % 3600;                    hh = right("0"||hh,2)
 mm = (time1 % 60) - (hh * 60);        mm = right("0"||mm,2)
 ss = time1 - (hh * 3600) - (mm * 60); ss = right("0"||ss,2)
 fr = time //  1000;                   fr = right("0"||fr,2)
rtime = hh||":"||mm||":"||ss||":"||fr
return rtime
AVG: procedure expose interval
/* REXX - calculate avg. value   */
arg var
   SELECT
    when var  ¬=  0 Then do
       a  = var / interval
       b  = trunc(a,2)
      end
     otherwise  b = 0.0
   END
 return b
```

Mile Pekic
Systems Programmer (Serbia and Montenegro)

Remote batch command upgrade

The two programs below, srvcmds and client, are upgrades to the code given in the article 'Remote batch command' published in *TCP/SNA Update*, September 2005, issue 59. They add three important functions:

1 Sending remote command to the console.

2 Integrating with job scheduling.

3 Printing the report to a dataset.

The server program (srvcmds) can be installed on more than one z/OS partition, while the client program (client) is installed on only one z/OS partition.

Servers and client must be in the same network.

SRVCMDS

This code is added at the end of the SERVERCMDS code published in the previous article.

```
/conscmds:
cnscmd = parms
 cart   = sysvar('SYSUID')||time()
/*-----------------------------------------------------------------*/
/*  Console Environment                                            */
/*-----------------------------------------------------------------*/
 x = MSG('OFF')
  SD=SYSVAR("SOLDISP")
  USD=SYSVAR("UNSDISP")
  If SD = 'YES' Then
     "CONSPROF SOLDISP(NO)"        /* Console Profile: solicited    */
  If USD = 'YES' Then              /* Unsolicited must be NO to be  */
     "CONSPROF UNSOLDISP(NO)"      /* able to catch command response*/
  "CONSPROF SOLNUM(9999) UNSOLNUM(Ø)"
  if rc <> Ø then do
                   Say "**** Userid" userid "needs CONSOLE authority"
                   "CONSOLE DEACTIVATE"
                   Exit
                  end
```

```
  x = MSG('ON')
  "CONSOLE ACTIVATE NAME(MYNAME)"      /* Activate CONSOLE service */
 if rc <> 0 then do
                  Say "Console Activate RC= " rc
                  "CONSOLE DEACTIVATE"
                 Exit
                 end
   "CONSOLE SYSCMD("cnscmd") CART("cart")"
    getcode = GETMSG('cons_msg.','SOL',cart,,20)
    y=cons_msg.0               /* Send back data in reverse order */
    do i = 1 to cons_msg.0     /* Send back data in reverse order */
    msg.wsock.y=cons_msg.i     /* Send back data in reverse order */
    y = y -1                   /* Send back data in reverse order */
    drop cons_msg.i
    end
    "CONSOLE DEACTIVATE"                 /* Close console session    */
    If SD = 'YES' Then
       "CONSPROF SOLDISP("SD")"    /* Restore soldisp origial value */
    If USD = 'YES' Then
       "CONSPROF UNSOLDISP("USD")" /* Restore unsoldisp origial value */
    x = MSG('OFF')
return cons_msg.0
prep_cmd:
poscmd = pos('CMD:',stringword.wsock)
posdsn = pos('DSN:',stringword.wsock)
if posdsn = 0 then posdsn = length(stringword.wsock) + 1
pcmdp4 = (poscmd + 4)
diff = posdsn - pcmdp4
stringall = substr(stringword.wsock,pcmdp4,diff)
command = word(stringall,1)
lcommand = length(command)
lp1 = lcommand + 1
parms = strip(substr(stringall,lp1))
return
```

CLIENT

```
/*  rexx                                                              */
/*  The program has two target :                                      */
/* 1) Retrieve System info from the servers                           */
/* 2) Send TSO command to the servers:                                */
/*    can be used to manage RACF on other systems                     */
/*    or to send a tso command                                        */
/*    on more systems (tso netstat,ping,listc...)                     */
/*  Syntax :                                                          */
/* client server_ipaddress port_number command                        */
/*EG:                                                                 */
/*client IP:10.10.98.22 PORT:35000 CMD:sysinfo (retrieve system info) */
/*client IP:10.10.98.22 PORT:35000 CMD:TSO lu userid  (Racf command)  */
```

```
/*client IP:10.10.98.22 PORT:35000 CMD:TSO netstat (get tcpip info)   */
/*client IP:10.10.98.22 PORT:35000 CMD:CONSOLE D J,L (Display jobs)   */
/*client IP:10.10.98.22 PORT:35000 CMD:shutdown (Shtdwn remote server)*/
/*client IP:10.10.98.22 PORT:35000  + */
/*       CMD:CONSOLE D SMF DSN:alias.report (The console command    */
/*        will be printed in the file : alias.report)               */
x='SOCKET'('SocketSetStatus')
/***** This call returns the status of the socket set ******
If the socket is connected the result could be :
--> 0 myID Connected Free 17 Used 23
0         -> Return code
myID      -> Socket set ID
Connected -> Socket set is connected
Free 17   -> Number of free sockets in the socket set
Used 23   -> Number of allocated sockets in the socket set
If the socket is NOT connected the result could be :
--> 2005 ESUBTASKNOTACTIVE Subtask not active
This is a error code.
It indicates that the socket it is not allocated
************************************************************/
IF WORD(x,1)='0' THEN DO
   x='SOCKET'('Terminate')
   END
ARG string
PARSE VAR string ipaddress sport command dsnprint
string = strip(string)
sport = strip(sport)
string = strip(string)
swprt = 0
if pos('DSN:',dsnprint) > 0 then call check_input
else do
       ipaddress = strip(substr(ipaddress,4))
       sport = strip(substr(sport,6))
       command = strip(substr(command,5))
     end
/* Debug
say "ip     " ipaddress
say "sport  " sport
say "command" command
say "prtdsn " prtdsn
*/
if swprt = 1 then
                do
                 rc_lis = LISTDSI(prtdsn)  /* Rc=0 file already exist*/
                 if rc_lis = 0 then
                                 do
                                  prtdsn = strip(prtdsn)
                                "alloc dd(prtfile) da("prtdsn") shr reu"
                                  say " "
                                  say "File: "prtdsn" already exist"
```

```
                                    say " "
                                    say "Report printed in: "prtdsn
                                    say " "
                                    say " "
                                   end
                                   else
                                      do
                                       say " "
                                       say " "
                                       say " "
                                       "ALLOC DD(prtfile) da("prtdsn")",
                                       "UNIT(SYSALLDA)" ,
                                       "NEW TRACKS SPACE(9,9)",
                                "REUSE LRECL(8Ø) RECFM(F B) BLKSIZE(312Ø)"
                                       if rc = Ø then
                                         say "File :"prtdsn" allocated"
                                         else do
                                              say "File " prtdsn ,
                                              "not allocated.Rc="rc
                                              exit
                                              end
                                       say " "
                                       say " "
                                      end
                 end
/* Initialize client control information */
rc_exit = Ø
port = sport
/* Initialization */
x='SOCKET'('Initialize','CLICMDS')
/*** This call preallocates the number of sockets in a socket set ***
The result could be : Ø CLICMDS 4Ø TCP1ØØØ1
Ø         -> Return code
CLICMDS   -> Socket set name
4Ø        -> The number of preallocated sockets in a socket set
TCP1ØØØ1  -> The name of the TCP/IP service
************************************************************/
IF WORD(x,1)¬='Ø' THEN DO
   SAY 'Error initializing CLICMDS'
   EXIT
   END
IF ipaddress='NONE' THEN DO
  x='SOCKET'('GetHostId')
/****** This call returns the ipaddress for the current host ****
The result could be : Ø 128.228.1.2
Ø            -> Return code
128.228.1.2  -> Host ipaddress
************************************************************/
  IF WORD(x,1)¬='Ø' THEN DO
     SAY 'Error trying to get host id'
```

```
     SIGNAL clean_up
     END
  ELSE ipaddress=WORD(x,2)
  END
/* Initialize for receiving lines sent by the server. */
x = 'SOCKET'('Socket')
/* This call creates an IPv4 socket in the active socket set ****
 * and returns a socket identification ***
The result could be : Ø 1
Ø            -> Return code
1            -> Socket ID
*****************************************************************/
IF WORD(x,1)¬='Ø' THEN DO
   SAY 'Error issuing socket'
   SIGNAL clean_up
   END
/* Pick up the client socket id */
clisock=WORD(x,2)
/* Get the host name */
x='SOCKET'('GetHostName')
/****** This call returns the name of the host ****
The result could be : Ø PROD
Ø            -> Return code
PROD         -> Host name
*****************************************************************/
IF WORD(x,1)¬='Ø' THEN DO
   SAY 'Error getting host name'
   SIGNAL clean_up
   END
hostname = WORD(x,2)
/* Issue af_inet */
x='SOCKET'('Connect',clisock,'AF_INET' port ipaddress)
/****** Tries to establish a connection to another socket ***
The result could be : Ø
Ø            -> Return code
*****************************************************************/
IF WORD(x,1)¬='Ø' THEN DO
   SAY 'Error issuing AF_INET. Rc :' word(x,1)
   if word(x,1) = 61 then say "==> The server is down"
   SIGNAL clean_up
   END
/* Send the information to the server */
x='SOCKET'('Send',clisock,userid() string)
/****** Sends the outgoing data message to a connected socket **
The result could be : Ø 14
Ø            -> Return code
14           -> Length of the data sent
*****************************************************************/
IF WORD(x,1)¬='Ø' THEN DO
   SAY 'Error issuing send'
```

```
   SIGNAL clean_up
   END
/* Wait for lines sent by the server */
DO FOREVER
/* Read the data. Data is returned as a rc len data field */
x='SOCKET'('Read',clisock)
/****** This call reads up to maxlength bytes of data, **
 **     the default is 10,000 **
The result could be : 0 1613 The IPL LOAD PARM used was 3D21SVM1
0            -> Return code
1613         -> Data length
The IPL LOAD PARM used was 3D21SVM1 --> data
*************************************************************/
IF WORD(x,1) ¬='0' THEN DO
   PARSE VAR x . error
   SAY 'Error issuing recv' error
   SIGNAL clean_up
   END
/* allow for the line being null. Abort the connection if it is. */
IF WORD(x,2)='0' THEN LEAVE
/* Get the actual data */
PARSE VAR x . . dataline
DO UNTIL INDEX(dataline,'0D'x)=0
   PARSE VAR dataline trueline '0D'x dataline
   SAY trueline
   if swprt = 1 then do
                     push trueline
                     "execio 1 diskw prtfile"
                     end
   if pos('UNABLE',trueline) > 0 |,
       pos('ERROR',trueline) > 0 |,
        pos('INVALID',trueline) > 0 |,
         pos('MISSING',trueline) > 0 |,
          pos('NOT FOUND',trueline) > 0 then rc_exit = 99
   END
END
if swprt = 1 then
                do
                "EXECIO 0 DISKW prtfile (FINIS"
 /*              "free dd(prtfile) da("prtdsn")"
                say "rc free  " rc  */
                end
EXIT rc_exit
/* Terminate and exit  */
clean_up:
x='SOCKET'('Terminate','CLICMDS')
/***** This call closes all sockets in the socket set ******
The result could be : 0 CLICMDS
0         -> Return code
CLICMDS   -> Subtask Id
```

```
************************************************************/
RETURN
check_input:
ippre=substr(ipaddress,1,3)
portpre=substr(sport,1,5)
cmdpos = pos('CMD:',string)
cmdpre=substr(string,1,4)
dsnpos = pos('DSN:',string)
diff   = dsnpos - cmdpos
if dsnpos > 0
            then do
                   prtpre = 'DSN:'
                   prtdsn = strip(substr(string,dsnpos+ 4))
                   swprt = 1
                  end
 if cmdpos > 0 then
                  do
                   cmdpre = 'CMD:'
                   command = strip(substr(string,cmdpos,diff))
                  end
ipaddress = strip(substr(ipaddress,4))
sport = strip(substr(sport,6))
command = strip(substr(command,5))
/* debug
say "ip " ipaddress "pre " ippre
say "spo" sport     "portpre " portpre
say "cmd" command   "cmdpre " cmdpre
say "prt" prtdsn    "prtpre " prtpre
*/
if ipaddress = ' ' | sport = ' ' | command = ' ' |,
   prtdsn = ' '  ,
    | ippre ¬= 'IP:' | portpre ¬= 'PORT:' ,
     | cmdpre ¬= 'CMD:' | prtpre ¬= 'DSN:'
   then do
          say " "
          say " "
           say "************************************************"
           say "  Invalid Syntax.The correct syntax is:"
           say "  Example -> IP:1Ø.1Ø.1Ø.3 PORT:1946 CMD:sysinfo +"
           say "               DSN:userid.prt.report"
           say "************************************************"
          say " "
         exit
         end
return
```

Each server must be authorized to execute the console command. The RACF commands are:

- Add the console profile key to the tsoauth class:

```
RDEFINE TSOAUTH CONSOLE
```

- Authorize user1 and user2 to access tsoauth class:

```
PERMIT CONSOLE ACCESS(READ) CLASS(TSOAUTH) ID(user1,user2,etc)
```

Modify the IKJTSO*xx* member of SYS1.PARMLIB library. If these statements are not present, add them:

```
CONSOLE INITUNUM(1000)  /* ALLOCATE COMMAND CONSOLE  */  +
        INITSNUM(1000)  /*                           */  +
        MAXUNUM(10000)  /*                           */  +
        MAXSNUM(10000)  /*                           */
PLATPGM NAMES(          /* ALLOCATE COMMAND CONSOLE  */  +
        CONSPROF)       /*                           */
```

Modify the AUTHCMD NAMES section to add a CONSPROF statement:

```
AUTHCMD NAMES(
   CONSPROF
```

The syntax for using the client is:

```
client IP:server_ipaddress PORT:port_number CMD:command DSN:dataset-name
```

The commands are:

- SYSINFO – this parameter obtains the system report from the server.
- SHUTDOWN– this parameter executes the server shutdown.
- TSO – tsocommand.
- CONSOLE – this parameter sends commands to the console.

The *DSN:dataset-name* parameter redirects the command output to the file specified. If the command started from the client has a return code not equal to zero, the job step return code is set to 99. For example:

```
//useridCL JOB (TSO,SIG),'.JSERVER.',MSGCLASS=R,
//          NOTIFY=userid,REGION=8M,CLASS=A,TIME=1440
//STEP01 EXEC PGM=IKJEFT01,DYNAMNBR=50,REGION=6M
//SYSPROC DD DSN=alias.user.lib,DISP=SHR
//SYSTCPD  DD DSN= alias.user.lib(TCPIPPAR),DISP=SHR
```

```
//SYSTSPRT DD SYSOUT=*
//SYSTSOUT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
PROFILE NOPREFIX
CLIENT IP:172.22.64.31 PORT:1957 CMD:TSO LISTC ENT(alias.lib.jcl)
/*
//STEP02  EXEC PGM=IKJEFT01,DYNAMNBR=50,REGION=6M,
//  COND=(4,LE,STEP01)
//SYSTSPRT DD SYSOUT=*
//SYSTSOUT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
PROFILE NOPREFIX
TSO LISTA
```

If we suppose that alias.lib.jcl does not exist, the step01 return code is 99 and step02 will be flushed.

If we want to change the return code value, edit the client REXX program and modify the 99 value – see line number 252:

```
00252             pos('NOT FOUND',trueline) > 0 then rc_exit = 99
```

Magni Mauro
Systems Engineer (Italy)

March 2003–December 2005 index

Items below are references to articles that have appeared in *TCP/SNA Update* since issue 53, March 2003. References show the issue number followed by the page number(s). Subscribers can download copies of all issues in Acrobat PDF format from Xephon's Web site.

For network printing, Capella Technologies has announced Multi-Host Print (MHP), which is a printer-resident memory module that converts IPDS and SCS datastreams into a format compatible with HP LaserJet printers. The MHP module also includes a forms overlay component for sprucing up SCS reports with graphics.

The first component takes basic SNA Character Stream (SCS) and high-end Intelligent Printer Data Stream (IPDS) output from mainframe and OS/400 servers, and converts it into the Printer Control Language (PCL) format supported by the HP LaserJets.

For further information contact:
URL: www.capellatech.com/documents/MHP%20Press%20Release.pdf.

* * *

AES has announced Version 4.0 of CleverView for cTrace Analysis. The product's real-time tracing capability allows it to identify unexpected complications and facilitate their resolution by visualizing and diagnosing the problem.

Key features include session response time reports that summarize all sessions between a local and remote IP, and a sequence of execution report that provides details of the packets exchanged during a given session. Message data from different sources can be viewed, including EBCDIC and ASCII.

For further information contact:
URL: www.aesclever.com/index.php?f=ctrace&page=solutions.

* * *

Rocket Software has announced Version 2.2 of NetCure, its service-assurance software for enterprises and service providers.

NetCure is designed to reduce the time required to isolate, diagnose, and resolve business infrastructure problems. NetCure also identifies potential performance degradation in applications and networks in advance, automatically notifying appropriate operations personnel or triggering corrective actions before business services are critically impacted.

NetCure discovers both physical and logical network topology including routers, switches, chassis, slot, port, interface, and VLAN configurations. NetCure continuously monitors and detects any changes to the environment and automatically updates its topology model to reflect the changing environment.

For further information contact:
URL: www.rocketsoftware.com/netcure.

* * *

NewEra Software has announced that it will support the IBM Health Checker Framework, an integral part of IBM system availability assurance strategy delivered as part of z/OS V1.7.

NewEra views the Health Checker Framework s an enabler for thousands of System Inspectors currently available in its products, IMAGE Focus and IMAGE Sentry. Their System Inspectors are designed to validate the correctness, integrity and state of z/OS configuration components, most notably the operating system and its subsystems JES, VTAM, and TCP/IP.

For further information contact:
URL: www.newera.com.