165 MVS update

June 2000

In this issue

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 33598
From USA: 01144 1635 33598
E-mail: Jaimek@xephon.com

North American office
Xephon/QNA
PO Box 350100,
Westminster, CO 80035-0100
USA
Telephone: (303) 410 9344
Fax: (303) 438 0290

Contributions
Articles published in *MVS Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*, or you can download a copy from www.xephon.com/contnote.html.

Editor
Jaime Kaminski

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

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

Subscriptions and back-issues
A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.

Printed in England.

Internet resources for systems programmers

INTRODUCTION

Today more and more workers have high-speed Internet connections on their office (and home) workstations, and thus they have access to a plethora of Web sites and newsgroups that can help considerably with some of their tasks.

THE RESOURCES

This article is an attempt to document some of the resources that I have come to use, many on a daily basis, and without which it is becoming difficult to imagine being able to function efficiently. The links and descriptions shown below will provide a useful overview of the resources available. I would recommend 'bookmarking' any links that you may use regularly. This will save you time, because you will not need to re-type the URL each time you access the site.

http://www.ibmlink.ibm.com/

IBMLink is the premier port of entry to a wide range of IBM services. Especially useful on the US IBMLink page are the Announcement Letters and Sales Manual, both of which give searchable descriptions of IBM's offerings in both hardware and software. And although it is called US IBMLink, there seems to be no restriction about accessing these sections from anywhere in the world. The exception to this is ServiceLink, which requires a user-id and password. This gives access to the Problem Resolution, Q/A, and Preventive Service sections of IBMLink. Amongst other facilities, you can search through an APAR database which is very extensive and up-to-date, track APARs and PTFs, order fixes and PTFs, and communicate with IBM support service personnel. The ability to communicate with IBM personnel is almost the equivalent of having a panel of IBM specialists permanently at your disposal to assist diagnosing complex problem situations.

http://www2.s390.ibm.com/bookmgr-cgi/bookmgr.cmd/library

The IBM BookManager(r) BookServer Library is a huge library of IBM manuals. This specific URL is actually the front-end for a search engine that searches book titles, names, or document numbers containing the argument you specify. For instance, entering CICS currently finds 642 manuals. I find it best to use a wide search argument and then use my Web browser's FIND function to look for more specific books in the resulting list. So, if for instance I am looking for CICS for MVS/ESA Version 4 Release 1 Intercommunication Guide, I search for CICS and then use the browser to find Intercommunication. While reading manuals on the Web is not to everyone's liking, and I must count myself amongst those who still prefer to use a hardcopy when I am doing a great deal of reading in one specific manual, the sheer number of titles available makes this an invaluable resource for getting started on researching an issue.

http://www.redbooks.ibm.com/

While the previous link points to some Redbooks, the Redbooks homepage is an especially well designed entry port to the Redworld. Redbooks, Redpieces, and Redpapers are all here, and also information about IBM Residencies, the program whereby one can participate in the team which develops a Redbook.

http://www.s390.ibm.com/os390/bkserv/

Also on the subject of manuals, this is the OS/390 Internet library. Again there is some duplication with the Bookserver Library mentioned above, but this link seems to have the very latest versions of manuals before the other site. For instance, at the time of writing I find OS/390 Version 2 Release 9 manuals here, but only Version 2 Release 8 at the Bookserver Library site.

http://techsupport.services.ibm.com/support/s390

This is similar to the ServiceLink offering mentioned above, but this is available to the public without a user-id/password. It has a searchable database of APARs as well as sections on Hints and Tips, System/390 Technical Documentation, and Enhanced HOLDDATA.

http://www1.s390.ibm.com:80/lspr/lspr.html

This is the IBM System/390 Large Systems Performance Reference site, where IBM publishes the results of the LSPR benchmarks for its own and some competitor systems. At time of writing the latest results available on the site are for Generation 6 Turbo Enterprise Servers (up to the 9672-ZZ7) running OS/390 Version 2 Release 4. There is also a description of the LSPR workloads and methodology. All the information can be downloaded in a PDF file.

http://www-1.ibm.com/support/techdocs/atsmastr.nsf

This is not just for System/390-related products; this site gives access to the IBM Technical Support Technical Information Database. This includes Flashes, such as those from the Washington Systems Center, and IBM White Papers.

http://www.S390.ibm.com/

The marketing side of the System/390 world; everything you ever wanted to know about System/390 complete with attractive graphics.

http://www.s390.ibm.com/marketing/gf225122.html

This is a useful non-technical explanation of the differences between the world of the System/390 enterprise server and the world of the Unix server.

http://www.s390.ibm.com/os390/installation/

This page is currently targeted at OS/390 Version 2 Release 9 installation and ordering, but it has links to information on prior OS/390 releases back to Version 2 Release 5. The OS/390 Version 2 Release 9 Installation Planning Assistant is an interactive version of the manual *OS/390 Version 2 Release 9.0*.

http://www.s390.ibm.com/cfsizer/

This an interactive tool which helps you to estimate structure sizes for IBM products that exploit the Coupling Facility. You select a product from a list (say CICS) and are presented with a list of possible CF structures for that product (say CICS temporary storage). You are then

prompted to enter values for the relevant variables for this structure, and the tool returns suggested sizing and sample IXCMIAPU policy statements.

http://www.s390.ibm.com/rmf/rmfhtmls/rmftools.htm

These are some tools developed by the RMF group, such as a Java edition of the RMF Performance Monitoring of OS/390 product, and the RMF Spreadsheet Reporter Version 4 for Windows NT and 95/98.

http://www.s390.ibm.com/srm/

This is an up-to-date list of IBM System/390 Processor version codes and SRM constants as documented in the *MVS Initialization and Tuning Guide*. It spans systems ranging from the 9221-120 at 83.5008 SU/sec to the (12 way) 9672-ZZ7 at 77701.3356 SU/sec.

http://www-4.ibm.com/software/ts/cics/txppacs/txpc1.html

The IBM CICS SupportPacs are a set of utilities, sample code, and documentation for various functions for CICS implementations on all platforms. Some of the pacs are actually for fee-based services and cannot be downloaded, but others are freeware offerings. The OS/390 pacs cover such topics as Migration Planning for CICS/TS, DBCTL Implementation, Replicating shared data tables across a sysplex, CICSPlex SM (Administration and operation sample utilities), and CICSPlex SM (Sample API programs).

http://www-4.ibm.com/software/ts/mqseries/txppacs/

As above, these are SupportPacs for MQSeries.

http://www.s390.ibm.com/products/oe/bpxa1toy.html

OS/390 Unix System Services Tools and Toys is a large collection of freeware and unsupported packages that are available for download. These are specifically designed for OS/390 Unix by IBM developers and testers.

http://www.hursley.ibm.com/cwuf/

This is not an IBM page, despite the URL. It is the CICS World Wide User Forum, which is a forum for discussion about the requirements

users have in relation to the CICS family of products, and it is maintained by user group organizations. Additionally, an extensive hints and tips section is accessible from this page, for every CICS environment as well as for MQSeries. There seems to be little organization to these tips though, and no search capability.

http://support.cai.com/catotalclientcare.html

Computer Associates Total Client Care site, for CA customers with current maintenance agreements. It provides Web access to CA's centralized client support database. This includes searchable access to Program Temporary Fixes (PTFs) and Product Information Bulletins (PIBs) via the CA Knowledge Base, and the ability to directly download PTFs once identified. But probably the best feature is the direct problem reporting and tracking mechanism whereby one can report problems to and communicate with CA technical support staff in much the same way as using IBM's ServiceLink.

http://frontline.compuware.com/

Compuware Corporation's customer-only on-line technical support site allows you to search for and download PTFs and to post a question or report a problem, but not to have an on-going conversation with Compuware staff on an issue. There are also PDF versions of Compuware product manuals available at this site.

http://www.ecs.landmark.com/

Landmark System Corporation's version of the above. Again customers can search for and download PTFs. However, to report a problem, Landmark customers have to go to the following URL, http://www.support.landmark.com/, where they can also find PDF manuals and technical articles and hints and tips from Landmark technical staff.

http://www.RexxLA.org/

The REXX Language Association is an independent organization dedicated to promoting the use and understanding of the REXX programming language. REXX fans, amongst whom I count myself, will find links to many other sites where mostly freeware REXXcode

is available for as many different functions as you can think of.

http://www.cicscentral.com/

If the name does not say it all, then this description of Bob Juch's site does – CICS Central is the first place to go for information on IBM's CICS.

http://www.yelavich.com/

Bob Yelavich spent some 40 years working at IBM, 30 of those with CICS. This site is a mine of useful and interesting CICS related information. Bob also authors an e-mail newsletter, on a random but very often daily basis, covering CICS related topics.

http://www.mvsbook.fsnet.co.uk/

This site, belonging to David Elder-Vass, has an abridged version of his book *MVS Systems Programming* (McGraw Hill, 1993).

http://www.loriaux.com/s390/

Eric Loriaux's System/390 home page is certainly the most comprehensive collection of System/390 sites and links I have come across.

http://www.watsonwalker.com/

The home page of Cheryl Watson of Watson & Walker Inc, this includes various articles by Cheryl and a downloadable version of Cheryl's Quickstart Service Policy.

http://www.snipix.freeserve.co.uk/hercules.htm

Hercules is a software implementation of S/370 and ESA/390 under Linux on a Pentium PC. Theoretically this allows you to run OS/390 on a PC, but the licensing issues would require a great deal of research first. Currently it is possible to run OS/360 on Hercules, and the site details how to go about getting the necessary resources together to do this.

http://www.xephon.co.uk

I could hardly leave this site out! In addition to downloadable code from all the *Update* series of publications, there are numerous Xephon Report articles in PDF format.

http://www.esj.com/

Enterprise Systems Journal, after Xephon publications, everyone's favourite source of large systems related news.

http://www.nascom.com/index.htm

The Network and Systems Professionals Association, NaSPA produces *Technical Support* magazine the articles go as far back as 1996 and are available in PDF format.

http://www.cbttape.org/

The CBT tape is the granddaddy of all MVS freeware, and is now available in its' entirety from this Web site. There is probably hardly a System/390 site anywhere in the world that does not use some software that originated on the CBT tape, or a systems programmer that did not at least get some good ideas as to how to tackle a knotty problem by viewing some of the material it contains. There are EBCDIC and ASCII format versions of File 1, which is the description of all the other files. The following URLs are mostly pages set up by individuals to share programs, snippets of code, or programming techniques that they have developed to deal with some of the problems that they have encountered while working in the IBM mainframe field. Where possible, I have noted the individual concerned and some major code that they are sharing with the community. There are many utilities which give the systems programmer a quick overview of various parameter settings and PARMLIB member entries such as linklist concatenations, LPAlist concatenations and APF libraries, and the first three of the URLs below all include examples of this functionality:

- Mark Zelden (IPLINFO) http://home.flash.net/~mzelden/mvsutil.html and http://www.mindspring.com/~somebody/.
- Doug Nadel (TASID) http://www.secltd.co.uk/home.htm.

- Scott Enterprise Consultancy (MXI) http://www.best.com/~ldw/mvs/.
- Leonard Woren (TAPEMAP) http://members.home.com/gsf/.
- Gilbert Saint-Flour http://home-5.worldonline.nl/~jjaeger/.
- Jan Jaeger (ZZSA Stand-alone editor) http://etk.com/download/index.htm.
- COBOL tools http://hometown.aol.com/rexxauthor/mainfram.htm. This contains a list of books on various mainframe topics, some written by Gabriel Gargiulo, and links to extracts and extensive information regarding REXX.

http://www.mks.com/s390/gnu/

This site contains a large number of OS/390 Unix utilities.

http://members.aol.com/os390info/

This is a free service to ask questions of system programmers who wish to promote the use of the OS/390 software platform. Responses are sent to your e-mail address. I have not tried this service so I cannot comment on response time.

http://mvshelp.com/

This is a similar service to the previous one, but it takes the form of a bulletin board where you post questions and anyone who has registered can post a reply. Questions are broadly categorized, eg JCL, CICS, VSAM, REXX, and each category has a sub-board which has an assigned moderator.

http://www.mainframes.com/

This is a general systems programmer help site with a large amount of static information on a wide variety of topics useful to the busy sysprog.

NEWSGROUPS AND LISTSERVERS

Perhaps the greatest aid to the systems programmer introduced by the Internet is the newsgroup or listserv. With a permanent high-speed Internet connection, these essentially e-mail-based services take on a new role as an almost instantly available expert assistant. The most widely subscribed general System/390 and OS/390 group is probably IBM-MAIN. Started back in the mid-'80s, IBM-MAIN has become a hugely popular forum for systems programmers and IBM and other software and hardware vendor support and development staff to discuss issues and resolve problems ranging across all aspects of IBM mainframes. The response is phenomenally quick, the only downside perhaps being that if you post a routine question you might be buried by the avalanche of advice from all parts of the globe. On numerous occasions I have seen posts describing a real-time problem which is solved by the collective skills and experience of the list within minutes. The archives of IBM-MAIN, which in themselves represent a huge body of useful information, are also available on-line in a searchable form at: http://bama.ua.edu/archives/ibm-main.html. There is also an unofficial FAQ list maintained for IBM-MAIN at the following URL: http://users.ticnet.com/davea/ibm-main/. This site includes everything required to get onto (or out of) the list as well as answering to some technical questions which come up every few weeks as new users join the list (such as, where do I find documentation for IPOUPDTE?).

There are many other lists which deal with various more specialized aspects of the System/390 world, a few of which I have listed below. How to join these lists is succinctly described on Eric Loriaux's site (mentioned above) at the following URL: http://www.loriaux.com/s390/mailing.html. The topic covered by a list is generally self-explanatory; here are a few which I find especially useful to monitor regularly. These are CICS-L, DB2-L, IMS-L, IBMTCP-L, and MVS-OE.

Channel information

THE PROBLEM

Recently my manager asked me what channels were available on one of our mainframes. Although I used HCD to provide an answer, I started wondering if I could make the information more easily available to others. Plus, because HCD is used on only one LPAR, it does not provide an easy mechanism for users of other machines/ LPARs. Anyway, it seemed worth having a look around to see if there was a more LPAR-friendly mechanism for retrieving channel data. Note that D M=CHP(xx) as an operator command was excluded because it was security restricted and not particularly user-friendly anyway. In the end I came up with two methods.

THE SOLUTIONS

The first is a simple REXX dialog based around extracting information from the ICHPT control block. This is a 256-byte block, addressed via CVTICHPT in the CVT. This block has one byte containing status information for each of the possible channels that can be attached to a processor. (Please see the help panel CHANPH1 for a description of what the bits mean.)

The second, meanwhile, is a much more comprehensive dialog that exploits the macro IOSCHPD to extract not just status information, but also what type a particular channel is (ESCON, CTC, etc).

I have included both in this article as, although the IOSCHPD system is considerably more powerful, the simpler version can often be enough and requires just one REXX and two panels for installation. For that matter, if the data is simply ‘SAY’ed to the screens you can get away with just the REXX and the help panel for information. It therefore should avoid any implementation issues because it can be run from your own user REXX library.

SOLUTION 1

To begin with then, the first dialog, CHANRES. This will return a screen similar to the following, where each channel can easily be read off and its bit status identified:

```
 ———————————— CHANNEL INFORMATION ——— Row 1 to 16 of 16
Command ===>                                          Scroll ===> PAGE

   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00 80_80_E0_80_C0_80_80_80_E0_E0_00_00_80_E0_80_E0
10 E0_80_80_80_80_80_80_80_80_80_80_80_80_80_80_80
20 E0_E0_E0_E0_80_E0_80_80_E0_80_E0_E0_E0_80_E0_E0
30 E0_E0_80_E0_E0_E0_E0_E0_E0_80_80_80_80_80_80_80
40 80_80_80_80_80_80_80_80_00_00_00_00_80_80_80_80
50 80_E0_80_E0_E0_E0_E0_E0_80_E0_E0_E0_E0_E0_E0_80
60 E0_E0_E0_E0_E0_E0_E0_E0_80_80_80_80_80_80_80_80
70 80_80_80_80_80_80_80_80_80_80_80_80_80_80_80_80
80 80_80_80_80_80_80_80_80_80_80_80_80_80_80_80_80
90 E0_E0_E0_E0_E0_80_80_E0_E0_80_E0_80_E0_E0_E0_80
A0 80_E0_E0_E0_E0_E0_E0_E0_E0_80_E0_80_80_E0_E0_E0
B0 80_80_80_80_80_80_80_80_80_80_80_80_80_80_80_80
C0 80_80_80_80_80_80_80_80_00_00_00_00_00_00_00_00
D0 00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00
E0 00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00
F0 00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00


.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
```

This dialog requires the following REXX code.

CHANRES

```
/* rexx */
CVTICHPT=D2X(C2D(STORAGE(10,4))+1232) /* point to cvtichpt */
ichpt_address=D2X(C2D(STORAGE(CVTICHPT,4)))
channels=STORAGE(ichpt_address,256)
ADDRESS ISPEXEC
'TBCREATE CHANNELS NAMES(XX LINE1) NOWRITE REPLACE'
DO x=1 to 256 by 16
line1=c2x(substr(channels,x,16))
   DO y=1 to 15
   line1=INSERT('_',line1,(y*3)-1)
   END
xx=D2X((x-1),2)
'TBADD CHANNELS'
END
```

```
'TBTOP CHANNELS'
‘TBDISPL CHANNELS PANEL(CHANPAN1)’
```

CHANPAN1

```
)Attr Default(%+_)
   ! type(output) intens(high) caps(on ) just(left )
   @ type(output) intens(low) caps(on ) just(left )
   _ type(input) intens(low ) caps(off) just(asis )
)Body  Expand(//)
%-/-/- CHANNEL INFORMATION -/-/-
%Command ===>_zcmd                                / /%Scroll ===>_amt +
+
    ØØ Ø1 Ø2 Ø3 Ø4 Ø5 Ø6 Ø7 Ø8 Ø9 ØA ØB ØC ØD ØE ØF
)Model
@xx!line1
)Init
  .Help = chanph1                /* insert name of tutorial panel  */
  &amt = PAGE
)PROC
)End
```

CHANPH1

```
)ATTR
 ' TYPE(PT)                          /* panel title line             */
)BODY
'———————— Help panel for Channel Display ——————————-
+
+Command ==>_ZCMD        +
+
This panel displays the current bit status for every channel on this
LPAR. Use the low intensity address markers on the panel to calculate
the channel number. Once that has been done, the bits have the following
meaning. Note that combinations of bits are possible, so translate the
value bit by bit.
x'8Ø' .... this channel is capable of accepting a cable.
x'4Ø' .... this channel belongs to this LPAR.
x'2Ø' .... this cannel is online to this LPAR.
x'1Ø' .... this channel is undergoing channel path recovery.
x'Ø8' .... a vary offline is in progress for this channel.
x'Ø4' .... a vary offline is in progress for this channel.
x'Ø2' .... channel path recovery has started its last UCB scan.

According to data areas, the only valid states are X'EØ',X'CØ',X'8Ø',X'ØØ'
X'FØ',X'E8',X'F8'
)PROC
&ZTOP=CHANPANH
```

```
&ZUP=CHANPANH
&ZCONT=CHANPANH
)END
```

THE IOSCHD-BASED SYSTEM

As with dialog 1, the following is an example screen produced by this system. Probably the first thing to notice with this is that not all the channels appear to be shown. This is because the dialog is not just a display, it allows selection of information. In this case the display shows a screen where only those channels that could be assigned (Column VALID contains a Y) are selected. This is done with a SHOW VALID command, which shows all the channels that are VALID (ie have a Y in that column). Any of the columns can be selected in this manner (ie SHOW column-name).

```
————————————— CHANNEL INFORMATION —— Row 1 to 2Ø of 194
Command ===>                                          Scroll ===> PAGE

 ID  Type                              Valid Own   Onlin Rec  Gooff Rfail
Rnear
ØØ  PARALLEL BLOCK MULTIPLEX          Y     N     N     N     N     N     N
Ø1  PARALLEL BLOCK MULTIPLEX          Y     N     N     N     N     N     N
Ø2  PARALLEL BLOCK MULTIPLEX          Y     N     N     N     N     N     N
Ø3  PARALLEL BLOCK MULTIPLEX          Y     N     N     N     N     N     N
Ø4  PARALLEL BLOCK MULTIPLEX          Y     N     N     N     N     N     N
Ø5  PARALLEL BLOCK MULTIPLEX          Y     N     N     N     N     N     N
Ø6  PARALLEL BLOCK MULTIPLEX          Y     N     N     N     N     N     N
Ø7  PARALLEL BLOCK MULTIPLEX          Y     N     N     N     N     N     N
Ø8  COUPLING FACILITY SENDER          Y     Y     Y     N     N     N     N
Ø9  COUPLING FACILITY SENDER          Y     Y     Y     N     N     N     N
ØC  UNKNOWN                           Y     N     N     N     N     N     N
ØD  PARALLEL BLOCK MULTIPLEX          Y     N     N     N     N     N     N
ØE  PARALLEL BLOCK MULTIPLEX          Y     N     N     N     N     N     N
ØF  PARALLEL BLOCK MULTIPLEX          Y     N     N     N     N     N     N
1Ø  PARALLEL BLOCK MULTIPLEX          Y     N     N     N     N     N     N
11  PARALLEL BLOCK MULTIPLEX          Y     N     N     N     N     N     N
12  UNKNOWN                           Y     N     N     N     N     N     N
13  PARALLEL BLOCK MULTIPLEX          Y     Y     Y     N     N     N     N
14  UNKNOWN                           Y     N     N     N     N     N     N
15  UNKNOWN                           Y     N     N     N     N     N     N
```

Each of the columns provides diagnostic information on the state of each channel. Please see the help panel for this dialog (CHANPANH below) for more detail on what each of the columns means.

Installation

This dialog consists of one Assembler program (which needs to be linked AMODE 31 into a library in your TSO STEPLIB concatenation), two panels (one display and one help), which need to be in your ISPPLIB concatenation, and one REXX, which will need to be in your SYSPROC (or SYSEXEC) concatenation. Note that the program does not need to be authorized, but you will need to check that you have APAR OW37043 installed on your system, or the IOSCHPD macro will fail with RCODE 8 reason code 2.

First, the REXX. Note the name is unimportant, and can be your choice. At my site it is called CHANSHOW.

CHANSHOW

```
/* REXX */
rowpos=1 /* primer variable for screen displays */
flag='Y'
/* */
/* Call assembler support routine to obtain relevant information */
/* about the channels.                                           */
/* */
looper:
CALL PATHLIST
ADDRESS ISPEXEC
'TBCREATE CHANNELS NAMES(channel type valid own onlin rec
gooff rfail rnear) NOWRITE REPLACE'
DO x=1 TO 256
channel=D2X(x-1,2)
type=STRIP(path_details.x)
valid=path_valid.x
own=path_owned.x
onlin=path_online.x
rec=path_recovery.x
gooff=path_being_offlined.x
rfail=path_recovery_failed.x
rnear=path_recovery_finishing.x
ADDRESS TSO
INTERPRET "IF "flag"=Y THEN ADDRESS ISPEXEC TBADD CHANNELS"
ADDRESS ISPEXEC
END
/* */
/* Now transfer the variables to ISPF for display purposes */
/* */
'TBTOP CHANNELS'
'TBSKIP CHANNELS NUMBER('rowpos')'
```



```
DROP rowcnt
'TBQUERY CHANNELS ROWNUM('rowcnt')'
IF rowcnt=Ø THEN DO
   zedsmsg='No data selected'
   zedlmsg='None of the channels have the requested column set to Y'
   'SETMSG MSG(ISRZØØ1)'
   END
'TBDISPL CHANNELS PANEL(CHANPAN)'
rowpos=ztdtop   /* keep position */
/* */
/* command processing section */
/* */
IF WORD(ZCMD,1)='SHOW' THEN CALL what_to_show
ELSE IF zcmd='REFRESH' THEN SIGNAL looper
IF reply='END' then EXIT
SIGNAL looper
/* */
what_to_show:
/* */
IF WORDS(zcmd)>2 THEN DO
   zedsmsg='Sorry incorrect show issued'
   zedlmsg='Please specify SHOW followed by the column you wish to
select'
   'SETMSG MSG(ISRZØØ1)'
   RETURN
   END
IF WORDS(zcmd)=1 THEN DO
   flag='Y' /* reset to display all */
   RETURN
   END
flag=WORD(zcmd,2)
SELECT
WHEN flag='VALID' THEN NOP
WHEN flag='OWN' THEN NOP
WHEN flag='ONLIN' THEN NOP
WHEN flag='REC' THEN NOP
WHEN flag='GOOFF' THEN NOP
WHEN flag='RFAIL' THEN NOP
WHEN flag='RNEAR' THEN NOP
OTHERWISE DO
   zedsmsg='Unknown column'
   zedlmsg='All channel information has been shown'
   'SETMSG MSG(ISRZØØ1)'
   flag='Y'
   END
END
RETURN
```

CHANPAN

```
)Attr Default(%+_)
   ! type(output) intens(high) caps(on ) just(left )
   @ type(output) intens(low ) caps(off) just(asis )
)Body  Expand(//)
%-/-/- CHANNEL INFORMATION -/-/-
%Command ===>_zcmd                                  / /%Scroll ===>_amt +
+
 ID  Type                                Valid Own   Onlin Rec  Gooff Rfail
Rnear
)Model
!z  !z                                 !z    !z    !z    !z    !z    !z    !z
)Init
  .Help = CHANPANH               /* insert name of tutorial panel  */
  .ZVARS = '(channel type valid own onlin rec gooff rfail rnear)'
  &amt = PAGE
)PROC
&REPLY = .RESP
)End
```

CHANPANH

```
)ATTR
 ' TYPE(PT)                          /* panel title line            */
 ? TYPE(PIN)                         /* panel instruction line      */
 # TYPE(NT)                          /* normal text attribute       */
 } TYPE(ET)                          /* emphasized text attribute   */
 ! TYPE(DT)                          /* description text            */
 ~ AREA(SCRL)                        /* scrollable area attribute   */
)BODY
'──────── Help panel for Channel Displays ─────────────
+
+Command ==>_ZCMD        +
+
+The Channel display shows the current status of all the possible
+channels that this machine could support.
+
+=============================================================================
~pnarea                                                                      ~
~                                                                            ~
~                                                                            ~
~                                                                            ~
~                                                                            ~
~                                                                            ~
~                                                                            ~
~                                                                            ~
~                                                                            ~
~                                                                            ~
~                                                                            ~
+=============================================================================
```

```
+
%Use ENTER to scroll downwards through the available data.
)AREA pnarea
#
}DESCRIPTION:
+Each of the columns is described as follows:
%ID+    The channel number
%TYPE+  What type of channel corresponds to this channel number. Note
+       that this contains%UNKNOWN+then either this channel does not
+       exist on this machine, or it hasn't been defined for use by
+       this OS390.
%VALID+ If this column is set to Y then this channel physically exists
+       on this machine. Use this in conjunction with the TYPE column
+       to determine what UNKNOWN means.
%OWN+   If this is set to Y then this path issued by this OS930.
%Online+If this is Y then this channel is on-line to this OS390.
%Rec+   If this is Y then this channel is undergoing recovery from a
+       problem.
%Gooff+ If this is Y then this channel is currently going off-line.
%Rfail+ If this is Y then recovery processing for this channel has
+       failed following a force channel off-line command.
%Rnear+ If this is Y then channel recovery processing is nearing
+       completion.
#
}SUBCOMMANDS:
+SHOW:  Using this followed by any of the column names Valid to Rnear
        will cause the display to be limited to only those channels
        having that condition set to Y.
        Should no channels have that condition, an error message is
        shown.
        To reset the display, either issue SHOW on its own, or specify
        an unknown column. This latter option will get a warning message
        but it can be ignored.
#
+REFRESH: this will cause a re-check of all the channels, and will
          redisplay the current data with the latest information.
)PROC
&ZTOP=CHANPANH
&ZUP=CHANPANH
&ZCONT=CHANPANH
)END
```

PATHLIST

```
//your job card
//*
//STEPA EXEC ASMFCL,PARM.LKED='NORENT,NOREUS'
//ASM.SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
//           DD DSN=SYS1.MODGEN,DISP=SHR
//ASM.SYSIN DD *
```

```
PATHLIST TITLE 'REXX FUNCTION TO RETRIEVE CHANNEL DETAILS'
*********************************************************************
*
* THIS ROUTINE ANALYSES ALL THE ATTACHED CHANNELS AND RETURNS THEIR
* DETAILS.
* NOTE: IN ORDER TO ASSEMBLE, APAR OW37Ø43 WILL NEED TO HAVE BEEN
* APPLIED TO THE SYSTEM.
* THE FOLLOWING ARRAYS VARIABLES ARE CREATED FOR EACH CHANNEL
*
* PATH_DETAILS.X ...THE TYPE OF CHANNEL. NOTE X CORRESPONDS TO THE
*                   CHANNEL NUMBER.
* PATH_VALID.X .....SET TO Y IF THIS PATH IS VALID FOR THIS CONFIG
* PATH_OWNED.X .....SET TO Y IF THIS MVS HAS THAT CHANNEL
* PATH_ONLINE.X ....SET TO Y IF THIS PATH ONLINE.
* PATH_RECOVERY.X ..SET TO Y IF PATH RECOVERY IN PROGRESS.
* PATH_BEING_OFFLINED.X ... SET TO Y IF OFFLINE IN PROGRESS
* PATH_RECOVERY_FAILED.X ... SET TO Y IF RECOVERY FAILED
* PATH_RECOVERY_FINISHING.X ...SET TO Y IF RECOVERY NEAR COMPLETION
*
*********************************************************************
        MACRO
        REXREGS
        LCLA &CNT
&CNT    SETA Ø
.LOOP   ANOP
R&CNT   EQU &CNT
&CNT    SETA &CNT+1
        AIF (&CNT LT 16).LOOP
        MEND
        MACRO
        SHOWSET
        AIF (D'SHOW_START).NONEED
        B  BY_SHOW_START
SHOW_START DS ØH
        ST R1Ø,COMRET
        LA 6,COMSHVB
        USING SHVBLOCK,R6
        XC COMSHVB(SHVBLEN),COMSHVB
        XC SHVNEXT,SHVNEXT
        MVI SHVCODE,C'S'
        BR 14
ABEND ØØ1 DS ØH
        ABEND 1   * REQUIRED FOR THE OTHER MACROS. SAVES SOME CODING.
BY_SHOW_START DS ØH
LITLOC  LOCTR
@_UNPACK DC CL16' '
         DC CL8' '  * FILL FIELD
         ORG @_UNPACK+8
@_UNPACKER DC CL8' '
         ORG
```

```
@_DWORD  DS CL8     * USED FOR THE DEBIN FUNCTION
&SYSECT LOCTR
.NONEED ANOP
        BAL 14,SHOW_START
        MEND
        MACRO
  SHOWARAY &LABEL,&ASNAME,&ERR=ABEND001,&LEN=,&SUBARRAY=,&DEBIN=,&LINK=
        PRINT NOGEN
*********************************************************************
*
* MACRO TO CREATE REXX ARRAY VARIABLES
*
* NOTE RESTRICTION: THIS MACRO IS LIMITED TO CREATING UP TO 9,999,999
*                   ENTRIES FOR EACH ARRAY.
*
* MACRO FORMAT:
*       SHOWARAY &LABEL,&ASNAME,&ERR=,&LEN=,&SUBARRAY=,&DEBIN=
* WHERE:
*       &LABEL IS THE NAME OF THE LABEL WHICH ADDRESS THE FIELD FROM
*              WHERE THE DATA TO BE DEFINED IN A REXX VARIABLE IS
*              LOCATED
*       &ASNAME IS THE NAME TO BE ASSIGNED TO THE DATA FOR USE IN REXX
*       &ERR= IS THE LABEL TO BRANCH TO SHOULD AN ERROR OCCUR WHILE
*             CREATING THE REXX VARIABLE. BY DEFAULT IT IS ABEND001
*       &LEN= IF THE DATA AT &LABEL IS NOT DEFINED SUCH THAT THE LENGTH
*             OF THE DATA IS WHAT YOU WANT, SIMPLY ENTER A NUMBER HERE
*             THAT DEFINES THE LENGTH REQUIRED. CAN ALSO BE USEFUL IF
*             NECESSARY TO DUMP OUT A LARGE AREA.
*       &SUBARRAY= IF A MULTI LEVEL ARRAY IS REQUIRED EG A.1.1 THEN
*                  SET THIS VALUE ACCORDINGLY.
*       &DEBIN= IF THE DATA TO BE CREATED IS BINARY, SETTING THIS TO A
*               VALUE WILL CONVERT THE SPECIFIED NUMBER OF BYTES FROM
*               BINARY TO CHARACTER. THE DEFAULT LENGTH FOR THE
*               OUTPUT DATA IS 4 BYTES. IF THIS IS INSUFFICIENT, THEN
*               SPECIFY A SUITABLE &LEN VALUE TO OVERRIDE IT.
*       &LINK= THIS IS A REXX NAME LABLE TO WHICH THE ARRAY COUNT IS
*              LINKED. THE PURPOSE OF THIS IS TO ALLOW A BRANCH OUT
*              OF ARRAY LOOPS WHILE STILL MAINTAINING NUMERIC
*              CONSISTENCY.
*
*********************************************************************
        PRINT GEN
        LCLA &DEFLEN
&DEFLEN SETA 16
        SHOWSET
LITLOC  LOCTR
&LABCHECK SETC '@_&ASNAME&SUBARRAY'
&LINKNAME SETC '@_&LINK'
         AIF   (D'&LABCHECK).BYPASS
         AIF   (T'&SUBARRAY EQ 'O').NORMNAME
```

```
&LABCHECK DC C'&ASNAME..&SUBARRAY'
         AGO   .EOFARRAY
.NORMNAME ANOP
&LABCHECK DC C'&ASNAME'
.EOFARRAY ANOP
&LABCHECK._ARRAY DC C'.       '
&LABCHECK._COUNTER DC PL4'Ø' * COUNTER FIELD FOR THIS ITEM
.BYPASS  ANOP
&SYSECT LOCTR
        AIF (T'&LINK EQ 'O').DOADD
        MVC &LABCHECK._COUNTER,&LINKNAME._COUNTER
        AGO .DOUNPK
.DOADD  ANOP
        AP &LABCHECK._COUNTER,=P'1' * INCREMENT THE COUNTER THIS PASS
.DOUNPK ANOP
        UNPK @_UNPACKER,&LABCHECK._COUNTER * UNPACK THE VALUE
        OI   @_UNPACKER+7,X'FØ'            * REMOVE THE SIGN
* NOW NEED TO WORK OUT THE LENGTH OF THE COUNTER BIT TO ADD TO ARRAY
        L    R15,&LABCHECK._COUNTER        * LOAD THE COUNTER VALUE TO WORK
*                                            OUT THE LENGTH
        SRL  R15,4                         * REMOVE THE SIGN
        XR   R14,R14                       * CLEAR R14 FOR A COUNTER
LOOP&SYSNDX DS ØH
        SRA  R15,4                         * MOVE DIGIT BY DIGIT
        LTR  R15,R15
        BZ   COUNT&SYSNDX
        LA   R14,1(,R14)
        B    LOOP&SYSNDX
COUNT&SYSNDX DS ØH
* NOW ADD COUNT FIELD TO NAME
        LA  R15,@_UNPACKER+7               * POINT TO END OF FIELD
        SR  R15,R14                        * AND COME BACK TO FIRST DIGIT.
        MVC &LABCHECK._ARRAY+1(7),Ø(R15)
        LA 1,&LABCHECK
        ST 1,SHVNAMA
* NOW CALCULATE NEW LENGTH
        LA 1,L'&LABCHECK
        LA 1,2(R14,R1)
        ST 1,SHVNAML
        AIF (T'&DEBIN EQ 'O').NORMLAB
*
*** NOW ALLOW FOR A BINARY CONVERSION
*** FIST CALCULATE THE ICM VALUE
*
&ICM    SETA (1 SLL &DEBIN)-1
        XR R15,R15
        ICM R15,&ICM,&LABEL                  * LOAD THE BINARY VALUE
        CVD R15,@_DWORD                      * CONVERT TO PACKED
        OI  @_DWORD+7,X'ØF'
        UNPK @_UNPACK,@_DWORD
```

```
*
*** IF THE LEN VALUE IS SUPPLIED THIS OVERRIDES THE DEFAULT OF 16
*
        AIF (T'&LEN EQ 'O').SETDEF          * LENGTH NOT SUPPLIED USE DEFLEN
&DEFLEN SETA &LEN                           * RESET DEFLEN TO SUPPLIED LEN
.SETDEF ANOP
        LA R1,@_UNPACK+(16-&DEFLEN)
        ST R1,SHVVALA
        LA R1,&DEFLEN
        AGO .OK
.NORMLAB ANOP
        LA 1,&LABEL
        ST 1,SHVVALA
        AIF (T'&LEN NE 'O').DOLEN
        LA 1,L'&LABEL
        AGO .OK
.DOLEN  ANOP
        LA 1,&LEN
.OK     ANOP
        ST 1,SHVVALL
        LR Ø,1Ø
        LA 1,COMS
        L 15,IRXEXCOM
        BALR 14,15
        LTR 15,15
        BNZ &ERR
        MEND
PATHLIST AMODE 31
PATHLIST RMODE ANY
PATHLIST CSECT
         REXREGS
         BAKR  R14,RØ
         LR    R12,R15
         LA    R11,2Ø48(,R12)               * ESTABLISH ADDRESSABILITY FOR
         LA    R11,2Ø48(,R11)               * UP TO 8K
         USING PATHLIST,R12,R11
*
         LR    R1Ø,RØ                       * R1Ø -> A(ENVIRONMENT BLOCK)
         USING ENVBLOCK,R1Ø
*
         L     R9,ENVBLOCK_IRXEXTE          * R9 -> A(EXTERNAL EP TABLE)
         USING IRXEXTE,R9
*
*
         STORAGE OBTAIN,LENGTH=GETLEN,ADDR=(8)
         USING COMSDS,R8
*
* PREPARE THE REXX AREA FOR USE
*
         XC    COMS(COMSLEN),COMS            * SET TO LOW VALUES
```

```
         LA    R15,COMID
         ST    R15,COMS
         LA    R15,COMDUMMY
         ST    R15,COMS+4
         ST    R15,COMS+8
         LA    R15,COMSHVB
         ST    R15,COMS+12
         LA    R15,COMRET
         ST    R15,COMS+16
         OI    COMS+16,X'8Ø'                    * INDICATE END OF PARMS
         MVC   COMID,=C'IRXEXCOM'
         XR 5,5
         USING PSA,5
*
         L    5,FLCCVT
         USING CVT,5
*
         L   5,CVTICHPT                         * POINT TO THE CHANNEL BLOCK
*
* COMMENCE THE CHANNEL LOOP. NOTE THAT ONLY 256 CHANNELS ARE POSSIBLE
*
          XR R3,R3                              * CLEAR R3 FOR A COUNT
LOOPER DS ØH
          STH R3,PATHPID
*
          IOSCHPD CHPID=PATHPID,DESC=DESCRIBE
*
          SHOWARAY DESCRIBE,PATH_DETAILS
*
          LA R2,Ø(R3,R5)                        * POINT TO BLOCK BYTE
          TM Ø(R2),X'8Ø'                        * IS THIS A VALID CHANNEL
          BC 8,BIT8ØNO                          * NO SO SET A NO
*
          SHOWARAY YES,PATH_VALID
*
          B BIT4ØTRY
*
BIT8ØNO   DS ØH
*
          SHOWARAY NO,PATH_VALID
*
BIT4ØTRY DS ØH
          TM Ø(R2),X'4Ø'               * DOES THIS CHANNEL BELONG TO THIS MVS
          BC 8,BIT4ØNO                          * NO SO SET A NO
*
          SHOWARAY YES,PATH_OWNED
*
          B BIT2ØTRY
*
BIT4ØNO   DS ØH
```

```
*
          SHOWARAY NO,PATH_OWNED
*
BIT2ØTRY DS ØH
          TM Ø(R2),X'2Ø' * IS THIS CHANNEL ONLINE?
          BC 8,BIT2ØNO   * NO SO SET A NO
*
          SHOWARAY YES,PATH_ONLINE
*
          B BIT1ØTRY
*
BIT2ØNO   DS ØH
*
          SHOWARAY NO,PATH_ONLINE
*
BIT1ØTRY DS ØH
          TM Ø(R2),X'1Ø' * IS THIS CHANNEL UNDERGOING RECOVERY?
          BC 8,BIT1ØNO   * NO SO SET A NO
*
          SHOWARAY YES,PATH_RECOVERY
*
          B BIT Ø8TRY
*
BIT1ØNO   DS ØH
*
          SHOWARAY NO,PATH_RECOVERY
*
BIT Ø8TRY DS ØH
          TM Ø(R2),X'Ø8' * IS THIS CHANNEL GOING OFFLINE?
          BC 8,BIT Ø8NO   * NO SO SET A NO
*
          SHOWARAY YES,PATH_BEING_OFFLINED
*
          B BIT Ø4TRY
*
BIT Ø8NO   DS ØH
*
          SHOWARAY NO,PATH_BEING_OFFLINED
*
BIT Ø4TRY DS ØH
          TM Ø(R2),X'Ø4' * HAS RECOVERY FAILED ON THIS CHANNEL?
          BC 8,BIT Ø4NO   * NO SO SET A NO
*
          SHOWARAY YES,PATH_RECOVERY_FAILED
*
          B BIT Ø2TRY
*
BIT Ø4NO   DS ØH
*
          SHOWARAY NO,PATH_RECOVERY_FAILED
*
```

```
BIT02TRY DS 0H
         TM 0(R2),X'02' * HAS RECOVERY NEARLY FINISHED?
         BC 8,BIT02NO   * NO SO SET A NO
         SHOWARAY YES,PATH_RECOVERY_FINISHING
         B ENDBITS
*
BIT02NO  DS 0H
         SHOWARAY NO,PATH_RECOVERY_FINISHING
*
ENDBITS  DS 0H
         LA R3,1(,R3) * INCREMENT R3 BY 1
         C  R3,=F'256' * HAVE ALL PATHS BEEN DONE?
         BNE LOOPER    * NO SO GET THE NEXT SET OF INFO.
*
ENDREXX  DS 0H
*
         STORAGE RELEASE,LENGTH=GETLEN,ADDR=(8)
         PR
         LTORG
YES      DC C'Y'
NO       DC C'N'
*
*********************************************************************
***      IRXEXCOM PARAMETER AREA                                  ***
*********************************************************************
COMSDS   DSECT
COMS     DS    5AL4
COMID    DS    CL8             * IRXEXCOM ID - C'IRXEXCOM'
COMDUMMY DS    AL4             * NOT USED
COMSHVB  DS    (SHVBLEN)X      * IRXEXCOM SHVBLOCK (LENGTH FROM DSECT)
COMRET   DS    AL4             * IRXECOM RC
COMSLEN  EQU *-COMS
PATHPID  DS H
DESCRIBE DS CL32
GETLEN   EQU *-COMS
         DS 0D
         CVT DSECT=YES
         IHAPSA
         IRXEFPL
         IRXARGTB
         IRXEVALB
         IRXENVB
         IRXEXTE
         IRXSHVB
         END
/*
//LKED.SYSLMOD DD DSN=your.steplib,DISP=SHR,UNIT=
//LKED.SYSIN DD *
         ENTRY PATHLIST
         NAME  PATHLIST(R)
```

Catalog maintenance utilities

INTRODUCTION

The following two programs, CATCHECK and VOLCHECK, should prove useful for sites in maintaining catalog entries for IPL volumes (and other non-SMS-managed volumes).

CATCHECK uses the CSI (Catalog Search Interface) to retrieve a list of catalog entries for a particular dataset filter (the default is SYS1.**) and then checks to see if the datasets are in the VTOC of the indicated volume. In addition it will also flag any datasets that are not indirectly catalogued.

VOLCHECK reads the VTOC of the specified volume and then does a locate to check that the dataset is catalogued. The program will also flag datasets where the catalog entry points to a different volume. This program can be useful in determining what catalog changes are needed to implement a new IPL volume (for example a new release of OS/390).

If the DDname SYSPUNCH is present in the JCL of the job then the programs will write IDCAMS 'define nonvsam' and 'delete noscratch' cards for the appropriate conditions.

Readers should review the code because it assumes that your IPL volume is IPL***.

The Catalog Search Interface is documented in the SMS Manual *Managing Catalogs* with several code examples in SYS1.SAMPLIB.

CATCHECK

```
CATCHECK CSECT
CATCHECK AMODE 31
CATCHECK RMODE 24
         USING CATCHECK,R15
         B     Start
         DC    CL8'CATCHECK'
         DC    CL8'&SYSDATE'
Start    DS    ØH
         BAKR  R14,Ø
         LR    R12,R15
```

```
         USING CATCHECK,R12
         DROP  R15

         LA    13,Save_area

         L     R1,Ø(,R1)
         SR    R2,R2
         ICM   R2,B'ØØ11',Ø(R1)            length of parm
         BZ    No_parm
         CHI   R2,8
         BH    Exit
         MVC   Dsn_filter(8),Blanks
         BCTR  R2,Ø
         EX    R2,MVCØØ1

No_parm  equ *
         L     R1,CVTPTR                   r1 -> cvt
         L     R2,CVTSMCA-CVT(,R1)         r2 -> smca
         USING SMCABASE,R2
         MVC   SMF_id(4),SMCASID
         DROP  R2

         L     R2,CVTLINK-CVT(,R1)         r2 -> dcb for sys1.linklib
         L     R2,DCBDEBAD-IHADCB(,R2)     r2 -> deb
         ICM   R2,B'Ø111',DEBSUCBB-DEBBASIC(R2) r2 -> ucb (3 bytes !)
         USING UCBOB,R2
         MVC   IPL_vol(6),UCBVOLI
         DROP  R2

         OPEN  (SYSPRINT,(OUTPUT))

         LA    R2,SYSPRINT
         TM    DCBOFLGS-IHADCB(R2),DCBOFOPN open ok ?
         BZ    Exit

         L     R2,PSATOLD-PSA              r2 -> tcb
         L     R2,TCBTIO-TCB(,R2)          r2 -> tiot
         SR    R9,R9
         SR    R1Ø,R1Ø
Scan_TIOT_loop EQU *
         IC    R9,TIOELNGH-TIOT1(R1Ø,R2)   length of dd entry
         LTR   R9,R9                       end ?
         BZ    Skip_open_for_SYSPUNCH
         LA    R6,TIOEDDNM-TIOT1(R1Ø,R2)   r6 -> ddname
         CLC   Ø(8,R6),=C'SYSPUNCH'        SYSPUNCH dd in jcl ?
         BE    Open_for_SYSPUNCH           yes - go use it
         AR    R1Ø,R9
         B     Scan_TIOT_loop              check next one
Open_for_SYSPUNCH equ *
         OPEN  (SYSPUNCH,(OUTPUT))
         LA    R2,SYSPUNCH
```

```
         TM    DCBOFLGS-IHADCB(R2),DCBOFOPN open ok ?
         BZ    Exit
         MVI   SYSPUNCH_flag,C'Y'

Skip_open_for_SYSPUNCH equ *
         TIME  DEC,Timedate,ZONE=LT,LINKAGE=SYSTEM,DATETYPE=DDMMYYYY
         PUT   SYSPRINT
         MVI   Ø(R1),C' '
         MVC   1(12Ø,R1),Ø(R1)
         MVC   1(4,R1),SMF_id
         MVC   1Ø(1Ø,R1),=C'IPL volume'
         MVC   21(6,R1),IPL_vol
         MVC   3Ø(6,R1),=C'Filter'
         MVC   37(8,R1),Dsn_filter
         MVC   6Ø(1Ø,R1),=X'2Ø2Ø612Ø2Ø612Ø2Ø2Ø2Ø'
         ED    6Ø(1Ø,R1),TIMEDATE+8

         PUT   SYSPRINT
         MVI   Ø(R1),C' '
         MVC   1(12Ø,R1),Ø(R1)

         MVC   CSIFILTK(44),Blanks         set up parms for CSI
         MVC   CSIFILTK(8),Dsn_filter
         MVC   CSICATNM(44),Blanks
         MVC   CSIRESNM(44),Blanks
         MVC   CSIDTYPS(16),Blanks
         MVI   CSICLDI,C' '
         MVI   CSIRESUM,C' '
         MVI   CSIS1CAT,C' '
         LH    R2,=H'1'
         STH   R2,CSINUMEN
         MVC   CSIFLDNM(8),=C'VOLSER  '

         STORAGE OBTAIN,LOC=ANY,COND=YES,LENGTH=(1Ø24ØØ,8192)
         LTR   R15,R15
         BNZ   Abend
         ST    RØ,Ø(,R1)                   save length
         ST    R1,CSI_parmlist+8           save storage addr

         LA    R1,CSI_parmlist
         CALL  IGGCSIØØ                    catalog search interface
         LTR   R15,R15
         BNZ   Abend

         L     R2,CSI_parmlist+8           addr of work area
         USING Work_area_info,R2
         LR    R3,R2
         A     R3,CSIUSDLN                 used length
         CLI   CSICFLG,X'ØØ'
         BNE   ABEND

         PUT   SYSPRINT
```

```
         MVI   Ø(R1),C' '
         MVC   1(12Ø,R1),Ø(R1)
         MVC   1(44,R1),CSICNAME          catalog name

         LA    R2,CSICRETN+4              r2 -> first entry
         DROP  R2
         USING Entry_info,R2

Process_next equ *
         CLI   CSIEFLAG,X'ØØ'             another catalog ?
         BE    Finished
         TM    CSIEFLAG,CSIENTER          error indicator set ?
         BO    Abend

         LH    R4,CSITOTLN                length

         CLC   =C'SYS1.VVDS',CSIENAME
         BE    Skip_vvds

         PUT   SYSPRINT
         LR    R9,R1
         MVI   Ø(R9),C' '
         MVC   1(12Ø,R9),Ø(R9)

         MVC   16(44,R9),CSIENAME         entry name
         MVC   62(1,R9),CSIETYPE          entry type
         C     R4,=F'12'
         BL    No_volser
         MVC   65(6,R9),CSIFDDAT          volser

         CLI   CSIETYPE,C'A'              nonvsam ?
         BNE   Continue_processing

         MVC   DSN(44),CSIENAME
         MVC   VOL(6),IPL_vol
         CLC   CSIFDDAT(6),=C'******'     indirect ?
         BE    Obtain_DSCB
         MVC   VOL(6),CSIFDDAT
         CLC   CSIFDDAT(3),=C'IPL'
         BNE   Obtain_DSCB
         MVC   1(L'Not_indirect,R9),Not_indirect

Obtain_DSCB equ *
         OBTAIN DSCB                      retrieve dscb
         LTR   R15,R15
         BZ    Continue_processing

         CH    R15,=H'8'
         BE    Dataset_not_found

         CH    R15,=H'4'
         BNE   Abend
```

```
         MVC   1(L'Not_mounted,R9),Not_mounted
         B     Continue_processing

Dataset_not_found equ *
         MVC   1(L'Not_found,R9),Not_found
         CLI   SYSPUNCH_flag,C'Y'
         BNE   Continue_processing
         PUT   SYSPUNCH
         MVI   Ø(R1),C' '
         MVC   1(79,R1),Ø(R1)
         MVC   2(6,R1),=C'DELETE'
         MVC   1Ø(44,R1),DSN
         MVC   6Ø(9,R1),=C'NOSCRATCH'

Continue_processing equ *
No_volser           equ *
Skip_vvds           equ *
         LA    R2,CSITOTLN
         AR    R2,R4                      r2 -> next entry

         CR    R2,R3
         BL    Process_next

         CLI   CSIRESUM,C'Y'              resume flag ?
         BNE   Finished

         LA    1,CSI_parmlist
         CALL  IGGCSIØØ                   catalog search interface
         LTR   R15,R15
         BNZ   Abend

         L     R2,CSI_parmlist+8          addr of work area
         USING Work_area_info,R2
         LR    R3,R2
         A     R3,CSIUSDLN                used length
         CLI   CSICFLG,X'ØØ'
         BNE   Abend

         LA    R2,CSICRETN+4              r2 -> first entry
         DROP  R2
         B     Process_next

Finished equ *
         L     R1,CSI_parmlist+8          storage addr
         L     RØ,Ø(,R1)                  length
         STORAGE RELEASE,LENGTH=(Ø),ADDR=(1)

         CLOSE (SYSPRINT)

         CLI   SYSPUNCH_flag,C'Y'
```

```
         BNE   Exit
         CLOSE (SYSPUNCH)

Exit     PR                                  return to caller

Abend    ABEND 99,DUMP

MVC001   MVC   Dsn_filter(0),2(R1)

DSCB     CAMLST    SEARCH,DSN,VOL,WORK
DSN         DC     CL44' '           dataset name
VOL         DC     CL6' '            volume serial
WORK        DS     140C              140-byte work area

*************************************************************************
* PARAMETER LIST FOR IGGCSI00 INVOCATION                                *
*************************************************************************
CSI_parmlist   DS 0D
         DC    A(MODRSNRT)               MODULE/REASON/RETURN
         DC    A(CSIFIELD)
         DC    A(0)
*************************************************************************
* MODULE ID/REASON CODE/RETURN CODE                                     *
*************************************************************************
MODRSNRT    DS 0F
PARMRC      DS 0CL4
MODID       DC XL2'0000'     MODULE ID
RSNCODE     DC XL1'00'       REASON CODE
RTNCODE     DC XL1'00'       RETURN CODE
*************************************************************************
* PARAMETER FIELDS FOR CATALOG SEARCH INTERFACE (CSI)                   *
*************************************************************************
CSIFIELD    DS 0C
CSIFILTK    DS CL44           FILTER   KEY
CSICATNM    DS CL44           CATALOG NAME OR BLANKS
CSIRESNM    DS CL44           RESUME NAME OR BLANKS
CSIDTYPD    DS 0CL16          ENTRY TYPES
CSIDTYPS    DS 16CL1          ENTRY TYPES
CSIOPTS     DS 0CL4           CSI OPTIONS
CSICLDI     DS CL1            RETURN D&I IF C A MATCH Y OR BLNK
CSIRESUM    DS CL1            RESUME FLAG         Y OR BLANK
CSIS1CAT    DS CL1            SEARCH CATALOG      Y OR BLANK
CSIRESRV    DS XL1            RESERVED
CSINUMEN    DS H              NUMBER OF ENTRIES FOLLOWING
CSIENTS     DS 0CL8           VARIABLE NUMBER OF ENTRIES FOLLOW
CSIFLDNM    DS CL8            FIELD NAME
Save_area      DC    18F'0'
Blanks         DC    CL100' '
Dsn_filter     DC    CL8'SYS1.**'
SMF_id         DC    CL4' '
IPL_vol        DC    CL6' '
```

```
SYSPUNCH_flag  DC    C'N'
Not_found      DC    C'* not found   '
Not_mounted    DC    C'* not mounted '
Not_indirect   DC    C'* not indirect'
Timedate       DS    CL16

SYSPRINT DCB    DDNAME=SYSPRINT,DSORG=PS,MACRF=PL,LRECL=121,BLKSIZE=Ø
SYSPUNCH DCB    DDNAME=SYSPUNCH,DSORG=PS,MACRF=PL,LRECL=8Ø,BLKSIZE=Ø

         LTORG

Work_area_info DSECT
CSIUSRLN DS    F
CSIREQLN DS    F
CSIUSDLN DS    F
CSINUMFD DS    H
*
CSICFLG  DS    CL1
CSICTYPE DS    CL1
CSICNAME DS    CL44
CSICRETN DS    ØCL4
CSICRETM DS    CL2
CSICRETR DS    CL1
CSICRETC DS    CL1

Entry_info     DSECT
CSIEFLAG DS    XL1
CSIENTER EQU   B'Ø1ØØØØØØ'
CSIETYPE DS    XL1
CSIENAME DS    CL44
CSITOTLN DS    XL2
         DS    XL2
CSILENF1 DS    XL2
CSIFDDAT DS    XL1
         @REGS
         CVT   DSECT=YES
         IEESMCA
         IHAPSA
         IKJTCB
         IEFTIOT1
         IEZDEB
         IEFUCBOB
         DCBD   DSORG=PS,DEVD=DA
         END
```

VOLCHECK

```
VOLCHECK CSECT
         USING VOLCHECK,R15
         B     Start
```

```
         DC    CL8'VOLCHECK'
         DC    CL8'&SYSDATE'

Start    DS    ØH
         BAKR  R14,Ø
         LR    R12,R15
         USING VOLCHECK,R12
         DROP  R15
         LA    R13,Save_area
         OPEN  (SYSPRINT,(OUTPUT))
         LA    R1Ø,SYSPRINT
         TM    DCBOFLGS-IHADCB(R1Ø),DCBOFOPN    open ok ?
         BZ    EXIT
         RDJFCB VTOC
         LTR   R15,R15
         BNZ   Exit
         LA    R1Ø,VTOC_JFCB
         MVC   JFCBDSNM-INFMJFCB(44,R1Ø),VTOC_name
         OPEN  VTOC,TYPE=J
         MVC   Volume(6),JFCBVOLS-INFMJFCB(R1Ø)
         CLC   =C'IPL',JFCBVOLS-INFMJFCB(R1Ø)
         BNE   Skip_open_for_SYSPUNCH
         L     R2,PSATOLD-PSA              r2 -> tcb
         L     R2,TCBTIO-TCB(,R2)          r2 -> tiot
         SR    R9,R9
         SR    R1Ø,R1Ø

Scan_TIOT_loop EQU *
         IC    R9,TIOELNGH-TIOT1(R1Ø,R2)   length of dd entry
         LTR   R9,R9                       end ?
         BZ    Skip_open_for_SYSPUNCH
         LA    R6,TIOEDDNM-TIOT1(R1Ø,R2)   r6 -> ddname
         CLC   Ø(8,R6),=C'SYSPUNCH'        SYSPUNCH dd in jcl ?
         BE    Open_for_SYSPUNCH           yes - go use it
         AR    R1Ø,R9
         B     Scan_TIOT_loop              check next one

Open_for_SYSPUNCH equ *
         OPEN  (SYSPUNCH,(OUTPUT))
         LA    R2,SYSPUNCH
         TM    DCBOFLGS-IHADCB(R2),DCBOFOPN open ok ?
         BZ    Exit
         MVI   SYSPUNCH_flag,C'Y'

Skip_open_for_SYSPUNCH equ *
         TIME  DEC,Timedate,ZONE=LT,LINKAGE=SYSTEM,DATETYPE=DDMMYYYY
         L     R1,CVTPTR
         L     R1,CVTSMCA-CVT(,R1)
         USING SMCABASE,R1
         MVC   SMF_id(4),SMCASID
         DROP  R1
         BAL   R3,Put_SYSPRINT
```

```
         MVC   1(4,R4),SMF_id
         MVC   10(12,R4),=C'Datasets on '
         MVC   22(6,R4),Volume
         MVC   60(10,R4),=X'2020612020612020202020'
         ED    60(10,R4),TIMEDATE+8
         BAL   R3,Put_SYSPRINT
         BAL   R3,Put_SYSPRINT
         MVC   2(L'Header1,R4),Header1
         MVC   47(L'Header2,R4),Header2
         LA    R11,DSCB

Read_loop equ *
         READ  VTOC_ECB,SF,VTOC,(R11)           read vtoc
         CHECK VTOC_ECB
         CLI   DS1FMTID-IECSDSL1(R11),C'1'      format 1 ?
         BNE   Read_loop                        no - read next
         CLC   =C'SYS1.VTOCIX',DS1DSNAM-IECSDSL1(R11)
         BE    Read_loop                        not interested
         CLC   =C'FDRABR',DS1DSNAM-IECSDSL1(R11)
         BE    Read_loop                        not interested

         BAL   R3,Put_SYSPRINT
         MVC   2(44,R4),DS1DSNAM-IECSDSL1(R11)  dataset name
         MVC   DSN(44),DS1DSNAM-IECSDSL1(R11)
         LOCATE BY_NAME                         search catalog
         LTR   R15,R15
         BNZ   Not_cataloged
         MVC   47(6,R4),INFO+6                  volser in catalog
         CLC   INFO+6(6),Volume                 right volume ?
         BE    Read_loop                        yes - ok
         MVC   60(11,R4),=C'** mismatch'        no - flag it
         B     Read_loop

Not_cataloged equ *
         MVC   60(16,R4),=C'** not cataloged'
         CLI   SYSPUNCH_flag,C'Y'
         BNE   Read_loop
         MVC   Define_dsn(44),DS1DSNAM-IECSDSL1(R11)
         PUT   SYSPUNCH
         MVC   0(80,R1),Define
         PUT   SYSPUNCH
         MVC   0(80,R1),Define_cont
         B     Read_loop

VTOC_eof CLOSE (SYSPRINT,,VTOC)
         CLI   SYSPUNCH_flag,C'Y'
         BNE   Exit
         CLOSE (SYSPUNCH)
Exit     PR                                     return to mvs
Put_SYSPRINT equ *
         PUT   SYSPRINT
         MVI   0(R1),C' '
```

```
         MVC   1(120,R1),0(R1)
         LR    R4,R1
         BR    R3
Save_area      DC 18F'0'
Volume         DC CL6' '
SMF_id         DC CL4' '
SYSPUNCH_flag  DC C'N'
Timedate       DS CL16
Header1        DC C'——— dataset name ———'
Header2        DC C'catalog entry'
DSCB           DC CL140' '
VTOC_name      DC 44X'04'
               DS 0F
VTOC_exit_list DC X'87'
               DC AL3(VTOC_JFCB)
               DS 0F
VTOC_JFCB      DC XL176'00'
Define         DC     0CL80
               DC     C' DEF NVSAM(NAME('
Define_dsn     DC     CL44' '
               DC     C' ) - '
               DC     CL(80-*+Define)' '
Define_cont    DC     CL80'   DEVT(0000) VOL(******))'
               LTORG

BY_NAME  CAMLST NAME,DSN,,INFO
DSN            DC CL44' '
INFO           DS 0D
               DS 265C

SYSPRINT DCB   DDNAME=SYSPRINT,DSORG=PS,MACRF=PL,LRECL=121,BLKSIZE=0
SYSPUNCH DCB   DDNAME=SYSPUNCH,DSORG=PS,MACRF=PL,LRECL=80,BLKSIZE=0

VTOC     DCB   DDNAME=VTOC,DSORG=PS,MACRF=R,LRECL=96,BLKSIZE=96,       X
               RECFM=F,KEYLEN=44,EXLST=VTOC_exit_list,EODAD=VTOC_eof
         DCBD  DSORG=PS,DEVD=DA
         CVT   DSECT=YES
         IEESMCA
         IHAPSA
         IKJTCB
         IEFTIOT1
         DSECT
         IEFJFCBN
         DSECT
         IECSDSL1 (1)
         @REGS
         END
```

Invoking MVS commands

INTRODUCTION

MVSCMD issues an MVS command and displays the response. It can be used on-line or in batch. When used on-line in TSO/ISPF, it writes the MVS response messages into a dataset, then invokes ISPF BROWSE. Here is a sample output, sending command 'D TS,L' to all systems in a sysplex:

```
IEE421I RO *ALL,D TS,L Ø71
MVSA     RESPONSES ————————————————————————-
IEE114I 17.32.12 2ØØØ.124 ACTIVITY 963
 JOBS     M/S    TS USERS    SYSAS    INITS   ACTIVE/MAX VTAM     OAS
ØØØ19    ØØØ78    ØØØØ5      ØØØ26    ØØØ24    ØØØØ9/ØØØ2Ø       ØØØ15
VOTT    OWT      OPER5   OWT      BV11572 OWT      *LOGON* OWT
VDOERN  OWT
MVSB     RESPONSES ————————————————————————-
IEE114I 17.32.12 2ØØØ.124 ACTIVITY Ø68
 JOBS     M/S    TS USERS    SYSAS    INITS   ACTIVE/MAX VTAM     OAS
ØØØ18    ØØ124    ØØØØ8      ØØØ26    ØØØ22    ØØ137/ØØ4ØØ       ØØØ29
OPER9   OWT      VSHOFF  OWT      YWEBER  OWT      VFREIT  OWT
YWEBER  OWT      XV88483 OWT      XV88666 OWT      XVØ578Ø OWT
MVSC     RESPONSES ————————————————————————-
IEE114I 17.32.12 2ØØØ.124 ACTIVITY 158
 JOBS     M/S    TS USERS    SYSAS    INITS   ACTIVE/MAX VTAM     OAS
ØØØ11    ØØ1Ø4    ØØØØ7      ØØØ26    ØØØ15    ØØØØ5/ØØØ2Ø       ØØØ16
OPER8   OWT      VFREIT  OWT      SGOLKE  OWT      SIVENA  OWT
XV88Ø15 OWT      XV12443 OWT      VOTTKU  OWT
MVSD     RESPONSES ————————————————————————-
IEE114I 17.32.12 2ØØØ.124 ACTIVITY 358
 JOBS     M/S    TS USERS    SYSAS    INITS   ACTIVE/MAX VTAM     OAS
ØØØØ8    ØØØ99    ØØØ2Ø      ØØØ26    ØØØ12    ØØØ75/ØØ26Ø       ØØØ13
FBPA    OWT      FTSCHJ  OWT      FTKUHL  OWT      FTRAUH  OWT
YMUENN  OWT      SIVENS  OWT      FTLOTZ  OWT      XV14227 OWT
R213730 OWT      FUSCWD  OWT      FTANRE  OWT      FABBOP  OWT
FTBHAN  OWT      FTJORT  OWT      FTSMEY  OWT      YROTHU  OWT
FTSCHN  OWT      *LOGON* OWT      FUFORS  OWT      FTDRES  OWT
```

It needs parameters for system-id and command-text, like the following:

- TSO MVSCMD MVSA D R,L – display PENDING EQUESTS on system MVSA
- TSO MVSCMD ALL D TS,L – display TSO users on all systems in a sysplex.

But it is easier to use when defined as an ISPF command like the following:

```
Command . .: MVSA
Trunc . . .: Ø
Action  . .: SELECT CMD(%MVSCMD MVSA &ZPARM) NEWAPPL(ISR)
```

Description: invoke MVS command on MVSA.

Then you could enter 'MVSA D IPLINFO' from any ISPF command line to see IPL information about system MVSA.

Note: your TSO user-id needs CONSOLE authority to use this command.

MVSCMD can also be used in a batch job as a normal batch TSO step with no ISPF datasets needed. Then the response messages go to SYSPRINT, which is usually SYSOUT but could also be a sequential file (for input to another step for example). Some sample JCL is shown below:

```
//*
//BATCHCMD EXEC PGM=IKJEFTØ1,DYNAMNBR=2Ø
//SYSEXEC  DD DISP=SHR,DSN=UTILITY.EXEC  <- MVSCMD in this library
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  MVSCMD MVSA V 1612,ONLINE
  MVSCMD MVSA m 1612,vol=(sl,volØØ1),use=private
  MVSCMD mvsa d u,,,1612,1
//*
```

MVSCMD EXEC

```
/*============================== REXX ================================*/
/*  MVSCMD - ISSUE MVS COMMAND AND BROWSE THE RESPONSE               */
/*                                                                   */
/*        It is invoked with 2 parameters (system-id & MVS-command)  */
/*          TSO %MVSCMD sysid command                                */
/*                                                                   */
/*        It can be usefully defined as ISPF commands:               */
/*          MVSCMD: 'SELECT CMD(%MVSCMD &ZPARM)'                     */
/*          MVSALL: 'SELECT CMD(%MVSCMD ALL &ZPARM)'                 */
/*          sysid : 'SELECT CMD(%MVSCMD sysid &ZPARM)'               */
/*                                                                   */
/*        The MVS console output is put into a dataset called        */
/*        'userid.MVSCMD.sysid' & browsed using ISPF BROWSE.         */
/*                                                                   */
/*        In TSO batch the response goes to SYSPRINT.                */
```

```
/*                                                                    */
/*        JES3 command response goes directly to the screen in        */
/*        line mode.  This is forced because GETMSG cannot handle     */
/*        multi-line messages from JES3 (but it's OK with JES2).      */
/*        In TSO batch the JES3 response can be seen only in the      */
/*        job's messages or in SYSLOG.                                */
/*                                                                    */
/*        The user requires CONSOLE authority to use this EXEC.       */
/*                                                                    */
/*--------------------------------------------------------------------*/
/* Version:  1.4                          Last updated: 2000/05/03    */
/*====================================================================*/
   Arg system cmd
   If system = '' ! cmd = '' Then Do
      Say '*** MVSCMD needs 2 parameters: System-Id and MVS-command'
      Say "***                    syntax: 'MVSCMD sysid command'"
      Exit 8
      End

   If Left(cmd,1) = '*' Then jes3 = 'YES'             /* JES3 command */
   If SYSVAR('SYSENV') = 'BACK' Then batch = 'YES'    /* TSO batch    */
   If SYSVAR('SYSISPF') = 'ACTIVE' Then ispf = 'YES'  /* ISPF active  */

   If (batch <> 'YES' & ispf <> 'YES') ! jes3 = 'YES'
      Then soldisp = 'YES'                    /* line-mode display    */
      Else soldisp = 'NO'

   If (batch <> 'YES' & ispf = 'YES') Then    /* on-line TSO/ISPF     */
      Call INITVARS                           /* initialize variables */

   If system <> MVSVAR('SYSNAME'),            /* different system     */
    & jes3 <> 'YES' Then Do                   /* not a JES3 command   */
      If system = 'ALL'
         Then cmd = 'ROUTE *ALL,'cmd
         Else cmd = 'ROUTE' system','cmd
      End

   /*-----------------------*/
   /* console environment */
   /*-----------------------*/
   "CONSPROF SOLDISPLAY("soldisp") UNSOLDISPLAY(NO)",
            "SOLNUM(9999) UNSOLNUM(0)"
   If rc <> 0 Then Do
      Say '*** Userid' USERID() 'needs CONSOLE authority to use MVSCMD'
      Exit 8
      End

   "CONSOLE ACTIVATE"

   cartval = USERID()!!TIME()             /* create unique CART value */
```

```
   "CONSOLE SYSCMD("cmd") CART("cartval")"  /* issue the command      */

   If batch = 'YES'               /* set maximum wait time for GETMSG */
      Then wait_time = 30
      Else wait_time = 5
   get_rc = GETMSG('resp.','SOL',cartval,,wait_time)  /* get response */

   "CONSOLE DEACTIVATE"                   /* finished with console    */

   /*-------------------------------------------------------------------*/
   /* copy the command response to appropriate output destination */
   /*-------------------------------------------------------------------*/
   If soldisp = 'NO' Then Do
      If get_rc = 0 Then Do     /* GETMSG was OK */
         If batch = 'YES' Then
            "EXECIO * DISKW SYSPRINT (STEM resp. "
         Else Do                /* write header & messages in dataset */
            Call ALLOCDS                      /* allocate dataset     */
            "EXECIO * DISKW MVSCMD (STEM hdr. OPEN"
            "EXECIO * DISKW MVSCMD (STEM blnks."
            "EXECIO * DISKW MVSCMD (STEM resp. "
            "EXECIO * DISKW MVSCMD (STEM blnks. FINIS "
            "FREE F(MVSCMD)"

            Address ISPEXEC "ISPEXEC BROWSE DATASET("outds")"

            pmsg = MSG('OFF')
            "DELETE" outds   /* delete it, or else HSM may migrate it */
            x = MSG(pmsg)

            End
         End
      Else
         Say "GETMSG error retrieving message.  RC =" get_rc
      End

   If (batch = 'YES' & jes3 = 'YES') Then
      Say "JES3 response messages can be seen only in this job's",
          "messages or in SYSLOG"

   Exit get_rc

/*====================================================================*/
/* INITIALIZE VARIABLES  (for ISPF on-line)                           */
/*--------------------------------------------------------------------*/
INITVARS:
   Address ISPEXEC "VGET (ZPREFIX ZUSER)"
   If ZPREFIX <> ZUSER             /* prefix for output dataset name */
      Then tso_prefix = ZPREFIX'.'ZUSER
      Else tso_prefix = ZUSER
```

```
   /*--------------------*/
   /* variables for writing to dataset */
   /*--------------------*/
   blnks.0 = 2
   blnks.1 = ' '
   blnks.2 = ' '
   hdr.0 = 3
   hdr.1 = ' '
   hdr.2 = ' System:' system COPIES(' ',44) TIME()',' DATE()
   hdr.3 = 'Command:' cmd
   Return

/*====================================================================*/
/* ALLOCATE DATASET FOR CONSOLE OUTPUT (to be browsed)                */
/*------------------------------------*/
ALLOCDS:
   outds = "'"tso_prefix".MVSCMD."system"'"    /* output dataset name */

   If SYSDSN(outds) = "OK"
      Then alloc_info = "SHR REUSE"
      Else alloc_info = "NEW UNIT(3390) SPACE(1 1) TRACKS RECFM(V B)",
                        "LRECL(125) DSORG(PS) REUSE"

   "ALLOC F(MVSCMD) DATASET("outds")" alloc_info
   If rc <> 0 Then Do
      Say '*** ERROR: unable to allocate dataset' outds
      Say '***        as' alloc_info
      Say '***'
      Say '***        The command response will be in SYSLOG.'
      Exit rc
      End
   Return
```

CONCLUSION

MVSCMD provides a simple method of issuing commands, which you can use from (almost) any ISPF panel. This becomes increasingly useful as the number of interesting MVS DISPLAY commands grows to get various MVS system information very quickly and easily. It can also be useful in batch jobs.

Ron Brown
Systems Programmer (Germany)

Searching with COBOL

INTRODUCTION

This facility has been around for quite a while. We just forget it is there because it does not get much use.

THE SLOW WAY

Most of the time, if we want to find something in an internal table, we do a serial search against it, bumping up a subscript or index until we find a match or run out of table space. If the table is large, this can eat up quite a bit of CPU time (although modern CPU horsepower makes this less of a problem than it used to be).

A FASTER WAY

The SEARCH verb will usually find a match faster. There are two methods – serial searches and binary searches.

SERIAL SEARCH

The code needed is:

```
DATA DIVISION.
Ø5 EXAMPLE-TBL     OCCURS 4ØØ TIMES
                    INDEXED BY E-INDX.
    1Ø EXT-KEY           PIC X(5).
    1Ø EXT-DATA          PIC X(2Ø).

PROCEDURE DIVISION.
```

We will assume that the program exercises some code to load the table.

```
SET E-INDX TO 1.
SEARCH EXAMPLE-TBL VARYING E-INDX
  AT END
    PERFORM 9ØØ-TBL-ERROR
  WHEN EXT-KEY (E-INDX) = SEARCH-ITEM
    PERFORM 2ØØ-KEY-FOUND
  WHEN EXT-DATA (E-INDX) = SPACES
    PERFORM 9ØØ-TBL-ERROR
END-SEARCH.
```

The code includes the following:

- The INDEXED BY statement is required since the SEARCH verb uses an index. You can specify more than one index if needed.
- The index used for the search must be initialized. It does not have to start at the beginning of the table. The search will start wherever the index points and proceed to the end of the table.
- The VARYING statement is optional. If omitted, it will use the first (or only) index specified for the table. We could have omitted VARYING for this search.
- AT END is optional. The default is to proceed to the next statement after the SEARCH statement if a match is not found.

BINARY SEARCH

Firstly the table has to be in sequence by the key you want to search on, so set it up or load it that way. It can be in ascending or descending sequence, eg:

```
DATA DIVISION.
Ø5 EXAMPLE-TBL      OCCURS 4ØØ TIMES
                    ASCENDING EXT-KEY
                    INDEXED BY E-INDX
                               E-INDX2.
  1Ø EXT-KEY           PIC X(5).
  1Ø EXT-DATA          PIC X(2Ø).

PROCEDURE DIVISION.
```

We will assume that the program exercises some code to load the table.

```
SEARCH ALL EXAMPLE-TBL
  AT END
    PERFORM 9ØØ-TBL-ERROR
  WHEN EXT-KEY (E-INDX) = SEARCH-ITEM
   AND EXT-DATA (E-INDX) NOT = SPACES
    PERFORM 2ØØ-KEY-FOUND
END-SEARCH.
```

The code includes the following:

- Note that the index was not initialized. The entire table will be searched using binary search techniques.

- AT END is optional. The default is to proceed to the next statement after the SEARCH statement if a match is not found.
- The first (or only) index specified for the table is ALWAYS used.

Search has the ability to handle multi-level tables up to seven deep, with multiple key fields. Here is a fairly simple example that uses a two-dimensional table:

```
DATA DIVISION.
Ø5 EXAMPLE-TBL      OCCURS 4ØØ TIMES
                    ASCENDING EXT-KEY
                    INDEXED BY E-INDX
                               E-INDXB.
  1Ø EXT-KEY           PIC X(5).
  1Ø EXAMPLE-SECONDARY  OCCURS 2Ø TIMES
                         DESCENDING EXT-KEY2
                         INDEXED BY E-INDX2.
    15 EXT-KEY2        PIC X(3).
    1Ø EXT-DATA        PIC X(2Ø).

PROCEDURE DIVISION
```

We will assume that the program exercises some code to load the table.

```
SEARCH ALL EXAMPLE-TBL
  AT END
    PERFORM 9ØØ-TBL-ERROR
  WHEN EXT-KEY (E-INDX) = SEARCH-ITEM1
   AND EXT-KEY2 (E-INDX, E-INDX2) = SEARCH-ITEM2
   AND EXT-DATA (E-INDX, E-INDX2) NOT = SPACES
    PERFORM 2ØØ-KEY-FOUND
END-SEARCH.
```

Alan Kalar
Systems Programmer (USA)

If you want to contribute an article to *MVS Update*, a copy of our *Notes for contributors* can be downloaded from the Xephon Web site. The URL is: www.xephon.com/contnote.html.

A Java client/server application on OS/390

INTRODUCTION

The use of the Java language is not only restricted to applets programming. Java can be very useful to implement user-friendly interfaces to display on a PC workstation data collected from a server on OS/390. This article describes the use of Java to implement a simple client/server application on OS/390.

This application, whose real goal is only to be used as an example to detail Java programming concepts, will allow a Java client on a PC workstation to interact with a Java server located on OS/390.

This example shows how to implement a TCP/IP communication in Java. It also describes how to use the Java Native Interface (JNI) to allow Java programs on OS/390 to communicate with C/C++ and Assembler routines.

This application is a case study and implements only two actions which demonstrate how to manage input/output flows between the client and the server. These two actions are:

- Collect and display information about the last IPL of the OS/390 system.
- Send an MVS command for execution to the OS/390 system.

APPLICATION ARCHITECTURE

Both the client and the server are written in Java and communicate through TCP/IP sockets.

- Server – the server runs on OS/390 and calls Assembler routines to execute elementary actions.

  Java programs cannot directly communicate with an Assembler routine. The only way to communicate with an Assembler program (or a COBOL program) from Java is to use the Java Native Interface (JNI), which is a C/C++ interface to Java.

- Client – the client will run typically on a PC workstation. It uses a graphical interface to communicate with the end-user.

JAVA PROGRAMMING

First, you have to remember that Unix System Services (OpenEdition) are required in order to run Java application on OS/390. Detailed information about Java on OS/390 can be found at URL: http://www.s390.ibm.com/java/

The JDK level which was available when I wrote this article was JDK 1.1.8. I used the build of 8 January 2000. You can check out the exact version of JDK you are running by typing the following command:

```
I990557:/: >java -fullversion
java full version «JDK 1.1.8 IBM build m118-20000108 RØ6 BFP (JIT enabled:
jitc
V3.Ø-20000108)»
I990557:/: >
```

It is highly recommended to always get the latest version of the JDK, because the product is improved in functionality and performance continually.

Because Java programming concepts are not familiar to OS/390 systems programmers, I will first try to explain, using very simple samples, the main functions used in my application:

- The Java Native Interface (JNI).
- The TCP/IP communication.
- The interface between C/C++ and Assembler programs.

Java Native Interface (JNI)

The Java Native Interface defines a C/C++ interface to Java. Other programming languages cannot communicate directly with Java. Thus, to integrate a COBOL or an Assembler module in Java you have to write a piece of C/C++ code, that in turn performs the link to the COBOL or the Assembler routine.

JNI programming is a huge topic. In this article, I will show only the basic concepts and some of the OS/390 specifics of JNI. These include:

- How to call a C/C++ routine from Java.
- How to pass data fields between Java and C/C++.

For a more detailed introduction to JNI in general refer to the Sun JNI Web site: http://java.sun.com/products/jdk/1.1/docs/guide/jni/.

A detailed description of the JNI specifics for OS/390 can be found at: http://www.ibm.com/s390/java/jni_oe.html.

Calling a C/C++ routine from Java

In order to describe the basic step to use JNI, I will use very simple Java and C/C++ programs.

Step 1 : write the Java program

First, we should write a Java class named HelloWorld:

```
import java.io.*;

public class HelloWorld
  {
    public native void displayHello();
      static
      {
       System.loadLibrary("HelloJNI");       /* call C/C++ routine */
      }
    public static void main(String[] args)
      {
       System.out.println("From HelloWorld");
       new HelloWorld().displayHello();
      }
  }
```

The System.loadLibrary statement loads the shared library (or DLL) containing our C/C++ routine. For the JVM to find the DLL at run time, the directory where the DLL resides must be part of the LIBPATH variable in your profile. The name of the physical C/C++ module routine which will called by Java will be libHelloJNI.so.

The JVM will automatically complete the library name depending on the run time platform: it will add the lib prefix for a Unix platforms like OpenEdition.

Step 2: compile the Java program with javac

We should use the javac command to compile the previous Java program:

```
I990557:/u/i990557/java/jni/pgm00: >javac HelloWorld.java
I990557:/u/i990557/java/jni/pgm00: >
```

Step 3: create a C-Header file with javah

The C/C++ routine must include a C-header providing a function prototype for the implementation of the displayHello method. This header can be generated using the javah command:

```
I990557:/u/i990557/java/jni/pgm00: >javah -jni HelloWorld
I990557:/u/i990557/java/jni/pgm00: >
```

The javah tool comes with the Java Development Kit (JDK) for OS/390. In our example javah generates a file HelloWorld.h containing a function prototype Java_HelloWorld_displayHello.

Step 4: write the C/C++ program

Then, we should write the following very simple C/C++ routine.

```
#include <stdio.h>
#include <jni.h>
#include "HelloWorld.h"     /* generated by javah */

JNIEXPORT void JNICALL Java_HelloWorld_displayHello
  (JNIEnv *env, jobject obj)
{
  printf("Hello World, this is C, called from Java ...\n");
}
```

Step 5 : compile and link-edit the C/C++ program

At this point, we should use the C/C++ compiler to get a DLL function. This can be done with native c89 commands:

```
c89 -c -W c,expo,dll -DNEEDSIEEE754 -DNEEDSLONGLONG -o HelloWorld.o -I. -I/
usr/l pp/java/J1.1/include -I/usr/lpp/java/J1.1/include/mvs HelloWorld.c


c89 -W l,dll -o libHelloJNI.so HelloWorld.o /usr/lpp/java/J1.1/lib/mvs/
native_th
reads/libjava.x
```

But the easiest way to compile the Java and the C/C++ programs is to use a makefile:

```
MAIN = HelloWorld
CC = c89 -c -W c,expo,dll -DNEEDSIEEE754 -DNEEDSLONGLONG
CFLAGS := -I. -I/usr/lpp/java/J1.1/include -I/usr/lpp/java/J1.1/include/mvs
LL = c89 -W l,dll
LFLAGS := /usr/lpp/java/J1.1/lib/mvs/native_threads/libjava.x

$(MAIN): $(MAIN).o ; $(LL) -o libHelloJNI.so $(MAIN).o $(LFLAGS)
$(MAIN).o: $(MAIN).c $(MAIN).h ; $(CC) -o $(MAIN).o $(CFLAGS) $(MAIN).c

$(MAIN).class: $(MAIN).java ; javac $(MAIN).java
$(MAIN).h: $(MAIN).class ; javah -jni -o $*.h $(MAIN)
```

This utility compiles the Java and the C/C++ programs in a single step:

```
I990557:/u/i990557/java/jni/pgm00: >make
javac HelloWorld.java
javah -jni -o HelloWorld.h HelloWorld
c89 -c -W c,expo,dll -DNEEDSIEEE754 -DNEEDSLONGLONG -o HelloWorld.o -I. -I/
usr/l
pp/java/J1.1/include -I/usr/lpp/java/J1.1/include/mvs HelloWorld.c
c89 -W l,dll -o libHelloJNI.so HelloWorld.o /usr/lpp/java/J1.1/lib/mvs/
native_th
reads/libjava.x
I990557:/u/i990557/java/jni/pgm00: >
```

This makefile will create the shared library libHelloJNI.so, which contains the implementation of the native method displayHello.

Step 6 : execute HelloWorld

The last step is to run the HelloWorld class. You have to make sure that the shared library can be found by the JVM at run time. This can be achieved by making the directory the shared library resides in part of the LIBPATH environment variable. If everything is set up correctly, our JNI example application will display:

```
I990557:/u/i990557/java/jni/pgm00: >java HelloWorld
From HelloWorld
Hello World, this is C, called from Java ...
I990557:/u/i990557/java/jni/pgm00: >
```

Passing data fields between Java and C/C++

In order to describe how to pass data fields between Java and C/C++, we enhance the HelloWorld class of the previous paragraph.

Step 1: write the Java program

We add the String helloString and replace the method displayHello with the native method modifyHello. Now our Java code looks like this:

```
import java.io.*;

public class HelloWorld
  {
    String helloString = "Value Before";

    public native void modifyHello();
      static
      {
       System.loadLibrary("HelloJNI");
      }

    public static void main(String[] args)
      {
       HelloWorld hw = new HelloWorld();
       System.out.println("Before: " + hw.helloString);
       hw.modifyHello();
       System.out.println("After: "+ hw.helloString);
      }
  }
```

Step 2: write the C/C++ program

The C/C++ program implementing JNI looks like:

```
#include <stdio.h>
#include <locale.h>
#include <jni.h>
#include <jni_convert.h>
#include "HelloWorld.h"

JNIEXPORT void JNICALL Java_HelloWorld_modifyHello
  (JNIEnv *env, jobject obj)
{
  int rc;
  jclass jcls;
  char *fieldName;
  char *signature;
  jfieldID field;
  const char *cstring;
  jstring jstr;

  /* Get a reference to the Class object */
  jcls = (*env)->GetObjectClass(env, obj);
```

```
  /* === Manage String === */

  /* convert the name of the field to ascii */
  fieldName = "helloString";
  __etoa(fieldName);

  /* convert the signature to ascii */
  signature = "Ljava/lang/String;";
  __etoa(signature);

  /* obtain the field ID */
  field = (*env)->GetFieldID(env, jcls, fieldName, signature);


  /* create a new jstring */
  cstring = "Hello World, this is C ...";
  rc = NewStringPlatform(env, cstring, &jstr, 0);

  /* modify the String object in Java */
  (*env)->SetObjectField(env, obj, field, jstr);
}
```

I will try to comment step by step the structure of this program. To set the string field displayHello of the class HelloWorld from JNI you would have to follow these steps:

The JNI allows C/C++ routines to access the fields of Java objects. The JNI identifies fields by their symbolic names and type signatures.

In order to directly access elements of a calling Java object from native code, such as fields, methods, or exceptions, we first have to get a pointer (a reference) to the underlying Java class. This is achieved by the following method:

```
/* Get a reference to the Class object */
  jcls = (*env)->GetObjectClass(env, obj);
```

A JNI function has at least two parameters:

- env is a pointer to the JNI interface structure JNIEnv, which is unique for every single Java thread. The JNI interface structure itself holds information about available JNI interface functions.
- obj is a pointer to a structure that represents the calling Java object.

JNI for OS/390 requires the text strings to be converted to ASCII before they can be passed to a JNI function:

```
/* convert the name of the field to ascii */
  fieldName = "helloString";
  __etoa(fieldName);
```

Now, fieldName contains the field name converted to ASCII. The signature string has to be converted in the same manner:

```
/* convert the signature to ascii */
  signature = "Ljava/lang/String;";
  __etoa(signature);
```

- Now that we have converted the name and the signature of countHello we can get a reference to its field ID. This is achieved by the JNI function GetFieldID:

  ```
  /* obtain the field ID */
    field = (*env)->GetFieldID(env, jcls, fieldName, signature);
  ```

- The JNI expects all textual information that is passed to or returned from JNI, such as function parameters, to be in ASCII. On OS/390 this implies that every string parameter to a JNI function has to be converted to ASCII before you can call the function. The JNI implementation on OS/390 provides a few JNI APIs to help with conversion, namely: GetStringPlatform,and GetStringPlatformLength.
- NewStringPlatform

  ```
  /* create a new jstring */
  cstring = "Hello World, this is C ...";
  /* convert to ascii */
  rc = NewStringPlatform(env, cstring, &jstr, Ø);

  /* modify the String object in Java */
  (*env)->SetObjectField(env, obj, field, jstr);
  ```

- We should use the make command to compile in one step the Java and the C/C++ programs:

```
MAIN = HelloWorld
CC = c89 -c -W c,expo,dll -DNEEDSIEEE754 -DNEEDSLONGLONG
CFLAGS := -I. -I/usr/lpp/java/J1.1/include -I/usr/lpp/java/J1.1/include/mvs
LL = c89 -W l,dll
```

```
LFLAG1 := /usr/lpp/java/J1.1/lib/mvs/native_threads/libjava.x
LFLAG2 := /usr/lpp/java/J1.1/lib/mvs/native_threads/libJNIConvert.x

$(MAIN): $(MAIN).o ; $(LL) -o libHelloJNI.so $(MAIN).o $(LFLAG1) $(LFLAG2)
$(MAIN).o: $(MAIN).c $(MAIN).h ; $(CC) -o $(MAIN).o $(CFLAGS) $(MAIN).c

$(MAIN).class: $(MAIN).java ; javac $(MAIN).java
$(MAIN).h: $(MAIN).class ; javah -jni -o $*.h $(MAIN)
```

The result of the compilation is :

```
I990557:/u/i990557/java/jni/pgm01: >make
javac HelloWorld.java
javah -jni -o HelloWorld.h HelloWorld
c89 -c -W c,expo,dll -DNEEDSIEEE754 -DNEEDSLONGLONG -o HelloWorld.o -I. -I/
usr/l
pp/java/J1.1/include -I/usr/lpp/java/J1.1/include/mvs HelloWorld.c
c89 -W l,dll -o libHelloJNI.so HelloWorld.o /usr/lpp/java/J1.1/lib/mvs/
native_th
reads/libjava.x /usr/lpp/java/J1.1/lib/mvs/native_threads/libJNIConvert.x
I990557:/u/i990557/java/jni/pgm01: >
```

- Our JNI sample application will display now:

```
I990557:/u/i990557/java/jni/pgm01: >java HelloWorld
Before: Value Before
After: Hello World, this is C ...
I990557:/u/i990557/java/jni/pgm01: >
```

TCP/IP COMMUNICATION

Socket communication can be easily realized between a Java client and a Java server. Java provides functions to manage TCP/IP communication.The java.net package provides the classes for implementing networking applications:

- Socket
- ServerSocket
- InetAddress.

I will use the same method I used to explain JNI concepts. I will use a very simple client/server application to demonstrate basic communication concepts.

Server – Java coding

This very basic server will:

- Open a ServerSocket on port number 5000.
- Process 4 client requests, exchanging simple strings of data.
- Then stop.

The whole server Java code looks like:

```
import java.io.*;
import java.net.*;           // import java.net package

public class Server
{
 public Server()
 {
  int Nb_client = 4;
  try
  {
   System.out.println("Server started...");
   ServerSocket server_socket = new ServerSocket(5000);
   System.out.println(">> Waiting for " + Nb_client + " client(s)...");

   for (int i = 1; i <= Nb_client; i++)
   {
    System.out.println(">> Waiting for client # " + i);
    Socket client_socket;

    client_socket = server_socket.accept();
    System.out.println("Client Socket opened...");

    BufferedReader read_buffer =           // input stream
    new BufferedReader(new InputStreamReader(client_socket.getInputStream()));

    BufferedWriter write_buffer =          // output stream
    new BufferedWriter(new
OutputStreamWriter(client_socket.getOutputStream()));

    System.out.println(read_buffer.readLine());

    write_buffer.write("Data from server...\n");
    write_buffer.newLine();
    write_buffer.flush();

    write_buffer.close();
    read_buffer.close();
    client_socket.close();  // close socket
   }
  }
  catch (Exception e) {System.out.println(e);}
```

```
 }
 public static void main(String args..)
 {
  Server s = new Server();
 }
}
```

I am going to try to comment step by step the structure of this program.

Step 1 : create a ServerSocket

The first action of the server is to create an instance of a ServerSocket on a specific TCP/IP port (in our case, the server will use port number 5000). This server socket will wait for requests to come in over the network:

```
ServerSocket server_socket = new ServerSocket(5000);
```

Step 2 : listen for a client

Then, the server should listen for a connection to be made to this socket and accepts it:

```
client_socket = server_socket.accept();
```

The method waits until a connection is made.

Step 3 : create read and write buffers to communicate with the client

At this point, the server must create input and ouput text streams to communicate with the client:

```
    BufferedReader read_buffer =           // input stream
    new BufferedReader(new InputStreamReader(client_socket.getInputStream()));

    BufferedWriter write_buffer =          // output stream
    new BufferedWriter(new
OutputStreamWriter(client_socket.getOutputStream()));
```

Step 4 : read and write data

To read and write data, the server has to use methods implemented on BufferedReader and BufferedWriter classes:

```
    System.out.println(read_buffer.readLine());

    write_buffer.write("Data from server...\n");
    write_buffer.newLine();                     // Write a line separator
```

```
    write_buffer.flush();                          // Send buffer
```

Step 5: close buffers and socket

When the communication is over, the server must close input/output buffers and the socket:

```
    write_buffer.close();
    read_buffer.close();
    client_socket.close();  // close socket
```

Client – Java coding

On the other hand, the client program will:

- Determine the TCP/IP address of the server using its hostname.
- Open a socket with the server.
- Exchanging simple strings of data.
- Then stop.

The whole client Java code looks like:

```
import java.io.*;
import java.net.*;

public class Client
{
 public Client()
 {
  try
  {
   String nom_serveur = new String("mzsmvs");     // hostname of the server
   int port_serveur = 5000;                       // tcpip port
   InetAddress address;
   Socket client_socket;

   System.out.println("Client started...");

   address = InetAddress.getByName(nom_serveur);  // get IP address

   client_socket = new Socket(address, port_serveur); // open socket

   BufferedWriter write_buffer =          // output stream
   new BufferedWriter(new
OutputStreamWriter(client_socket.getOutputStream(),"Cp037"));

   BufferedReader read_buffer =           // input stream
   new BufferedReader(new
InputStreamReader(client_socket.getInputStream(),"Cp037"));
```

```
   write_buffer.write("Data from client...");
   write_buffer.newLine();
   write_buffer.flush();

   System.out.println(read_buffer.readLine());

   write_buffer.close();
   read_buffer.close();
   client_socket.close();  // close socket
  }
  catch (Exception e) {System.out.println(e);}
 }
 public static void main(String args..)
 {
  Client c = new Client();
 }
}
```

In detail, the client has to execute the following steps.

Step 1: Get the IP address of the server

First, the client must determine the IP address of the host using its hostname:

```
address = InetAddress.getByName(nom_serveur);
```

Step2 : open a socket with the server

The client must open a TCP/IP socket with the server:

```
client_socket = new Socket(address, port_serveur); // open socket
```

Step 3 : create read and write buffers to communicate with the server

Then the client must create input and output buffers to communicate with the server on OS/390. The Java Virtual Machine (JVM) on OS/390 runs in an EBCDIC environment (Cp1047) whereas the JVM on the PC workstation runs in a Unicode environment.

This means that there is character conversion going on between the server and the client.

This can be done automatically using the codepage parameter Cp037 when creating input and output buffers:

```
   BufferedWriter write_buffer =          // output stream
   new BufferedWriter(new
```

```
OutputStreamWriter(client_socket.getOutputStream(),"Cp037"));

   BufferedReader read_buffer =           // input stream
   new BufferedReader(new
InputStreamReader(client_socket.getInputStream(),"Cp037"));
```

Step 4: read and write data

To read and write data, the client has also to use methods implemented on BufferedReader and BufferedWriter classes:

```
   write_buffer.write("Data from client...");
   write_buffer.newLine();
   write_buffer.flush();

   System.out.println(read_buffer.readLine());
```

Step 5: close buffers and socket

When the communication is over, the client must close input/output buffers and the socket:

```
   write_buffer.close();
   read_buffer.close();
   client_socket.close();  // close socket
```

Execute the sample application

After compiling the server and the client with javac, you can start the server on OS/390 and the client. If everything is set up correctly, you will get the following results for the Server.class:

```
I990557:/u/i990557/java/communication/pgm00: >java Server
Server started...
>> Waiting for 4 client(s)...
>> Waiting for client # 1
Client Socket opened...
Data from client...
>> Waiting for client # 2
Client Socket opened...
Data from client...
>> Waiting for client # 3
Client Socket opened...
Data from client...
>> Waiting for client # 4
Client Socket opened...
Data from client...
I990557:/u/i990557/java/communication/pgm00: >
```

And for the Client.class:

```
I990557:/u/i990557/java/communication/pgm00: >java Client
Client started...
Data from server...
I990557:/u/i990557/java/communication/pgm00: >
```

COMMUNICATION BETWEEN C/C++ AND ASSEMBLER PROGRAMS

I am going to show how to call an Assembler routine from a C/C++ program using some very simple programs.

Step 1: write the C/C++ program

The following C/C++ program initializes one integer variable, x, and one string variable, y. It passes control to an Assembler routine, which updates x and z values:

```
******************************** Top of Data ************************
  #include <stdio.h>
  #include <stdlib.h>

  #pragma linkage (ASM_F,COBOL)

  typedef void ASM_F(char*, int*);
  ASM_F *ASM_P;

  main(int argc, char *argv[])
  {
        int x;
        char y[16] = "Before";
        x = 3;

        printf ("In C/C++ program C\n");
        printf ("x = %d\n",x);
        printf ("y = %s\n",y);

                                        /* Fetch Assembler routine */

        ASM_P = (ASM_F*) fetch("C2ASMAS0");

                                        /* Call  Assembler routine */
        ASM_P(y, &x);

        printf ("Back in C/C++ program\n");
        printf ("x = %d\n",x);
        printf ("y = %s\n",y);
        return 0;
  }
```

Step 2: write the Assembler routine

```
C2ASMASO CSECT
C2ASMASO AMODE 31
C2ASMASO RMODE ANY
*
         SAVE  (14,12)
         BASR  R12,Ø
         USING *,R12                    R12 = BASE REGISTER
*
         LR    R9,R1                    SAVE PARAMETER ADDRESS
*
         GETMAIN R,LV=WORKL
*
         ST    R1,8(R13)
         ST    R13,4(R1)
         LR    R13,R1
         USING WORK,R13
*
         WTO   'IN C2ASMASO ROUTINE',ROUTCDE=(11)
*
         L     R3,Ø(,R9)                POINT TO Y
         L     R4,4(,R9)                POINT TO X
*
         LA    R6,5Ø                    UPDATE X
         ST    R6,Ø(,R4)
         MVC   Ø(8,R3),=CL8'AFTER'      UPDATE Y
*
RETURN   L     R13,4(R13)               RESTORE R13
         L     R1,8(R13)
         FREEMAIN R,LV=WORKL,A=(R1)
         L     R14,12(R13)
         LM    RØ,R12,2Ø(R13)
         SR    R15,R15                  SET UP RC
         BSM   Ø,R14                    RETURN
*
WORK     DSECT
SAVEAREA DS    18F
WORKL    EQU   *-WORK
*
         REGISTER
*
         END
```

Step 3: execute the C/C++ program

When you run C2ASMSO, you get the following result:

```
In C/C++ program C
x = 3
z = Before
```

```
Back in C/C++ program
x = 50
z = AFTER
```

When the C/C++ program gets control back after the Assembler routine, x and y values are updated.

Server implementation

The server is implemented using:

- Server.java – it is the main Java class.
- iplInfo.java – this Java class manages the communication with C/C++ routine iplInfo.c.
- iplInfo.c – this C/C++ routine is used to communicate with the Assembler program IPLCASO.
- IPLCASO – an Assembler program wich collects information about the last IPL.
- MVScmd.java – this Java class manages the communication with C/C++ routine MVScmd.c.
- MVScmd.c – this C/C++ routine is used to communicate with the Assembler program COMASO.

Server.java

```
import java.io.*;
import java.net.*;

public class Server
  {
    static final String IPLINFO_str    = "IPLINFO_";
    static final String STOP_str       = "STOP____";
    static final String SEND_CMD_str   = "SEND_CMD";

    public static void main(String[] args)
      {
       int server_port = 5000;
       if (args.length > 0)
       {
        server_port = Integer.parseInt(args[0]);
       }
```

```
       try
       {
        Socket client_socket;
        String input_data;
        String output_data;

        System.out.println("Server started using port " + server_port);
        ServerSocket server_socket = new ServerSocket(server_port);

        while(true)
        {
         System.out.println(">> Waiting for client...");
         client_socket = server_socket.accept();
         System.out.println("   Client Socket opened...");

         BufferedReader read_buffer =           // input stream
         new BufferedReader(new
InputStreamReader(client_socket.getInputStream()));

         BufferedWriter write_buffer =          // output stream
         new BufferedWriter(new
OutputStreamWriter(client_socket.getOutputStream()));


         input_data = read_buffer.readLine();

         System.out.println("   Data from Client : " + input_data);

         String request_type = input_data.substring(0,8);

         System.out.println("   Request type : " + request_type);

         if (request_type.equals(STOP_str))
               {
                output_data = "Server is stopping...";
                write_buffer.write(output_data);
                write_buffer.newLine();
                write_buffer.flush();
                write_buffer.close();
                break;
               }


         if (request_type.equals(SEND_CMD_str))
              {
               System.out.println("   ->> MVS command");
               String mvs_command  = input_data.substring(8);
               System.out.println("       MVS command  : " + mvs_command);
               MVScmd  mc = new MVScmd();
               mc.callASM(mvs_command);
```

```
              write_buffer.write("MVS Command issued...");
             }
          if (request_type.equals(IPLINFO_str))
             {
              System.out.println("   ->> IPLINFO");
              iplInfo ii = new iplInfo();
              ii.callASM();
              write_buffer.write(ii.myLine);
             }


          write_buffer.newLine();
          write_buffer.flush();

          write_buffer.close();
          read_buffer.close();
          client_socket.close();  // close socket
         }
        }
        catch (Exception e) {System.out.println(e);}
       }
  }
```

Server.make.

```
MAIN = Server
$(MAIN).class: $(MAIN).java ; javac $(MAIN).java
```

To use this makefile (whose name is not Makefile), you should enter the following command:

```
make -f Server.make
```

iplInfo.java.

```
import java.io.*;

public class iplInfo
  {
    String myLine = "Before";
    public native void callASM();
      static
      {
       System.loadLibrary("iplInfoJNI");
      }
  }
```

iplInfo.c.

```
  #include <stdio.h>
```

```
#include <stdlib.h>
#include <jni.h>
#include "iplInfo.h"

#pragma linkage (ASM_F,COBOL)

typedef void ASM_F(char*);
ASM_F *ASM_P;

JNIEXPORT void JNICALL Java_iplInfo_callASM
  (JNIEnv *env, jobject obj)
{
      char li1.8Ø. = "";

      int rc;
      jclass jcls;
      char *fieldName;
      char *signature;
      jfieldID field;
      const char *cstring;
      jstring jstr;
      jint count;

      /* Get a reference to the Class object */
      jcls = (*env)->GetObjectClass(env, obj);

      /* printf ("li1 = %s\n",li1);  */

                                    /* Fetch Assembler routine */

      ASM_P = (ASM_F*) fetch("IPLCASO");

                                    /* Call  Assembler routine */
      ASM_P(li1);

      /* printf ("li1 = %s\n",li1);   */

      /* === Manage String === */

      /* convert the name of the field to ascii */
      fieldName = "myLine";
      __etoa(fieldName);

      /* convert the signature to ascii */
      signature = "Ljava/lang/String;";
      __etoa(signature);

      /* obtain the field ID */
      field = (*env)->GetFieldID(env, jcls, fieldName, signature);

      /* create a new jstring */
```

```
        cstring = li1;
        rc = NewStringPlatform(env, cstring, &jstr, Ø);

        /* modify the String object in Java */
        (*env)->SetObjectField(env, obj, field, jstr);

}
```

iplInfo.make.

```
MAIN = iplInfo

CC = c89 -c -W c,expo,dll -DNEEDSIEEE754 -DNEEDSLONGLONG
CFLAGS := -I. -I/usr/lpp/java/J1.1/include -I/usr/lpp/java/J1.1/include/mvs
LL = c89 -W l,dll
LFLAG1 := /usr/lpp/java/J1.1/lib/mvs/native_threads/libjava.x
LFLAG2 := /usr/lpp/java/J1.1/lib/mvs/native_threads/libJNIConvert.x

$(MAIN): $(MAIN).o ; $(LL) -o libiplInfoJNI.so $(MAIN).o $(LFLAG1) $(LFLAG2)
$(MAIN).o: $(MAIN).c $(MAIN).h ; $(CC) -o $(MAIN).o $(CFLAGS) $(MAIN).c

$(MAIN).class: $(MAIN).java ; javac $(MAIN).java
$(MAIN).h: $(MAIN).class ; javah -jni -o $*.h $(MAIN)

To use this makefile (whose name is not « Makefile »), you should enter the
following command :

make -f iplInfo.make
```

IPLCASO ASSEMBLER PROGRAM

```
IPLCASO  CSECT
IPLCASO  AMODE 31
IPLCASO  RMODE ANY
*
* IPL PARMS
* ==========
*
* DATE AND TIME :
*
*     PSA — FLCCVT —> CVT — CVTSMCA —> SMCA
*     +++               +++                ++++
*
*                                    DATE:  SMCAIDTE CL4 ØØYYDDDF
*                                    TIME:  SMCAITME CL4 BINARY
*                                    SMFID: SMCASID
* SYSRES VOLUME:
*
*                       CVT — CVTSYSAD —> UCB
*                       +++                +++
*
```

```
*                                    ADDRESS: UCBNAME
*                                    VOLSER:  UCBVOLI
* IPLPARM:
*
*                        CVT - CVTSCPIN -> SCCB
*                        +++                 ++++
*
*                                    LOADPARM: SCCBPARM
*
*                        CVT - CVTASMVT -> ASMVT
*                        +++                 ++++
*
*                                    CLPA: ASMFLAG2
* MVS VERSION :
*
*                        CVT (PREFIX)
*                        +++
*
*                                    PRODUCT NAME : CVTPRODN
*                                    PRODUCT FMID : CVTPRODN
*
* COMMUNICATION AREA WITH C/C++
*
* OFFSET FIELD           LENGTH
*
* 0000   SMFID           004
* 0004   DATEJ           006
* 0010   TIME            008
* 0018   SYSRES - VOLSER 006
* 0024   SYSRES - DEVN   003
* 0027   LOADPARM        008
* 0035   SP VERSION      006
*
         STM   R14,R12,12(R13)
         BALR  R12,0
         USING *,R12                  R12 = BASE REGISTER
*
         LR    R2,R1                  SAVE PARAMETER ADDRESS
*
         LA    R0,WORKLEN
         GETMAIN R,LV=(R0),LOC=BELOW  NEED TO ALLOCATE SAVEAREA
*                                     BELOW THE LINE FOR A24 MODULE
*                                     IT IS ALSO TRUE FOR PARMS
         LR    R3,R1
         USING WORKAREA,R3
*
         ST    R1,ADDR                SAVE WORK ADDRESS FOR FREEMAIN
*
         ST    R1,8(R13)
         ST    R13,SAVEAREA+4
         LR    R13,R1
```

```
         MVC   MSG,=CL8Ø' '
         SR    R11,R11
         USING PSA,R11
         L     R4,FLCCVT
         USING CVTMAP,R4
         L     R5,CVTSMCA
         USING SMCABASE,R5
*
* SMFID
*
         MVC   MSG+ØØ(Ø4),SMCASID
*
* DATE
*
         LOAD  EP=DATEDSO              LOAD DATE CONVERTION ROUTINE
         ST    RØ,ADDRESSD
*
         LA    R1,PARMSD
         ST    R1,A2
         LA    R1,PARMLSTD
*
         MVC   DATEF,SMCAIDTE
         CLC   DATEF(1),=X'ØØ'
         BE    EQ19
         MVC   DATEF(1),=X'2Ø'
         B     COD
EQ19     EQU   *
         MVC   DATEF(1),=X'19'
COD      EQU   *
         L     R9,DATEF                YYYYDDDF
         SRL   R9,Ø4                   SHIFT TO RIGHT
*        MVC   COMMD,=XL4'Ø1999278'
         ST    R9,DATEF
         MVC   COMMD,DATEF
*
         L     R15,ADDRESSD
         BASSM R14,R15
*
         MVC   SMàDATE,SMCAIDTE
         BAL   R14,SMFàDATE
         MVC   MSG+Ø4(Ø6),SMàDCL6
*
* TIME
*
         MVC   SMàTIME,SMCAITME
         BAL   R14,SMFàTIME
         MVC   MSG+1Ø(8),SMàTCL8
*        BAL   R14,àPUT
* SYSRES
*
```

```
         L     R7,CVTSYSAD
         USING UCBOB,R7
         MVC   MSG+18(6),UCBVOLI
         MVC   MSG+24(3),UCBNAME
*
         DROP  R7
*
* LOADPARM
*
         L     R7,CVTSCPIN
         USING SCCB,R7
         MVC   MSG+27(Ø8),SCCBPARM
*
         DROP  R7
*
*
* MVS VERSION
*
         LA    R6,CVTMAP-CVTFIX
         LR    R7,R4
         SR    R7,R6                          POINT TO CVT PREFIX
         USING CVTFIX,R7
*
         MVC   MSG+35(8),CVTPRODN
         MVC   MSG+43(8),CVTPRODI
*
         L     R2,Ø(,R2)                      POINT TO COMMAREA
         MVC   Ø(8Ø,R2),MSG
*
         DROP  R7
*
RETURN   L     R13,SAVEAREA+4
*
         L     R1,ADDR                       FREE WORKAREA
         LA    RØ,WORKLEN
         FREEMAIN R,LV=(RØ),A=(R1)
*
         LM    R14,R12,12(R13)
         LA    R15,Ø
         BR    R14
*
*
ASMFLAG  DS    ØF
ASMFLAG1 DS    XL1
ASMFLAG2 DS    XL1
FILLER   DS    XL2
IOTOKEN  DS    CL48
*
*
* ROUTINE TO CONVERT SMF DATE FROM FULLWORD TO CL6'YY.DDD'
```

```
*
SMFàDATE DS    ØH
         UNPK  SMàDCL5,SMàDATE            X'ØØYYDDDF' TO C'YYDDD'
         MVC   SMàDCL6+Ø(2),SMàDCL5
         MVC   SMàDCL6+3(3),SMàDCL5+2
         BR    R14
*
SMàDCL5  DS    CL5
SMàDCL6  DC    CL6'YY.DDD'
SMàDATE  DS    XL4
*
*
* ROUTINE TO CONVERT SMF TIME (CENTISEC.) FROM FULLWORD
* TO CL8'HH:MM:SS'
*
SMFàTIME DS    ØH
         ICM   R7,B'1111',SMàTIME          TOD IN CENTISECONDS
         SR    R6,R6
         LA    R8,1ØØ
         DR    R6,R8                       R7 = TOD IN SECONDS
*
         SR    R6,R6
         LA    R8,6Ø                       DIVIDE BY 6Ø => SS
         DR    R6,R8
         CVD   R6,SMàTDW                   SS VALUE FOR HHMMSS
         UNPK  SMàTCL8+6(2),SMàTDW+6(2)
         OI    SMàTCL8+7,X'FØ'             REVERSE LAST DIGIT
*
         SR    R6,R6
         LA    R8,6Ø                       DIVIDE BY 6Ø => MM
         DR    R6,R8
         CVD   R6,SMàTDW                   MM VALUE FOR HHMMSS
         UNPK  SMàTCL8+3(2),SMàTDW+6(2)
         OI    SMàTCL8+4,X'FØ'             REVERSE LAST DIGIT
         CVD   R7,SMàTDW                   HH VALUE FOR HHMMSS
         UNPK  SMàTCL8+Ø(2),SMàTDW+6(2)
         OI    SMàTCL8+1,X'FØ'             REVERSE LAST DIGIT
         BR    R14
SMàTDW   DS    D
SMàTCL8  DC    CL8'HH:MM:SS'
SMàTIME  DS    XL4
* ROUTINE TO PUT MSG FIELD IN DDNAME REPORT USING AS24SO
àPUT     DS    ØH
         LA    R1,PARMLST
         MVC   TYPE,=C'P'              PUT
         L     R15,ADDRESS
         BASSM R1Ø,R15
         BR    R14
*
WORKAREA DSECT
```

```
SAVEAREA DS    18F
ADDRESS  DS    F                       ADDRESS OF A24 MODULE
ADDRESSD DS    F                       ADDRESS OF DATE CONV MODULE
ADDR     DS    F                       ADDRESS OF WORKAREA
PARMS    DS    ØF
TYPE     DS    C
MSG      DS    CL8Ø
PARMLST  DS    ØF
A1       DS    F
PARMSD   DS    ØF
COMMD    DS    CL15
DATEF    DS    F
PARMLSTD DS    ØF                      PARMS TO PASS TO DATE ROUTINE
A2       DS    F
WORKLEN  EQU   *-WORKAREA
         REGISTER
*
*
          IHAPSA LIST=YES
          CVT   DSECT=YES,LIST=YES,PREFIX=YES
          IEESMCA
          IHAPCCA
          IEFUCBOB
          IHASCCB
*         ILRASMVT
*
         END
```

MVScmd.java.

```
import java.io.*;

public class MVScmd
  {
    public native void callASM(String s);
      static
      {
       System.loadLibrary("MVScmdJNI");
      }
    public static void main(String.. args)
      {
       String myLine = "D A,i9905*";
       MVScmd ii = new MVScmd();
       ii.callASM(myLine);
      }
  }
```

MVScmd.c.

```
  #include <stdio.h>
  #include <stdlib.h>
```

```
#include <jni.h>
#include “MVScmd.h”

#pragma linkage (ASM_F,COBOL)


typedef void ASM_F(char*,int*);
ASM_F *ASM_P;

JNIEXPORT void JNICALL Java_MVScmd_callASM
  (JNIEnv *env, jobject obj, jstring jstr)
{
        char *cmd;
        const char *eretstr = “This string is returned”;
        jstring jretstr;
        jint length;
        jint ret;
        int cmdl;

        /* Turn jstring into EBCDIC string */

        ret = GetStringPlatformLength(env, jstr, &length, 0);

        cmd = (char*) malloc(length);
        cmdl = (int) length;
        cmdl—;
        ret = GetStringPlatform(env, jstr, cmd, length, 0);

        /* printf(“Native method received = %d\n”, length);  */
        /* printf(“Native method received = %s\n”, cmd);     */


                                        /* Fetch assembler routine */

       ASM_P = (ASM_F*) fetch(“COMASO”);

                                        /* Call  assembler routine */
       ASM_P(cmd,&cmdl);

}
```

Editor’s note: This article will be concluded in the next edition.

Patrick Reynard
Systems Programmer (France)

MVS news

IBM has announced Version 3.0 of its VisualAge COBOL for Windows NT, with claimed better OS/390 host connectivity, a better interface for the Workframe project tool, an improved common tool to debug workstation and remote OS/390 applications, support for the development of DB2 stored procedures for OS/390 systems, and HTML-based on-line help.

Remote edit/compile/debug provides a workstation environment for developing and maintaining COBOL applications targeted for the host.

Version 3.0 includes updates to improve host connectivity, simplify the setup tasks, and provide new graphical interfaces on the workstation for interacting with the host.

A new remote file access system client is introduced for Version 3.0 that utilizes the IBM HTTP Server in OS/390 rather than requiring any separate NFS client products. This is designed to simplify the setup between the workstation and the host by reducing the number of potential conflicts with existing host software.

Also new are workstation graphical user interfaces for completing programming tasks that require interaction with the host without having to log on to it.

The job monitor interface lets users submit a job to an OS/390 host and then perform actions on the job such as view status, view output, cancel, release, hold, and purge.

Contact your local IBM representative for further information.
http://www.ibm.com

* * *

Advanced Software Products Group (ASPG) has announced the North American release of its MegaCryption cryptography for MVS, to encrypt and decrypt any file in the MVS environment.

It is said to provide protection during data transmission and acts as an additional line of defence to current security measures. If security is penetrated, internally or externally, MegaCryption encoded files cannot be accessed without the proper encryption key.

It provides encryption/decryption, signing, and integrity-checking in one utility. It incorporates three industry-compliant algorithms: DES, Triple-DES, and Blowfish.

For further information contact:
Advanced Software Products Group, Inc,
South Naples, FL 34104, USA.
Tel: 941 649 1548
Fax: 941 649 6391

http://www.aspg.com

* * *

Xephon will be holding its annual *MVS 2000* conference at the Radisson Mountbatten Hotel in London, 7-8 June 2000. *MVS 2000* is designed specifically for technical managers, systems programmers, strategic planners, and other system specialists at MVS/ESA and OS/390 installations.

The attendance fee for *MVS Update* subscribers is £570.00 plus £66.50 VAT. For further information, please telephone the registrar, Angela Scott, on (01635) 33823.

* * *