51 TCP/SNA update

September 2003

In this issue

TCP/SNA Update

Published by

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

North American office

Xephon
Post Office Box 350100
Westminster, CO 80035-0100
USA
Telephone: (303) 410-9344

Subscriptions and back-issues

A year's subscription to *TCP/SNA Update*, comprising four quarterly issues, costs £130.00 in the UK; $190.00 in the USA and Canada; £136.00 in Europe; £142.00 in Australasia and Japan; and £140.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the March 1999 issue, are available separately to subscribers for £33.00 ($49.50) each including postage.

Editor

Fiona Hewitt

Contributions

When Xephon is given copyright, articles published in *TCP/SNA Update* are paid for at £170 ($260) per 1000 words for original material. To find out more about contributing an article, please download a copy of our *Notes for Contributors* from http://www.xephon.com/index/nfc

TCP/SNA Update on-line

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

Printed in England.

Making plans to exploit the new z990 mainframe

The new 32-way-capable z990 mainframe has been shipping since mid-June, and you can also upgrade to it from any z900 model except the 100. This article examines some of the key things that the TCP/SNA community should now be planning for, including:

- Absence of SNA management repertoire.
- Inability to use parallel channels.
- Bypassing of z/OS Version 1 Release 1.
- The new 'exploitation' mode operating systems – especially z/OS 1.6, scheduled to be available in September 2004.
- Much greater OSA-Express, ESCON, and FICON connectivity.
- Mainframe workload consolidation possibilities.
- Linux options.
- Increased use of TCP/IP HiperSockets.

The 40th anniversary of IBM mainframes will be upon us next year, and 2004 will also be the 30th birthday of SNA, which until recently was an integral part of mainframe computing. The new z990 mainframe – the first major enhancement to the 64-bit z900 family introduced in October 2000 – reaffirms the durability of IBM mainframe computing, but is also a timely reminder of how TCP/IP continues to usurp SNA.

With the z990 mainframe, another trademark SNA capability bites the dust: z990 mainframes will not support SNA Operations Management, either in the context of NetView or the Systems Automation for OS/390 products. With the universal move towards TCP/IP-centric networking, IBM feels, justifiably, that it's time that data centres standardized on Simple Network Management Protocol

(SNMP) for all of their management needs. So, when you finally upgrade to a z990, you'll have no choice but to migrate to NetView's SNMP agent in order to issue operations management commands. If you're already using IBM's System Automation for OS/390, you'll have to upgrade to Version 2.2 (or later) and redefine the automation policies using the SNMP API.

Despite de-emphasizing SNA, a major highlight of the z990 is the significant enhancement it offers when it comes to overall mainframe connectivity. The z990 shatters many previous barriers when it comes to channels, mainframe 'virtualization', and TCP-based intra-mainframe communications.

Backwards compatibility is a hallmark of mainframe computing. Continuous technological evolution, however, has underscored the four decade time continuum spanned by mainframes, with some of the technical innovations falling into the category of quantum leaps. Pivotal among these was virtual storage (VS), which became available on the System/370 in 1972. 31-bit addressing came along in 1981 and 64-bit addressing came to be in 2000, with 1990 seeing the introduction of both sysplex and ESCON. Hardware-based logical partitioning (LPARs) with Processor Resource/System Manger (PR/SM) came along in February 1988, with a maximum of four LPARs per mainframe.

MORE PARTITIONS AND I/O FOR SERVER CONSOLIDATION

The z990, with a tongue-in-cheek codename of 'T-Rex' to indicate that the supposed dinosaur is still very much on the prowl, extends the LPAR limit to 30. This doubles the previous LPAR limit set in June 1997. The other significant evolutionary leap made by the z990 – which, by the way, also doubles the number of processors per mainframe from 16 to 32 – has to do with channel I/O. It breaks the hitherto set-in-concrete 256 channels per machine barrier. With the z990, it's possible to have 512 ESCON channels per z990 – albeit in the form of two, 256-channel Logical Channel SubSystems (LCSSs) per system. The z990, however, doesn't support parallel channels. This is consistent with a statement of direction made by IBM back in

October 2000, when the z900 was being unveiled. If you still need to use I/O devices with parallel channel interfaces, you'll have to use a parallel channel converter such as IBM's 9034-001 ESCON Converter Model 1.

The z990 also doubles the number of OSA-Express ports possible on a system from 24 to 48, while increasing the 2 gigabit/second FICON Express channel capability from 96 to 120. It's also now possible to have up to 16, high-speed, 'TCP/IP network inside a mainframe' HiperSockets 'internal LANs' per z990. HiperSockets, introduced in October 2001, and as yet specific to the zSeries, provides for ultra-high-speed, 'near zero' latency TCP/IP communications between programs running on z/OS, z/VM, Linux for zSeries, and guest operating systems running on top of z/VM. The programs that can communicate with each other using HiperSockets can be in the same LPAR or on different LPARs with the same z990.

The bottom line here is that with the z990, IBM has redefined the I/O and networking capabilities of mainframes to reflect the changing role of these data centre stalwarts. The nature of the workloads being handled by mainframes is shifting fundamentally. Linux is beginning to make its mark on the mainframe world. IBM states that 17% of 2002 mainframe revenues came from Linux workloads, and also that 2002 saw more than 100 brand new mainframe customers. Some of these must have been lured towards mainframes by their powerful, compelling, and unique Linux capabilities. There is no other box that can so elegantly run as many Linux server images as a z900 or z990 mainframe.

To understand the allure of mainframes with regard to Linux, especially in the context of the Web, you need to look at companies such as Google, the Web search company. Right now, Google uses as many as 10,000 PC-based Linux servers to handle the 200 million queries it services a day, and to search and index the 3 billion Web pages that it has in its sights. While this 10,000 server configuration is certainly at the high-end of the server-farm spectrum, other companies – for example, large ISPs, e-retailers (eg eBay), public portals (eg AOL), telcos, the large automotive concerns, aerospace contractors, and the large travel industry players – have sprawling server farms too.

Blade computing as advocated by Sun (and even IBM), where you can typically pack around 16 servers per 3U (ie 5.25 inch) high space within a standard rack-mount 'shelf', is one (albeit uninspiring) approach to rationalizing this growing demand for co-located Unix/Linux servers. The other is to use a z900/z990 mainframe – particularly with z/VM, which will allow you to run tens of thousands of Linux images within each z/VM LPAR. To put it another way, rather than maintaining 10,000 PCs running Linux, Google could run all those Linux images on a single mainframe. And that's the rub. Maintaining large numbers of individual servers is complicated, inconvenient, and costly. Mainframe-based server consolidation, one of the key value propositions being put forward for today's brawny mainframes, is an elegant and extensible solution to this problem.

But server consolidation as offered by the z9*xx* isn't limited to Linux/Unix consolidation. These 32-way-capable machines, with highly flexible capacity-on-demand features (including the new temporary 'On/Off Capacity on Demand') are powerful enough to

	Intro-duced	**MIPS per CPU**	**Δ**	**Max no CPUs per machine**	**Max MIPS/ machine**	**Max memory**	**Max LPAR**	**Max OSA-Express**	**Max FICON**
S/390 G1	Sept '94	11-13		6-way	60		10	0	
S/390 G2	June '95	22	83%	10-way	165		10	0	
S/390 G3	Sept '96	45	105%	10-way	325		10	0	
S/390 G4	June '97	63	40%	10-way	450		15	0	
S/390 G5	Aug '98	152	141%	10-way	1,069	24GB	15	12	12
S/390 G6	May '99	201	32%	12-way	1,614	32GB	15	12	24
z900	Oct '00	225(?)		16-way	2,500	64GB	15	24	96
z900 Gen 1.5	April '02	270(?)	20%	16-way	2,925	64GB	15	24	96
z990	**May '03**	**410(?)**	**52%**	**32-way**	**8,134**	**256GB**	**30**	**48**	**120**
z800	Feb '02	185(?)		4-way	625	32GB	15	24	32

Figure 1: z990 compared with previous IBM mainframes

enable workloads from multiple existing mainframes to be consolidated onto one z990.

The z990 therefore sets out to address two distinct types of server consolidation:

- Mainframe workload consolidation, ie multiple mainframes to one z990.
- Linux server consolidation, ie multiple Linux servers to one z990 or one z990 partition running z/VM.

However, before looking at server consolidation mechanics, let's just examine how the z990 compares with previous IBM mainframes. Figure 1 clearly shows that the z990 has the raw horse-power in terms of MIPS as well as storage (ie up to 256GB) to accommodate workloads currently being run on multiple mainframes. Reliability and single-points of failure are really not an issue, with the z990 always providing built-in spare processors, a dual interconnect fabric to prevent full memory loss, hot-swappable I/O cards, and spare ESCON ports on each ESCON adapter. The mean time between failure (MTBF) of the z990 is higher than that of any previous mainframe and is reputed to be longer than the career span of an IT professional! What's more, this degree of mainframe consolidation will reduce overall operational costs and complexity. Hence the appeal.

The doubling of the LPAR limit to 30 and of ESCON channels to 512 makes it easy to port existing mainframe workloads 'in situ', maintaining their current LPAR and I/O structure. Appreciating that doubling these two key parameters will only go so far when it comes to concerted mainframe consolidation in the future, IBM has already stated in terms of a statement of direction that these limits will again be doubled to 60 and 1,024 (ie 4 LCSSs) in the future. This is most likely to occur towards the end of 2004, once z/OS 1.6 arrives.

EXPLOITING THE OPERATING SYSTEMS

The current mainframe OSs obviously don't support the 30 LPARs or 512 ESCON channels possible with the z990. In order

COMPATIBILITY MODE:

1 Up to 15 LPARs
2 Single Logical Channel Subsystem (LCSS); ie max 256 channels

Operating system	Versions	Availability
OS/390	Ver. 2 Rel. 10	June 2003
z/OS	Ver 1 Rel 2, Rel 3 & Rel. 4 Ver 1 Rel 1 [64-bit only]	
z/VM	Ver 3.1.0 Ver 4.2.0, 4.3.0, 4.4.0	
Linux for zSeries	Red Hat 7.1 & 7.2 SuSE SLES7 & SLES8	
VSE/ESA [31-bit]	2.5, 2.6, 2.7	
TPF/ESA [31-bit]	4.1	

EXPLOITATION MODE:

1 Up to 30 LPARs
2 Two Logical Channel Subsystem (LCSS); ie max 512 channels

Operating system	Versions	Availability
OS/390	N/A	N/A
z/OS	Ver. 1 Rel. 4 & Rel. 5	October 2003
z/VM	4.4.0	August 2003
Linux for zSeries	Red Hat 7.1 & 7.2 SuSE SLES7, SLES8 & SLES9	4Q2003
VSE/ESA	N/A	N/A
TPF/ESA	N/A	N/A

Figure 2: Key OS support on the z990

to realize this support, you need what IBM now refers to as the 'exploitation mode' operating systems. These are in effect new releases of z/OS, z/VM, and Linux for zSeries that will start to be available as of August 2003. Previous versions of mainframe OSs supported on the z990 are referred to as working in 'compatibility mode'. Note that z/OS Version 1 Release 1 is not supported on the z990. Figure 2 shows the key OS support on the z990, categorized in terms of compatibility and exploitation mode.

There are, however, still some key limitations even with the exploitation mode OSs. The most significant of these is that z/OS cannot yet support more than 16 CPUs per image. This means that provisioning an LPAR with more than 16 'business' CPs doesn't buy you anything. Recognizing this restriction, IBM has made sure that the z990 works only in LAPR mode. There is therefore no longer a notion of an unpartitioned base mode z990. This partitioning ensures that 17- to 32-way z990s will always be suitably partitioned so that no one LPAR has more than 16 business CPs.

IBM has already stated that a pivotal new version of z/OS – z/OS 1.6, expected to be available in September 2004 – will support more than 16 CPs per single image of z/OS, as well as supporting 60 LPARs and up to four LCSSs, each with 256 ESCON channels. The machines supported by z/OS 1.6 will be deemed to conform to a new zSeries 'Architecture Level Set'. This architecture level set, and consequently z/OS 1.6 (and greater), will be supported only by z800, z900, and z990 machines. ESA/390 architecture, ie 31-bit mainframes, will no longer be supported. As such, z/OSs after 1.5 will not work on any System/390 machines (even the latest G5 or G6 ones) or Multiprise 3000 Enterprise Servers.

MAINFRAME WORKLOAD PORTING

The z990 I/O subsystem is made up of I/O cages; there can be a maximum of three I/O cages per system, each with 28 I/O slots. It's therefore possible to have up to 84 I/O slots in a fully

z990 model	Processor books	'Business' Processor Units (PUs)	System Assist Process-ors (SAPs)	Spare processors	Memory	STI buses	Max I/O cages
A08	1	8	2	2	8 – 64GB	12	Three (3)
B16	2	16	4	4	8 – 128GB	24	Three (3)
C24	3	24	6	6	8 – 192GB	36	Three (3)
D32	4	32	8	8	8 – 256GB	48	Three (3)

Figure 3: The four z990 models

configured three-cage z990. Each z990 ships standard with one I/O cage in the so-called 'A-Frame' chassis (or rack). The Central Electronic Complex (CEC), which contains all of the processors, memory, and I/O adapters, structured within sealed, self-contained units known as 'books' (with a maximum of four books per z990), is also housed in the 'A-Frame', above the I/O cage. The additional I/O cages need to be housed in a second chassis referred to as the 'Z-Frame'. The whole purpose of the Z-Frame, which goes to the left of the 'A-Frame' when looking at a z990, is to hold these additional I/O cages.

With the new book-based architecture, you cannot, however, arbitrarily keep on adding I/O slots to the z990 without also having to increase the number of processors (ie the model) at some point. This has to do with the now 2GB/sec Self Timed Interconnect (STI) buses used for all of the system I/O (through Memory Bus Adapters (MBAs) that interface the STIs to main memory via the L2 cache). Each book supports 12 STIs. In turn, a single STI can support four I/O slots within a cage. You therefore need 7 STIs to support all 28 I/O slots in a cage. The up-to-8-way entry-level z990 Model A08 has only has one book, and therefore just 12 STIs (see Figure 3). A model A08 can therefore support only 48 (ie 12 x 4) I/O slots.

This means that if you install a second I/O cage on an A08 you can't use all 56 slots. To be able to do this, you need more STIs – and you can only get more STIs by purchasing another whole book. However, there is a fixed correlation between the number of books per z990 and the model number, with each ascending model number having one more book, up to the current maximum of four books. So, to be able to use all 84 I/O slots you need 21 STIs. To get 21 STIs you need to have a two-book model B16. The additional STIs available with the C24 and D32 indicate how IBM will increase ESCON channel connectivity alone to 1,024 channels within the next two years. You should therefore already be able to envisage a three-Frame z990 with five I/O cages and 140 I/O slots.

In marked contrast to IBM mainframes before this, it's no longer possible with the z990 to determine the number of active Processor Units (PUs) in a machine just from its model number – ie A08, B16, C24, D32 – since each model now offers a wide range of PU options. In order to facilitate software billing, there will now be a 'software' model associated with the number of PUs that are characterized as Central Processors (CPs or CPUs). This number will be obtained through the use of the Store System Information (STSI) instruction.

There will be no affinity between the hardware model and the number of active CPs within a machine. For example, it would be possible to have a model C24 which has eight PUs characterized as CPs. For software billing purposes, the STSI instruction would report 308. Since this represents a significant 'over-booking', it would be more normal for a customer to get a B16 if all they want initially is just eight active CPs. The STSI for such a machine would also say 308. This would also be the case if you were using a A08 or D32 with just eight active CPs.

The z990 I/O subsystem is sub-divided into LCSSs. With exploitation mode it would be possible to have two LCSSs per z990. This will be increased to four LCSSs within the next two years, especially after z/OS 1.6. The LCSS split is transparent to the software running in individual LPARs. Each LCSS can have from 1 to 256 channels. Each LCSS also supports from 1 to 15 LPARs. Each LPAR can be associated with only one LCSS. Within the context of a single LCSS, LPARs can enjoy Multiple Image Facility (MIF) channel-sharing. Currently, however, the only way to share spanned channels on different LCSSs among LPARs running on different LCSSs is by using HiperSockets or the internal channel coupling facility for parallel sysplex.

With a 15 LPAR LCSS with 256 channels, you can essentially create an identical image of any previous mainframe – given that 15 LPARs and 256 channels was the most supported by all previous mainframe models. So this LPAR/LCSS model provides you with 'stand-alone' mainframe images to facilitate mainframe workload porting.

For example, you could move the entire workload from a z/900 with LPARs and 200 channels onto a single LCSS image – and do so maintaining the exact LPAR split used before, irrespective of whether the LPARs were running z/OS, z/VM, or Linux. Even after moving such a big (15 LPAR) workload, the z990 still has space to accommodate another equally large workload. And that's with the current two LCSS capability; with four LCSSs, a single z990 will be able to run the workload of four 'fully maxed' z900s or System/390s.

LINUX ON THE z990

Mainframes are the perfect 'industrial strength' servers for Linux. They provide unparalleled scalability, legendary availability, and performance on tap, as well as the ability to run tens of thousands of Linux server images on a single hardware platform.

There are three very distinct modes through which Linux is supported on a z990:

- For a start, you can run SuSE Linux, Turbolinux, or Red hat Linux on any dedicated z990 LPAR using one or more of the standard CPs.
- Then there's IFL – the Integrated Facility for Linux. IFL is a Linux-only z990 engine. Any of the 'business' PUs available with a z990 can be designated as being an IFL. It would therefore be possible, in theory, to have an all-Linux, 32-way z990 D32 with all the CPs running as IFLs. The rationale for

using IFL is that IFL processors don't affect a z990's model designation when it comes to other software – since an IFL will run only Linux, IBM is happy to concede that IFL processors don't need to be included when calculating 'tier pricing' for non-Linux software.

- However, z/VM is the strategic and optimum way to truly exploit a z990 when it comes to Linux. z/VM specifically sets out to support very large numbers of concurrent Linux server images. (Note that the Virtual Image Facility (VIF) for Linux, which was promoted around 2001 as a means of running large numbers of Linux images on a mainframe, has now been withdrawn, since z/VM can do even better on its own.)

BOTTOM LINE

Whichever way you look at it, the z990 is a redoubtable offering. It pushes the mainframe envelope even further, and reaffirms that nothing can touch a big-block IBM mainframe when it comes to an industrial strength, 99.999% uptime, high-performance, ultra-scalable server. Thanks to IBM's concerted efforts over the last few years, mainframe computing is healthier than it has been for some time. The new LPAR/LCSS models, with HiperSockets to link everything together, make the new z990s even more compelling. It's now possible to effortlessly consolidate multiple mainframe workloads, not to mention Linux servers, into a single z990. The future upgrades that will emerge with z/OS 1.6 in September 2004 will further facilitate such workload consolidation. This is the new face of IBM mainframe computing. It's time to start planning how best to exploit the unprecedented performance, connectivity, and scalability being offered by this renaissance machine with the provocative 'T-Rex' codename.

Anura Gurugé
Strategic Consultant (USA)

TCP/IP performance monitoring review

This article reviews what can cause problems on your TCP/IP network or stack, explains how to find the source of the problems, and considers what changes or solutions are available to prevent the problem from happening again.

The first question you need to ask is, what is the TCP/IP network anyway? When there's a performance issue, you may have a problem in any of the following: TCP/IP stack, VTAM buffer pools, Communications Storage Manager (CSM), Unix Systems Services (USS), ftp, telnet/tn3290, listeners (Socket applications), routers, servers, or network connections.

In the sample architecture shown in Figure 1, the TCP/IP stack and VTAM are both controlling telecommunications access to

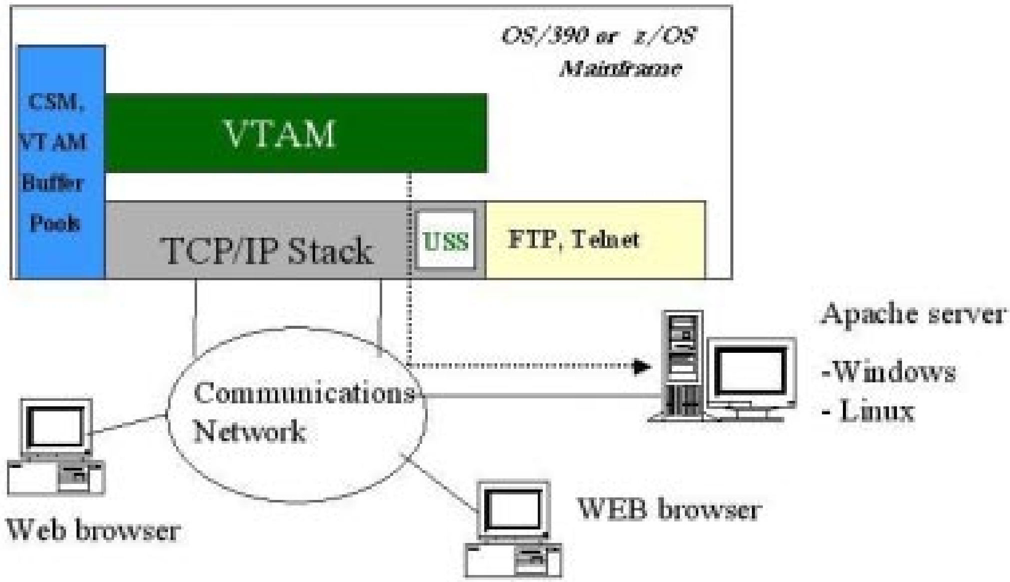


Figure 1: TCP/IP architecture

the mainframe. They run inside the OS/390 or z/OS mainframe and share use of the Communications Storage Manager (CSM) and VTAM buffer pools. Unix Systems Services is a part of the TCP/IP stack, and many critical functions such as the ftp server run under USS.

The other important pieces within the mainframe are the socket applications: these are what you need for your business functions. Socket applications vary from the 'well-known' ports such as ftp, http, or telnet, to DB2, CICS, MQSeries, and many others either developed in-house or sold by independent software vendors.

The TCP/IP communications network consists of communications lines, routers, hubs, servers, and many other devices. When there's a problem with performance, you need to be able to look both outbound to the communications network and inbound to the TCP/IP stack to MVS processing parameters.

TCP/IP STACK FUNCTIONS

Before we consider performance monitoring, we need to understand how the TCP/IP stack functions. The TCP/IP stack implements the four core Internet protocols: TCP, IP, UDP, and ICMP.

Figure 2 shows first the communications network – this is the actual physical hardware. The next layer is the Internet Protocol (IP), whose function is to make sure that each packet gets to the

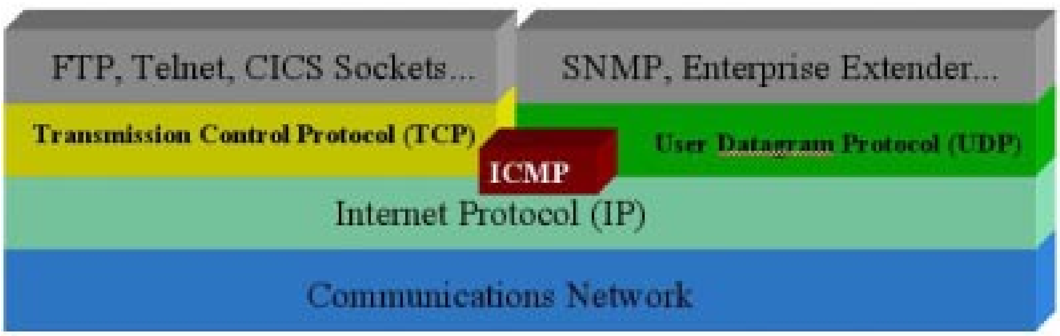


Figure 2: Core Internet protocols

right destination. IP handles addressing and discarding of packets.

You can use either the User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) to manage the assembling of a message or file into smaller packets. These packets flow over IP to another TCP or UDP that reassembles the packets into the original message. The basic difference between UDP and TCP is that TCP has an awareness of a connection between the two end-points until the connection is explicitly broken. UDP is a ‘connection-less’ protocol, which means that the UDP packet is sent to the other end but there is no on-going path between the two end-points.

Although the Internet Control Message Protocol (ICMP) uses the basic support of IP as if it were a higher level protocol, ICMP is actually an integral part of IP and must be implemented by every IP module (RFC792).

This is only scratching the very surface of the functions of the TCP/IP stack, and more explanation may be found in any of the excellent books that deal with this topic. But for the purposes of this article, we just need to be sure that all these functions are implemented in the mainframe TCP/IP stack. When monitoring, one of the key aspects is to make sure that basic TCP/IP stack functions are working properly.

TCP/IP PERFORMANCE PROBLEMS

When you make changes to try to improve TCP/IP performance, you need to be careful. A change made in one area can impact another area that you may never have suspected. Here is an e-mail I received a while ago:

“Within the last 7 months we made the quantum leap from OS/390 1.3 to OS/390 2.10. With this change we have been experiencing many interesting experiences with our TCP/IP stack. We had very limited prior use of TCP/IP but with the gates being open the flood of activity is growing quicker then we ever expected for our development department.

With this growth we have noticed (painfully) that TCP/IP has performance issues. For example one of our new applications uses a [third party application] to transmit gigabytes of data and they asked us to change our receive and send buffers to 131K.

After the change was made the [third party application] didn't perform any better or worse, but (the big one) a new high profile web application began getting very erratic response with delays up to 10 minutes.

We backed out the change."

I have sometimes been asked if a TCP/IP monitor can automatically fix problems. I would say that some problems which are clearly defined can be automated. But other problems, especially in the area of performance tuning, may be too complex to ever be automated.

The key to automation and even guidelines for performance tuning is that the problem must be well understood. In many networks, the complexity is daunting. With so many different types of application and combinations of equipment, it's extremely difficult to properly understand all the variables involved. This article aims to just get you started.

TOOLS TO FIND PROBLEMS

The tools that are currently available to help you find the source of any problems are as follows:

- Netstat, PING, USS, and TraceRoute commands
- SMF records
- SNMP (Simple Network Management Protocol)
- MVS/VTAM commands
- Traces
- TCP/IP monitors.

These are examined in detail below.

Netstat commands

Netstat commands exist for all TCP/IP networks – on Windows PCs, Unix, and the OS/390 or z/OS mainframe. In the workstation environments, the commands themselves are different, but they are a way to display information known to TCP/IP about connections and configuration. Some sample Netstat commands on the mainframe are shown below:

- NETSTAT All – details/debugging information
- NETSTAT AllConn – display all connections
- NETSTAT ARP – query ARP cache for a single address
- NETSTAT ARP All – query ARP cache for all addresses
- NETSTAT BYTEinfo Idletime – byte counts and idle time for connections
- NETSTAT BYTEinfo – byte counts for connections
- NETSTAT CACHinfo – fast response cache accelerator statistics
- NETSTAT CLients – client authorization and usage
- NETSTAT CONFIG – TCP/IP stack configuration
- NETSTAT COnn – display connections not closed or time-wait
- NETSTAT DEvlinks – display devices and status
- NETSTAT Gate – display base gateway configuration
- NETSTAT Gate Detail – display detailed gateway configuration
- NETSTAT Home – display home list
- NETSTAT IDS – intrusion detection services statistics.

With Netstat commands, you need to know what you're looking for. So how do you know when something is a problem? Let's take as an example a Netstat Byteinfo Idletime response, as

NETSTAT BYTEinfo Idletime - Byte Counts and Idle Time for Connections
Address:ansynova.no-ip.com

```
MVS TCP/IP NETSTAT CS V1R2       TCPIP NAME: TCPIP           10:00:20
11/08/2002          MVS TCP/IP REAL TIME NETWORK MONITOR
USER ID  B OUT   B IN    LPORT FOREIGN SOCKET          STATE    IDLETIME
-------- ------- ------- ----- ----------------------- -------- --------
$SLAPSRV 0000000 0000000 07980 0.0.0.0..0              LISTEN   31:57:02
$SLAPSRV 0000000 0000000 07979 0.0.0.0..0              LISTEN   31:57:02
$2CIPSRV 0000000 0000000 44444 0.0.0.0..0              LISTEN   31:56:31
$2CIPSRV 0000000 0000037 44444 63.187.73.176..4781     ESTABLSH 00:00:00
BPXOINIT 0000000 0000000 10007 0.0.0.0..0              LISTEN   99:59:59
FTPD1    0000000 0000000 00021 0.0.0.0..0              LISTEN   99:59:59
INETD4   0000000 0000000 00513 0.0.0.0..0              LISTEN   99:59:59
INETD4   0000000 0000000 01023 0.0.0.0..0              LISTEN   99:59:59
OSNMPD   0000000 0000000 01027 0.0.0.0..0              LISTEN   99:59:59
PORTMAP  0000000 0000000 00111 0.0.0.0..0              LISTEN   99:59:59
TCPIP    0000000 0000000 00023 0.0.0.0..0              LISTEN   99:59:59
TCPIP    0000000 0000000 01025 0.0.0.0..0              LISTEN   99:59:59
TCPIP    0000000 0001402 01026 127.0.0.1..1025         ESTABLSH 99:59:59
TCPIP    0001402 0000000 01025 127.0.0.1..1026         ESTABLSH 99:59:59
$SLAPSRV 0786509 0000000 01091 *..*                    UDP      31:57:01
$SLAPSRV 0000000 0786509 07981 *..*                    UDP      31:57:01
OSNMPD   014238K 016413K 00161 *..*                    UDP      41:38:33
PORTMAP  0000000 0000000 00111 *..*                    UDP      99:59:59
CONNECTIONS DISPLAYED: 18
```

Figure 3: Netstat Byteinfo idletime

shown in Figure 3. For the novice user, it wouldn't be obvious whether or not there was any kind of problem in the display (there isn't).

The display shows a number of connections which are in 'Listen' status. A socket application should have a 'listener' up and be waiting for connections. If a particular socket is expected to be in 'Listen' status and is not, there may be a problem; this depends on the installation.

Another problem which could be found using this display would be if a connection hasn't been used for a long time but hasn't been terminated by the application. This can happen if the socket application isn't coded correctly. I've seen this type of behaviour with a tn3270 emulator to the mainframe – each time the user logged on, another connection would be created, but the old one not terminated. After a few hours, there were 60 or 70 connections, many of them with an idle time of a few hours.

SMF records

SMF records have been used for many years to provide information on many aspects of mainframe performance. For TCP/IP, the standard SMF records for OS/390 are the type 118, which provide data on the following: TCP initiation, TCP termination, ftp server, ftp client, and statistics.

z/OS saw the introduction of the type 119 SMF record, which contains more types of data and more data in each record. The data types are as follows: TCP initiation, TCP termination, ftp server, ftp client, statistics, server, and interface.

Two interesting types of data have been added with the type 119 record: round trip times and retransmissions for each TCP connection in the TCP termination record.

SNMP

SNMP is a diagnostic architecture which, despite its drawbacks, has become the *de facto* standard for network management. Most network hardware and software vendors implement SNMP Management Information Bases (MIBs) or diagnostic databases in their products. Products exist which provide generic access to SNMP MIBs. Often each vendor provides a product to access its own MIBs.

Errors in TCP/IP stack can often be found by using SNMP. The new z/OS MIB (1.2, 1.3, 1.4) has hundreds of variables. The following major sections are in the z/OS MIB:

- SNMP public MIBs
- z/OS extensions
- OSA/ATM
- OMPRoute
- SLA subagent.

Many of these SNMP variables are also available under the later releases of OS/390.

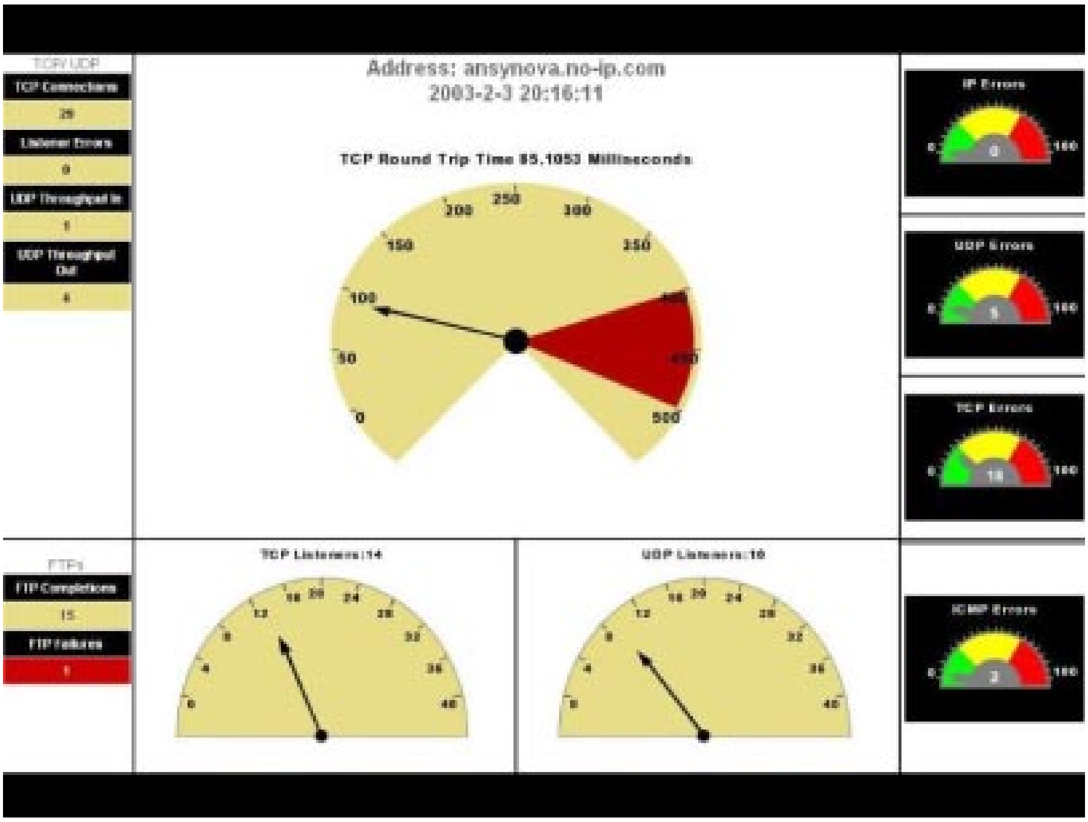


Figure 4: Performance dashboard

A full discussion of SNMP is beyond the scope of this article, but in the examples that follow we'll see how SNMP variables have been used in performance diagnostics.

Other tools

Other tools that can be used to diagnose TCP/IP performance problems are MVS or VTAM commands, packet traces, and TCP/IP monitors. Many TCP/IP monitors are available, each with a perspective on the important factors to monitor.

What follows is just one view of the important factors to monitor for TCP/IP. Many others exist, but at least once you've understood one perspective it gets easier to assess the others.

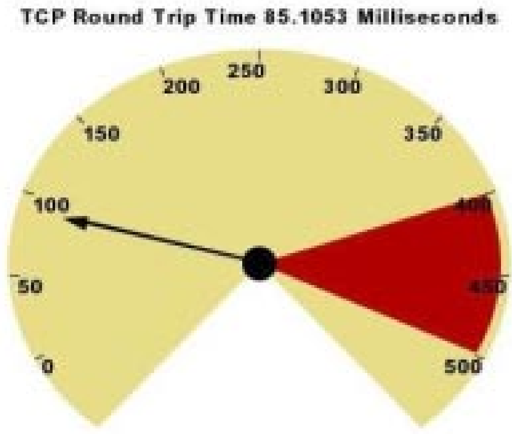


Figure 5: TCP round trip time

ONE VIEW OF IMPORTANT PARAMETERS TO MONITOR

To make performance monitoring more intuitive, let's use the concept of a performance dashboard showing the health of a particular TCP/IP stack. This enables many variables which contribute to problems with TCP/IP to be displayed on one screen, including round trip time, protocol errors, FTP failures, or listener problems (see Figure 4).

Let's look at each of the gauges and parameters shown in Figure 4, to see what they may indicate and why they've been chosen as important.

Round trip time

Many problems in the TCP network have only one symptom: bad response time. You may notice, however, that we're measuring 'round trip time', not response time (see Figure 5). So, is response time the same as round trip time? How is round trip time measured? Is it end-to-end time?

In the SNA world, we're used to thinking of round trip time as the time from pressing the Enter key to getting either the first or last character of the response back onto the screen. This number can be broken down into host time (VTAM time, application time) and network time (NCP, communications line, etc).

So is the same true for TCP/IP? Well, for some applications, the time from pressing the Enter key to getting either the first or last character of the response back onto your screen is a fine measurement. And, ideally, this number should also be broken down into host time (TCP/IP stack time, application time) and network time (routers, communications network, etc).

But what about ftp and UDP? We propose a practical approach to TCP/IP response time measurement, in order to get the time intervals for host, application, network, and so on. A connection must be time stamped at each of these points *and* there needs to be a way to get at the data – via an exit, log, or other monitoring data. Note that this can be expensive in terms of performance; the more instrumentation that's put into place, the longer the path length for any piece of code to execute. There is always a trade-off.

Some of this information is available through traces, but it must be correlated. The round trip time for each connection, however, can be obtained without running traces: it's available for z/OS systems only in the SMF 119 records. (If you need this type of information for

Name	***Port***	***Name***	***Port***
OSNMPD	1031	FTPD1	21
NFSS	2049	INETD4	1023
NFSS	1030	INETD4	513
NFSS	1029	TCPIP	23
NFSS	1028	TCPIP	1025
NFSS	1027	BPZOINIT	10007
PORTMAP	111		

Figure 6: TCP listeners which should be up

Figure 7: ftp monitoring

OS/390 systems, see the section on Listener Performance Profile.)

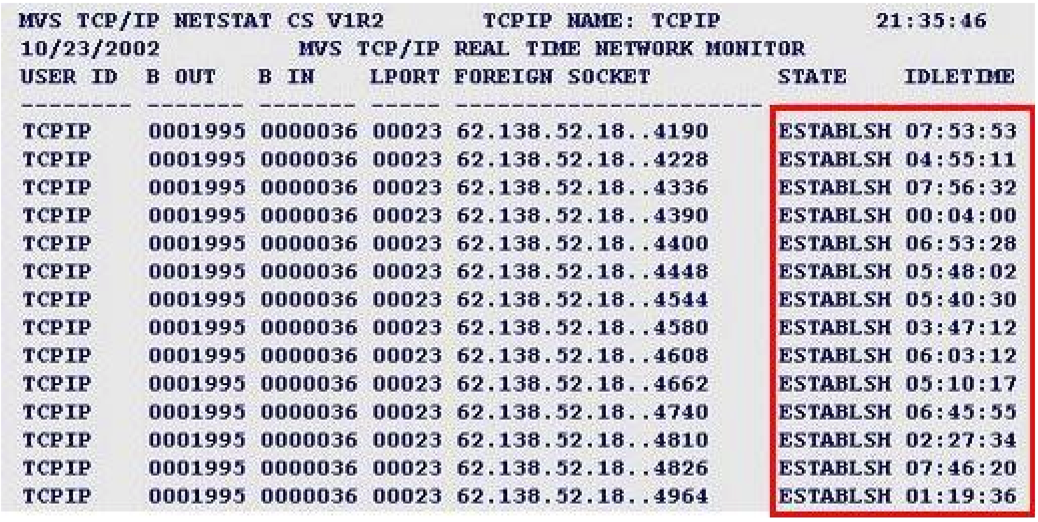
Round trip time (RTT) is basically network time. RTT measures from the time the last character is sent to when the ACK comes back. This is similar to a PING, except using the real data length used by the application and using TCP versus ICMP. The round trip time shown on the performance dashboard is the average last round trip time for all connections which completed within the past half hour.

RTT gives us some useful information. First, it at least allows us to rule out the network as a possible problem. Second, excessive round trip variance can point to underlying issues – perhaps duplicate acknowledgements and retransmissions, which can signal congestion on the network and poor network quality respectively.

For example, we saw a case where a DB2 application had thousands of duplicate acks and very erratic round trip variance. A trace route to the foreign address showed possible routing around a failing device.

Listener monitoring

Listeners or sockets are the applications on your TCP/IP system,



```
MVS TCP/IP NETSTAT CS V1R2       TCPIP NAME: TCPIP           21:35:46
10/23/2002          MVS TCP/IP REAL TIME NETWORK MONITOR
USER ID  B OUT   B IN    LPORT FOREIGN SOCKET          STATE    IDLETIME
-------- ------- ------- ----- ----------------------
TCPIP    0001995 0000036 00023 62.138.52.18..4190      ESTABLSH 07:53:53
TCPIP    0001995 0000036 00023 62.138.52.18..4228      ESTABLSH 04:55:11
TCPIP    0001995 0000036 00023 62.138.52.18..4336      ESTABLSH 07:56:32
TCPIP    0001995 0000036 00023 62.138.52.18..4390      ESTABLSH 00:04:00
TCPIP    0001995 0000036 00023 62.138.52.18..4400      ESTABLSH 06:53:28
TCPIP    0001995 0000036 00023 62.138.52.18..4448      ESTABLSH 05:48:02
TCPIP    0001995 0000036 00023 62.138.52.18..4544      ESTABLSH 05:40:30
TCPIP    0001995 0000036 00023 62.138.52.18..4580      ESTABLSH 03:47:12
TCPIP    0001995 0000036 00023 62.138.52.18..4608      ESTABLSH 06:03:12
TCPIP    0001995 0000036 00023 62.138.52.18..4662      ESTABLSH 05:10:17
TCPIP    0001995 0000036 00023 62.138.52.18..4740      ESTABLSH 06:45:55
TCPIP    0001995 0000036 00023 62.138.52.18..4810      ESTABLSH 02:27:34
TCPIP    0001995 0000036 00023 62.138.52.18..4826      ESTABLSH 07:46:20
TCPIP    0001995 0000036 00023 62.138.52.18..4964      ESTABLSH 01:19:36
```

Figure 8: Connections with high idle time

Figure 9: Listener errors

and monitoring the listeners or the TCP and UDP services is one of the most important parts of managing the TCP/IP network. You need to be sure that all your listeners are available, and that they are the right ones (see Figure 6). You need to know if a critical service such as ftp or telnet isn't available. And you should be alerted if a Listener drops.

ftp monitoring

ftp is an important part of monitoring TCP/IP. You should know how many ftps you're doing, and get an alert for any that are failing. The performance dashboard will indicate any problems and how many have completed (see Figure 7).

TCP connections

You should be able to see how many TCP connections exist on the TCP/IP network at any one time. As you watch, you'll get a feeling for what's normal for your system. For example, you may have thousands of connections in the morning and very few in the afternoon.

On a per-connection basis, you may want to look at the performance and error numbers as before (RTT, RTT variance, duplicate acks, retransmissions). Another important field to view is the Last Used time, which can be found in the Netstat Byteinfo Idletime command discussed earlier. You may be able to drop some connections which haven't been used for a long time, but make sure you check first that the connection isn't used by the system. Note that all Listener connections will appear to have a very high value in this field, and you don't want to drop those either (see Figure 8).

For telnet connections, you can use parameters for telnet server such as SCANINTERVAL or TIMEMARK to drop connections without activity.

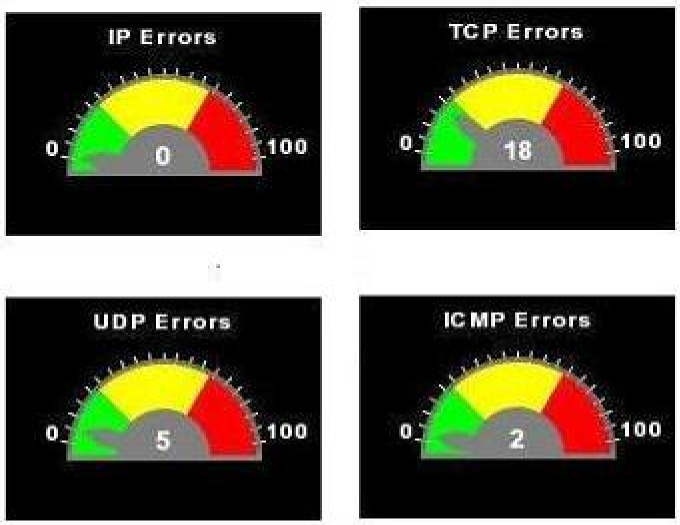


Figure 10: Errors with core Internet protocols

Errors/messages inbound – ICMP public MIB	
Messages in	*13,372*
Errors in	*0*
Destination unreachable in	*15*
Time exceeded in	*0*
Parameter problem in	*0*
Source quench in	*0*
Redirect in	*0*
Echo in	*13,357*
Echo reply in	*0*
Timestamp in	*0*
Timestamp reply in	*0*
Address mask in	*0*
Address mask reply in	*0*

Errors/messages outbound – ICMP public MIB	
Messages out	*13,372*
Errors out	*15*
Destination unreachable out	*15*
Time exceeded out	*0*
Parameter problem	*0*
Source quench out	*0*
Redirect out	*0*
Echo out	*0*
Echo reply out	*13,357*
Timestamp out	*0*
Timestamp reply out	*0*

Figure 11: ICMP MIB

Listener errors

The listener error indicator shows the backlog exceeded parameter for each TCP listener. Figure 9 is taken from the SNMP Listener MIB for z/OS. Other important fields include:

- *Current backlog*. The current number of connections in backlog.
- *Maximum in backlog*. The maximum number of connections allowed in backlog at one time.

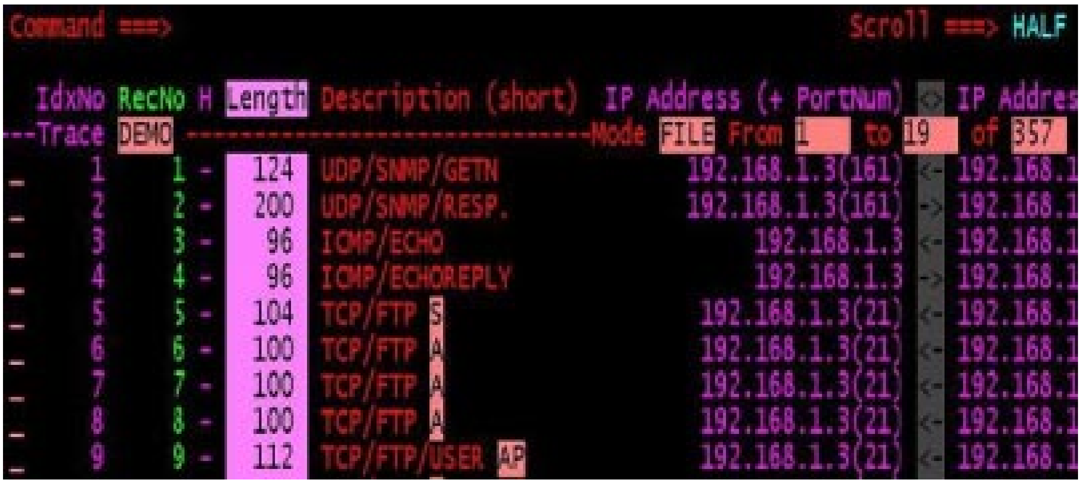


Figure 12: Sample packet trace

- *Exceed backlog.* The total number of connections dropped by the listener due to backlog exceeded.

You should monitor this field closely, as it can save you both time and expense. We've heard of sites escalating problems to IBM as an application hang when the problem turned out to be simply that the backlog queue was exceeded.

Errors with core Internet protocols

Once you progress to errors with core Internet protocols, you start to get into the real problems in the stack. SNMP is a very useful way to look at errors with the TCP/IP stack (see Figure 10). Remember that the function of the stack is to do TCP, UDP, and IP.

In the z/OS private MIB, you'll find new variables or extensions for IP, TCP, and UDP. Figure 11 is from the ICMP MIB, and shows ICMP errors for z/OS. The ICMP MIB is a public MIB, and is supported in any SNMP-capable device.

Note that the Echo In or Ping field is quite large; you'll need to do a packet trace to find out which address is doing all those PINGs to the mainframe. You sometimes need to use a combination of tools to diagnose TCP/IP problems.

Sample TCP/IP packet trace

Figure 12 shows a sample packet trace showing UDP, ICMP, and TCP activity. Note that lines 3 and 4 show the ICMP Echo and Echo Reply. Traces can be very large and it isn't always easy to find the offender. This time, we've captured the right information to help us diagnose the problem.

HOW TO FIX PROBLEMS

You sometimes need to change some of the many parameters in the TCP profile that control TCP, IP, UDP, and other behaviour. The parameters often aren't unique to the mainframe, but control TCP or IP behaviour for all TCP/IP implementations. The parameters are often defined in RFCs.

On the mainframe, the dataset which holds these definitions is the TCP profile.

Parameters for TCP

Some sample parameters which control the behaviour of TCP are shown below:

- Keep alive timer
- Send garbage enabled
- Send buffer size
- Receive buffer size
- Max receive buffer size
- Restrict low ports
- Maximum retransmit time
- Minimum retransmit time
- Round trip gain
- Variance gain
- Variance multiplier

- Timestamp.

Let's take a couple of these parameters and discuss them further.

Keep alive definition

A keep-alive packet is sent on idle TCP connections. If the remote TCP host fails to respond, the connection is terminated with an ETIMEDOUT status. The default value is 120 minutes and the range is from 0 to 35,791 minutes. A related keyword is SENDGARBAGE.

SendGarbage definition

SENDGARBAGE may be coded to indicate whether the keep-alive packet will contain 1 byte of random data. Some TCP/IP implementations can't handle a segment with no data, although they're supposed to do so. The keep-alive packet will also contain an invalid sequence number, and this will cause the receiving host to reject the packet.

Case study of problem caused by keep alive/send garbage

The following is a case study of a problem which may be caused by the send garbage parameter.

"After an MVS upgrade, we had problems with the communication from Sun Solaris Version 5 sender to the MVS receiver. The communication started, but we intermittently got error messages, and then the Solaris received a 'connection reset by peer' message.

"This said that there was an error receiving data and the connection timed out. We were only implementing fast channels and the keepalive option, and the problem seemed to occur when the channel wasn't used for a while.

"We looked at the keepalive option. We would see the problem about every 20 minutes, if there was no traffic, so we set keepalive to 20 minutes. When we ran a trace, we could see that MVS was sending a null packet and the Solaris didn't respond.

"We changed the SENDGARBAGE option to send garbage instead of a null packet and the Solaris responded – Solaris must be one of the platforms that can't handle a null packet."

CONCLUSION

Reams and reams could be written on how to find and diagnose TCP/IP problems – we've touched only the tip of the iceberg. We haven't even discussed network components such as OSA Express or Cisco CIP.

Remember that, to solve problems, you sometimes need to seek out historical trends over time. If a user says that he or she can't connect to an application, ask whether they've *ever* connected. Has *anyone* ever connected to the application?

If people complain about bad response time, ask is it just them? Is it everyone? Then, you need to measure both. These are basic rules for network diagnostics, but they are often forgotten.

Nalini Elkins
Inside Products (USA)

Telnet Stats

Many sites use PC-based 3270 terminal emulation (such as IBM's Personal Communications) instead of 'classic' terminals. Because such terminals usually get their ID dynamically, there's no way of knowing which PC user opened which terminal. Telnet Stats, presented here, enables you to associate a VTAM terminal identifier with an IP address.

Telnet Stats captures the output of the standard TCP/IP command – 'NETSTAT TELNET' – and then presents it using ISPF panels. It provides the following information:

- IP address along with port number

- VTAM application ID
- VTAM terminal ID
- Number of bytes sent and received
- Status of the connection.

The output can be sorted by any column except the status column.

To make it easier to filter the information (NETSTAT TELNET can produce an enormous amount of output if a lot of terminals are connected), I've prepared six different views:

- *IP information*, showing which sessions are opened from a given IP address.
- *Terminal information*, showing who (or, to be precise, which IP address) opened a given terminal ID.
- *TSO information*, showing all TSO sessions. This will work if the TSO application ID contains the string 'TSO'; if this isn't the case at your site, you can easily change it in the code by editing the 'TSO' member of the installation library.
- *CICS information*, showing all the sessions connected to a given application ID. Note that I've called it 'CICS info' simply because I use it to see who opened sessions for a given CICS region; you can use any application ID you like.
- *Unused sessions*, showing terminal emulation windows that are opened but not logged on to any application.
- *All sessions*.

All of these views are formatted to an ISPF table, so that they can be easily scrolled forwards and backwards.

All the code is written in REXX, which means that you can easily change it to fit your specific needs if you wish. I've put some comments into the code, but not too many!

INSTALLATION

To install Telnet Stats at your site, follow the procedure shown below:

1 Store the NETSTAT program, along with the subroutines, ISPF panel definitions, and ISPF messages in a sequential dataset (RECFM = FB, LRECL = 80) on your host machine in a library of your choice.

2 The library will contain the following members:

   – NETSTAT – the main program.

   – ALL, CICS, IP, NONE, TERMID, and TSO – subroutines called by the main program.

   – PNETSTAT, POUT, and PSETUP – ISPF panel definitions.

   – MNS00 – ISPF messages.

3 Edit the main program, NETSTAT, and change the variable 'CurLib' (line #3) to point to the library where you installed Telnet Stats. (This is important because I use LIBDEF to tell ISPF that my panels, messages, and tables can be found in the installation library.)

USE

To start Telnet Stats, execute the NETSTAT member from the installation library. You'll see the following panel:

```
Telnet Stats -------------------- Main Menu --------------------- by MG
Option ===>
_________________________________________________________________

   1  IP Info                  _______________    (Enter IP address)
   2  Terminal Info            ________           (Enter terminal ID)
   3  TSO Info
   4  CICS Info                ________           (Enter application name)
   5  Unused sessions
   6  All sessions

   Ø  Setup
```



```
#---------------------------------------------------------------#
| 1 - Sessions opened from a given IP address                    |
| 2 - Who opened given terminal ID                               |
| 3 - TSO sessions                                               |
| 4 - CICS or other appl. sessions                               |
| 5 - Unused opened sessions                                     |
| 6 - All active sessions                                        |
#---------------------------------------------------------------#
F1=Help    F2=Split   F3=Exit    F9=Swap   F12=Cancel
```

Choose one of the available options and press <ENTER>. If your option is 1 (IP Info), 2 (Terminal Info) or 4 (CICS Info), prior to pressing <ENTER> fill in appropriate fields.

As we saw above, the meanings of the various options are as follows:

1 Terminals opened from a given IP address.

2 Tells you which IP address opened the given terminal ID.

3 Terminals connected to TSO.

4 Terminals connected to a given CICS region (or any other application ID).

5 'Unused' terminals (terminals opened, but not logged on to any application).

6 All sessions.

Here is an example of possible output for option 6:

```
Telnet Stats ------------------ Output --------------- Row 1 to 9 of 77
Command ===>

Statistics:

  Sessions shown:       77       BytesIn  (all sessions shown):  3794674
  All sessions active:  77       BytesOut (all sessions shown): 31546854

All sessions:

IP                Port   ApplID    TermID    BytesIn   BytesOut  Status
-----------------------------------------------------------------------
1.1.1.253         45128  AØ6TSO17  SCØTCP2Ø  ØØØØ4441  ØØ135625  Establsh
1.1.1.253         49262  AØ6TSO19  SCØTCP27  ØØØØ7815  ØØ231Ø98  Establsh
```

```
1.1.2.113         4706   CICSS2    SCØTCPØ1  ØØØØØ575  ØØØ47617  Establsh
1.1.2.114         1626   -unused-  SCØTCP58  ØØØØ6262  ØØ175871  Establsh
1.1.2.115         1Ø96   CICSS2    SCØTCP73  ØØØØ181Ø  ØØ323587  Establsh
1.1.2.116         131Ø   -unused-  SCØTCP23  ØØØØØØ32  ØØØØ1842  Establsh
1.1.2.117         1Ø38   AØ6TSO1Ø  SCØTCP15  ØØØ79Ø7Ø  Ø1896Ø26  Establsh
1.1.2.125         1Ø48   CICSS2T   SCØTCP25  ØØØØØ8Ø4  ØØØ7Ø138  Establsh
1.1.2.125         1Ø91   CICSS2T   SCØTCP35  ØØØØ1Ø78  ØØØ56544  Establsh
 F1=Help  F2=Split  F3=Exit  F7=Backward  F8=Forward  F9=Swap F12=Cancel
```

There is an additional option (0) in the main panel, which allows you to set up sort order. The default is IP address (primary) and port (secondary). You can also sort by terminal ID, application ID, bytes sent, or bytes received. You can set up primary and secondary sort order, and the sort can be either ascending (default) or descending.

Note that everything is presented as strings, so that IP address 1.1.1.20 will be ahead of 1.1.1.3.

The set-up panel is shown below:

```
Telnet Stats -------------------- Setup -------------------------- by MG
Command ===>

 Primary Sort Column                   Primary Sort Order
 1  1.  IP                             1  1.  Ascending
    2.  Port                              2.  Descending
    3.  Application ID
    4.  Terminal ID
    5.  Bytes In
    6.  Bytes Out

 Secondary Sort Column                 Secondary Sort Order
 2  1.  IP                             1  1.  Ascending
    2.  Port                              2.  Descending
    3.  Application ID
    4.  Terminal ID
    5.  Bytes In
    6.  Bytes Out

Press ENTER to return to Netstat main panel


 F1=Help    F2=Split   F3=Exit    F9=Swap   F12=Cancel
```

SOME PROGRAMMING REMARKS

Telnet Stats captures the output of the standard TCP/IP command

'NETSTAT TELNET' and then presents it clearly using ISPF panels. The command actually used is 'NETSTAT TELNET REPORT'. The REPORT option makes NETSTAT write its output to a file. I could have used the REXX OUTTRAP facility to capture the output, but the lines of the output are longer that 80 bytes, so it was much easier to format it from a file. This means that every line can be read into a stem variable component and then easily parsed to make it more readable.

I use an ISPF table to present the formatted output. The table is written as an OUTTBL member in the installation library. However, if you prefer, you can write the output to a file and then view it.

REQUIREMENTS

Telnet Stats requires OS/390 with TCP/IP configured and ISPF. It was tested with OS/390 Version 2.8 and z/OS Version 1.1 with their appropriate communications servers, and ISPF Versions 4.8 and 5.0.

NETSTAT – THE MAIN PROGRAM

```
/* REXX NETSTAT                                    by Marcin Grabinski */

CurLib   = "'MARCIN.REXX.NETSTAT'"         /* Change the library!    */
IPAddr   = ''
CICSName = ''
TERMName = ''
PSortC = 'IP'
PSortO = 'A'
SSortC = 'Port'
SSortO = 'A'

SortOrd = PSortC' 'PSortO' 'SSortC' 'SSortO

User = SYSVAR(SYSUID)                      /* get the userid */
                                           /* will need it later     */
/* libdef the current library to ISPPLIB and ISPTLIB */

ADDRESS ISPEXEC
"LIBDEF ISPPLIB DATASET ID(&CurLib) UNCOND" /* panels */
"LIBDEF ISPTLIB DATASET ID(&CurLib) UNCOND" /* tables */
"LIBDEF ISPMLIB DATASET ID(&CurLib) UNCOND" /* messages */
```

```
/* display panel and wait for user action */

DO WHILE ZPFKEY /= 'PFØ3' & ZPFKEY /= 'PF12'
  zcmd = ''
  ADDRESS ISPEXEC
  "DISPLAY PANEL (PNETSTAT)"     /* Main menu                 */

  SELECT

    WHEN zcmd = 'Ø' THEN         /* Setup           */
    DO
      PARSE VALUE SortOrd WITH PSortC PSortO SSortC SSortO
      "DISPLAY PANEL(PSETUP)"
      SortOrd = PSortC' 'PSortO' 'SSortC' 'SSortO
      IF ZPFKEY = 'PFØ3' | ZPFKEY = 'PF12' THEN
        ZPFKEY = ''
    END

    WHEN zcmd = '1' THEN         /* IP info         */
    DO
      IF IPAddr <> '' THEN
        CALL IP User Curlib IPAddr SortOrd
      ELSE
        "DISPLAY MSG(MNSØØ1)"
    END
    WHEN zcmd = '2' THEN         /* Who's logged to a given CICS */
    DO
      IF TERMName <> '' THEN
        CALL TERMID User CurLib TERMName SortOrd
      ELSE
        "DISPLAY MSG(MNSØØ3)"
    END

    WHEN zcmd = '3' THEN         /* Who's logged in TSO */
        CALL TSO User Curlib SortOrd

    WHEN zcmd = '4' THEN         /* Who's logged to a given CICS */
    DO
      IF CICSName <> '' THEN
        CALL CICS User CurLib CICSName SortOrd
      ELSE
        "DISPLAY MSG(MNSØØ2)"
    END

    WHEN zcmd = '5' THEN         /* Who has unused sessions */
        CALL NONE User Curlib SortOrd

    WHEN zcmd = '6' THEN         /* All sessions            */
        CALL ALL User Curlib SortOrd

    WHEN zcmd = '' THEN
```

```
    DO
      IF ZPFKEY = 'PFØ3' | ZPFKEY = 'PF12' THEN
        LEAVE
      ELSE
        "DISPLAY MSG(MNSØØ4)"
    END

    OTHERWISE
      "DISPLAY MSG(MNSØØØ)"

  END /* SELECT */

END   /* main loop */

RETURN
```

SUBROUTINES

ALL

```
/* REXX                                                              */

PARSE ARG User Curlib SortOrd
SNum = Ø
ByteInS  = Ø
ByteOutS = Ø
SubTitle = 'All sessions:'

PARSE VALUE SortOrd WITH PSortC PSortO SSortC SSortO

ADDRESS TSO
cmd = '"ALLOC F(TLIB) DA('CurLib') SHR REUSE"'
INTERPRET cmd

/* call NETSTAT to get needed info */

cmd = '"PROFILE PREFIX('User')"'             /* set the user's prefix */
INTERPRET cmd
"NETSTAT TELNET REPORT"

/* Netstat returns data in 'userid.NETSTAT.TELNET' */

cmd = '"ALLOC F(FileIn) DA('"'"User'.NETSTAT.TELNET'"'"') SHR"'
INTERPRET cmd
"EXECIO * DISKR FileIn (OPEN STEM Input. FINIS"
"FREE F(FileIn)"

/* Prepare an ISPF table */

ADDRESS ISPEXEC
```

```
"TBCREATE OUTTBL NAMES(IP Port Appl ID BytesIn BytesOut State) WRITE
REPLACE"

ASNum = Input.Ø - 4                 /* number of all active sessions  */

DO I = 1 TO Input.Ø
  PARSE VALUE Input.i WITH Mssg Conn IP State BytesIn BytesOut Appl ID
Rest
  IF Mssg = 'EZZ28Ø3I' THEN      /* other records are not of interest */
  DO
    IPPort = LEFT(IP, 22)        /* IP address (along with port)      */
    IF POS('..', IPPort) <> Ø THEN
    DO
      PortPos = POS('..', IPPort) - 1
      IP = SUBSTR(IPPort, 1, PortPos)
      PortPos = PortPos + 3
      Port = SUBSTR(IPPort, PortPos, 5)
      IP = LEFT(IP, 15)
    END
    ID = LEFT(ID, 8)             /* TermID                            */
    Appl = LEFT(Appl, 8)         /* Application name                  */
    State = LEFT(State, 8)       /* Status of the session             */
    BytesIn = LEFT(BytesIn, 8)   /* Bytes got                         */
    BytesOut = LEFT(BytesOut, 8) /* Bytes sent                        */
    IF ID = '' THEN
    DO
      ID = Appl                 /* when ID is blank, it has been moved*/
      Appl = '-unused-'         /* to Appl| */
    END
    SNum = Snum + 1
    ByteInS = ByteInS + BytesIn
    ByteOutS = ByteOutS + BytesOut
    "TBADD OUTTBL"
  END /* IF Mssg = 'EZZ28Ø3I' */
END /* main loop */

"TBCLOSE OUTTBL REPLCOPY LIBRARY(TLIB)"

"TBOPEN OUTTBL"

cmd = '"TBSORT OUTTBL FIELDS('PSortC',C,'PSortO','SSortC',C,'SSortO')"'
INTERPRET cmd
"TBDISPL OUTTBL PANEL(POUT)"

"TBCLOSE OUTTBL REPLCOPY LIBRARY(TLIB)"

ADDRESS TSO
"FREE F(TLIB)"

RETURN
```

CICS

```
/* REXX                                                by Marcin Grabinski */

PARSE ARG User CurLib CICSName SortOrd
SNum = Ø
ByteInS  = Ø
ByteOutS = Ø
SubTitle = 'Sessions opened for 'CICSName':'

PARSE VALUE SortOrd WITH PSortC PSortO SSortC SSortO

ADDRESS TSO
cmd = '"ALLOC F(TLIB) DA('CurLib') SHR REUSE"'
INTERPRET cmd

/* call NETSTAT to get needed info */

cmd = '"PROFILE PREFIX('User')"'             /* set the user's prefix */
INTERPRET cmd
"NETSTAT TELNET REPORT"

/* Netstat returns data in 'userid.NETSTAT.TELNET' */

cmd = '"ALLOC F(FileIn) DA('"'"User'.NETSTAT.TELNET'"'"') SHR"'
INTERPRET cmd
"EXECIO * DISKR FileIn (OPEN STEM Input. FINIS"
"FREE F(FileIn)"

/* Prepare an ISPF table */

ADDRESS ISPEXEC
"TBCREATE OUTTBL NAMES(IP Port Appl ID BytesIn BytesOut State) WRITE
REPLACE"

ASNum = Input.Ø - 4                  /* number of all active sessions  */

DO i = 1 TO Input.Ø

  PARSE VALUE Input.i WITH Mssg Conn IP State BytesIn BytesOut Appl ID
Rest
  IF Mssg = 'EZZ28Ø3I' THEN      /* other records are not of interest */
  DO
    IPPort = LEFT(IP, 22)        /* IP address (along with port)      */
    IF POS('..', IPPort) <> Ø THEN
    DO
      PortPos = POS('..', IPPort) - 1
      IP = SUBSTR(IPPort, 1, PortPos)
      PortPos = PortPos + 3
      Port = SUBSTR(IPPort, PortPos, 5)
      IP = LEFT(IP, 15)
```

```
    END
    Appl = LEFT(Appl, 8)         /* Application name                */
    State = LEFT(State, 8)       /* Status of the session           */
    ID = SUBSTR(ID, 5, 4)        /* CICS TermID is 4 bytes long     */
    ID = LEFT(ID, 8)
    BytesIn = LEFT(BytesIn, 8)   /* Bytes got                       */
    BytesOut = LEFT(BytesOut, 8) /* Bytes sent                      */

    IF Appl = CICSName THEN    /* Show only given CICS sessions     */
    DO
      SNum = Snum + 1
      ByteInS = ByteInS + BytesIn
      ByteOutS = ByteOutS + BytesOut
      "TBADD OUTTBL"
    END /* IF CICSName = Appl THEN*/
  END
END /* DO I = 1 TO Input.Ø*/

"TBCLOSE OUTTBL REPLCOPY LIBRARY(TLIB)"

"TBOPEN OUTTBL"
cmd = '"TBSORT OUTTBL FIELDS('PSortC',C,'PSortO','SSortC',C,'SSortO')"'
INTERPRET cmd
"TBDISPL OUTTBL PANEL(POUT)"

"TBCLOSE OUTTBL REPLCOPY LIBRARY(TLIB)"

ADDRESS TSO
"FREE F(TLIB)"

RETURN
```

IP

```
/* REXX                                                             */

PARSE ARG User Curlib IPAddr SortOrd
SNum = Ø
ByteInS  = Ø
ByteOutS = Ø
SubTitle = 'Sessions opened from IP address 'IPAddr':'

PARSE VALUE SortOrd WITH PSortC PSortO SSortC SSortO

ADDRESS TSO
cmd = '"ALLOC F(TLIB) DA('CurLib') SHR REUSE"'
INTERPRET cmd

/* call NETSTAT to get needed info */

cmd = '"PROFILE PREFIX('User')"'             /* set the user's prefix */
```

```
INTERPRET cmd
"NETSTAT TELNET REPORT"

/* Netstat returns data in 'userid.NETSTAT.TELNET' */

cmd = '"ALLOC F(FileIn) DA('"'"User'.NETSTAT.TELNET'"'"') SHR"'
INTERPRET cmd
"EXECIO * DISKR FileIn (OPEN STEM Input. FINIS"
"FREE F(FileIn)"

/* Prepare an ISPF table */

ADDRESS ISPEXEC
"TBCREATE OUTTBL NAMES(IP Port Appl ID BytesIn BytesOut State) WRITE
REPLACE"

ASNum = Ø /* number of all sessions            */

DO i = 1 TO Input.Ø
  PARSE VALUE Input.i WITH Mssg Conn IP State BytesIn BytesOut Appl ID
Rest
  IF Mssg = 'EZZ28Ø3I' THEN      /* other records are not of interest */
  DO
    ASNum = ASNum + 1
    IPPort = LEFT(IP, 22)        /* IP address (along with port)      */

    IF ID = '' THEN              /* when ID is blank, it has been moved
                                    to Appl| */
    DO
      ID = Appl
      Appl = '-unused-'
    END /* IF ID = '' THEN */

    IF POS('..', IPPort) <> Ø THEN
    DO
      PortPos = POS('..', IPPort) - 1
      IP = SUBSTR(IPPort, 1, PortPos)
      PortPos = PortPos + 3
      Port = SUBSTR(IPPort, PortPos, 5)
      IP = LEFT(IP, 15)
    END
    Appl = LEFT(Appl, 8)         /* Application name                  */
    State = LEFT(State, 8)       /* Status of the session             */
    ID = LEFT(ID, 8)             /* TermID                            */
    BytesIn = LEFT(BytesIn, 8)   /* Bytes got                         */
    BytesOut = LEFT(BytesOut, 8) /* Bytes sent                        */

    IF IPAddr = IP THEN /* Filter IP */
    DO
      SNum = Snum + 1
```

```
      ByteInS = ByteInS + BytesIn
      ByteOutS = ByteOutS + BytesOut
      "TBADD OUTTBL"
    END /* IF POS(IPAddr, IP) > Ø & ID <> '' */
  END
END /* DO i = 1 TO Input.Ø*/

"TBCLOSE OUTTBL REPLCOPY LIBRARY(TLIB)"

"TBOPEN OUTTBL"
cmd = '"TBSORT OUTTBL FIELDS('PSortC',C,'PSortO','SSortC',C,'SSortO')"'
INTERPRET cmd
"TBDISPL OUTTBL PANEL(POUT)"

"TBCLOSE OUTTBL REPLCOPY LIBRARY(TLIB)"

ADDRESS TSO
"FREE F(TLIB)"

RETURN
```

NONE

```
/* REXX                                                                   */

PARSE ARG User Curlib SortOrd
SNum = Ø
ByteInS  = Ø
ByteOutS = Ø
SubTitle = 'Unsued sessions:'

PARSE VALUE SortOrd WITH PSortC PSortO SSortC SSortO

ADDRESS TSO
cmd = '"ALLOC F(TLIB) DA('CurLib') SHR REUSE"'
INTERPRET cmd

/* call NETSTAT to get needed info */

cmd = '"PROFILE PREFIX('User')"'             /* set the user's prefix */
INTERPRET cmd
"NETSTAT TELNET REPORT"

/* Netstat returns data in 'userid.NETSTAT.TELNET' */

cmd = '"ALLOC F(FileIn) DA('"'"User'.NETSTAT.TELNET'"'"') SHR"'
INTERPRET cmd
"EXECIO * DISKR FileIn (OPEN STEM Input. FINIS"
"FREE F(FileIn)"
```

```
/* Prepare an ISPF table */

ADDRESS ISPEXEC
"TBCREATE OUTTBL NAMES(IP Port Appl ID BytesIn BytesOut State) WRITE
REPLACE"

ASNum = Ø                           /* number of all active sessions  */

DO I = 1 TO Input.Ø
  PARSE VALUE Input.i WITH Mssg Conn IP State BytesIn BytesOut Appl ID
Rest
  IF Mssg = 'EZZ28Ø3I' THEN      /* other records are not of interest */
  DO
    ASNum = ASNum + 1
    IPPort = LEFT(IP, 22)        /* IP address (along with port)      */
    IF POS('..', IPPort) <> Ø THEN
    DO
      PortPos = POS('..', IPPort) - 1
      IP = SUBSTR(IPPort, 1, PortPos)
      PortPos = PortPos + 3
      Port = SUBSTR(IPPort, PortPos, 5)
      IP = LEFT(IP, 15)
    END
    Appl = LEFT(Appl, 8)         /* Application name                  */
    State = LEFT(State, 8)       /* Status of the session             */
    ID = LEFT(ID, 8)             /* TermID                            */
    BytesIn = LEFT(BytesIn, 8)   /* Bytes got                         */
    BytesOut = LEFT(BytesOut, 8) /* Bytes sent                        */
    IF ID = '' & IP <> 'Telnet' THEN
    DO
      ID = Appl                 /* when ID is blank, it has been moved*/
      Appl = '-unused-'         /* to Appl| */
      SNum = Snum + 1
      ByteInS = ByteInS + BytesIn
      ByteOutS = ByteOutS + BytesOut
      "TBADD OUTTBL"
    END
  END
END

"TBCLOSE OUTTBL REPLCOPY LIBRARY(TLIB)"

"TBOPEN OUTTBL"
cmd = '"TBSORT OUTTBL FIELDS('PSortC',C,'PSortO','SSortC',C,'SSortO')"'
INTERPRET cmd
"TBDISPL OUTTBL PANEL(POUT)"

"TBCLOSE OUTTBL REPLCOPY LIBRARY(TLIB)"
```



```
ADDRESS TSO
"FREE F(TLIB)"

RETURN
```

TERMID

```
/* REXX                                              by Marcin Grabinski */

PARSE ARG User CurLib TermID SortOrd
SNum = Ø
ByteInS  = Ø
ByteOutS = Ø
SubTitle = TermID' sesion:'

PARSE VALUE SortOrd WITH PSortC PSortO SSortC SSortO

ADDRESS TSO
cmd = '"ALLOC F(TLIB) DA('CurLib') SHR REUSE"'
INTERPRET cmd

/* call NETSTAT to get needed info */

cmd = '"PROFILE PREFIX('User')"'             /* set the user's prefix */
INTERPRET cmd
"NETSTAT TELNET REPORT"

/* Netstat returns data in 'userid.NETSTAT.TELNET' */

cmd = '"ALLOC F(FileIn) DA('"'"'User'.NETSTAT.TELNET'"'"') SHR"'
INTERPRET cmd
"EXECIO * DISKR FileIn (OPEN STEM Input. FINIS"
"FREE F(FileIn)"

/* Prepare an ISPF table */

ADDRESS ISPEXEC
"TBCREATE OUTTBL NAMES(IP Port Appl ID BytesIn BytesOut State) WRITE
REPLACE"

ASNum = Input.Ø - 4                  /* number of all active sessions   */

DO i = 1 TO Input.Ø

  PARSE VALUE Input.i WITH Mssg Conn IP State BytesIn BytesOut Appl ID
Rest
  IF Mssg = 'EZZ28Ø3I' THEN      /* other records are not of interest */
  DO
    IPPort = LEFT(IP, 22)        /* IP address (along with port)      */
```

```
    IF POS('..', IPPort) <> Ø THEN
    DO
      PortPos = POS('..', IPPort) - 1
      IP = SUBSTR(IPPort, 1, PortPos)
      PortPos = PortPos + 3
      Port = SUBSTR(IPPort, PortPos, 5)
      IP = LEFT(IP, 15)
    END
    Appl = LEFT(Appl, 8)         /* Application name                  */
    State = LEFT(State, 8)       /* Status of the session             */
    ID = LEFT(ID, 8)
    IF LENGTH(TermID) = 4 THEN
      ID = SUBSTR(ID, 5, 4)      /* CICS TermID is 4 bytes long       */
    BytesIn = LEFT(BytesIn, 8)   /* Bytes got                         */
    BytesOut = LEFT(BytesOut, 8) /* Bytes sent                        */

    IF ID = TermID THEN        /* Show given terminal ID              */
    DO
      SNum = Snum + 1
      ByteInS = ByteInS + BytesIn
      ByteOutS = ByteOutS + BytesOut
      "TBADD OUTTBL"
    END /* ID = TermID THEN*/
  END
END /* DO I = 1 TO Input.Ø*/

"TBCLOSE OUTTBL REPLCOPY LIBRARY(TLIB)"

"TBOPEN OUTTBL"
cmd = '"TBSORT OUTTBL FIELDS('PSortC',C,'PSortO','SSortC',C,'SSortO')"'
INTERPRET cmd
"TBDISPL OUTTBL PANEL(POUT)"

"TBCLOSE OUTTBL REPLCOPY LIBRARY(TLIB)"

ADDRESS TSO
"FREE F(TLIB)"

RETURN
```

TSO

```
/* REXX                                                               */

PARSE ARG User Curlib SortOrd
SNum = Ø
ByteInS  = Ø
ByteOutS = Ø
SubTitle = 'Sessions opened for TSO:'
```

```
PARSE VALUE SortOrd WITH PSortC PSortO SSortC SSortO

ADDRESS TSO
cmd = '"ALLOC F(TLIB) DA('CurLib') SHR REUSE"'
INTERPRET cmd

/* call NETSTAT to get needed info */

cmd = '"PROFILE PREFIX('User')"'            /* set the user's prefix */
INTERPRET cmd
"NETSTAT TELNET REPORT"

/* Netstat returns data in 'userid.NETSTAT.TELNET' */

cmd = '"ALLOC F(FileIn) DA('"'"User'.NETSTAT.TELNET'"'"') SHR"'
INTERPRET cmd
"EXECIO * DISKR FileIn (OPEN STEM Input. FINIS"
"FREE F(FileIn)"

/* Prepare an ISPF table */

ADDRESS ISPEXEC
"TBCREATE OUTTBL NAMES(IP Port Appl ID BytesIn BytesOut State) WRITE
REPLACE"

ASNum = Ø                          /* number of all active sessions  */

DO I = 1 TO Input.Ø
  PARSE VALUE Input.i WITH Mssg Conn IP State BytesIn BytesOut Appl ID
Rest
  IF Mssg = 'EZZ28Ø3I' THEN      /* other records are not of interest */
  DO
    ASNum = ASNum + 1
    IPPort = LEFT(IP, 22)        /* IP address (along with port)      */
    IF POS('..', IPPort) <> Ø THEN
    DO
      PortPos = POS('..', IPPort) - 1
      IP = SUBSTR(IPPort, 1, PortPos)
      PortPos = PortPos + 3
      Port = SUBSTR(IPPort, PortPos, 5)
      IP = LEFT(IP, 15)
    END
    Appl = LEFT(Appl, 8)         /* Application name                  */
    State = LEFT(State, 8)       /* Status of the session             */
    ID = LEFT(ID, 8)             /* TermID                            */
    BytesIn = LEFT(BytesIn, 8)   /* Bytes got                         */
    BytesOut = LEFT(BytesOut, 8) /* Bytes sent                        */
    IF POS('TSO', Appl) > Ø THEN
    DO
      SNum = Snum + 1
```

```
      ByteInS = ByteInS + BytesIn
      ByteOutS = ByteOutS + BytesOut
      "TBADD OUTTBL"
    END
  END
END

"TBCLOSE OUTTBL REPLCOPY LIBRARY(TLIB)"

"TBOPEN OUTTBL"
cmd = '"TBSORT OUTTBL FIELDS('PSortC',C,'PSortO','SSortC',C,'SSortO')"'
INTERPRET cmd
"TBDISPL OUTTBL PANEL(POUT)"

"TBCLOSE OUTTBL REPLCOPY LIBRARY(TLIB)"

ADDRESS TSO
"FREE F(TLIB)"

RETURN
```

ISPF PANEL DEFINITIONS

PNETSTAT

```
)PANEL
)ATTR
   % TYPE(TEXT) INTENS(HIGH)
   ¬ TYPE(TEXT) INTENS(LOW)
   _ TYPE(INPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT) PAD(_)
)BODY
%Telnet Stats ------------------ Main Menu ----------------------- by MG
%Option ===>_ZCMD
¬
¬
¬  %1¬ IP Info¬              _IPADDR        ¬ (Enter IP address)
¬  %2¬ Terminal Info¬        _TERMNAME¬       (Enter terminal ID)
¬  %3¬ TSO Info¬
¬  %4¬ CICS Info¬            _CICSNAME¬       (Enter application name)
¬  %5¬ Unused sessions
¬  %6¬ All sessions
¬
¬  %Ø¬ Setup
¬
¬
¬
¬       #----------------------------------------------------------------#
¬       | 1 - Sessions opened from a given IP address                    |
```

```
¬        | 2 - Who opened given terminal ID                         |
¬        | 3 - TSO sessions                                         |
¬        | 4 - CICS or other appl. sessions                         |
¬        | 5 - Unused opened sessions                               |
¬        | 6 - All active sessions                                  |
¬        #----------------------------------------------------------#
¬
)INIT
  .CURSOR = ZCMD
)PROC
  &ZPFKEY = .PFKEY
)END
```

POUT

```
)PANEL KEYLIST(ISRSNAB)
)ATTR DEFAULT(%+_)
      /*  % TYPE(TEXT) INTENS(HIGH)      defaults displayed for      */
      /*  + TYPE(TEXT) INTENS(LOW)       information only            */
      /*  _ TYPE(INPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT)             */
$ type(output) intens(low)
# type(output) intens(high) caps(off)
)body expand(//)
%Telnet Stats --------------------- Output --------------------- by MG
%Command ===>_ZCMD                                                 +
+
%Statistics:
+
%  Sessions shown:      $SNum   % BytesIn  (all sessions shown):
$ByteInS
%  All sessions active: $ASNum  % BytesOut (all sessions shown):
$ByteOutS
+
#SubTitle
+
+IP               Port   ApplID    TermID    BytesIn   BytesOut  Status
+-----------------------------------------------------------------------
)MODEL
$IP              $PORT  $Appl     $ID       $BytesIn  $BytesOut $State
MBRNAME
)INIT
&ZCMD = ''
)PROC
&ZCMD = TRUNC(&ZCMD,' ')
&PARM = .TRAIL
)END
```

PSETUP

```
)PANEL
```

```
)ATTR DEFAULT(___) FORMAT(MIX)
 ØA TYPE(TEXT) INTENS(HIGH)
 Ø9 TYPE(TEXT) INTENS(LOW)
 ØC TYPE(NT) SKIP(ON)
 11 TYPE(SAC)
 12 TYPE(CEF)
 13 TYPE(INPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT)
)BODY  CMD(ZCMD)
éTelnet Stats ------------------- Setup ------------------------- by MGé
éCommand ===>_Z
é
é
ç Primary Sort Column                ç Primary Sort Order
 _Z

_1. _IP                            _Z

_1. _Ascending                   é
    _2. _Port                            _2. _Descending            é
    _3. _Application ID                                             é
    _4. _Terminal ID                                                é
    _5. _Bytes In                    éé
    _6. _Bytes
é
ç Secondary Sort Column              ç Secondary Sort Order
 _Z

_1. _IP                            _Z

_1. _Ascending                   é
    _2. _Port                            _2. _Descending            é
    _3. _Application ID                                             é
    _4. _Terminal ID                                                é
    _5. _Bytes In                    éé
    _6. _Bytes Out
é
çPress ENTER to return to Netstat main panel
)INIT
.ZVARS = '(ZCMD PSC PSO SSC SSO)'
&ZCMD = ' '
&PSC = '1'
&PSO = '1'
&SSC = '2'
&SSO = '1'
IF (&PSortC='IP') &PSC='1'
IF (&PSortC='Port') &PSC='2'
IF (&PSortC='Appl') &PSC='3'
IF (&PSortC='ID') &PSC='4'
IF (&PSortC='BytesIn') &PSC='5'
IF (&PSortC='BytesOut') &PSC='6'
IF (&SSortC='IP') &SSC='1'
```



```
IF (&SSortC='Port') &SSC='2'
IF (&SSortC='Appl') &SSC='3'
IF (&SSortC='ID') &SSC='4'
IF (&SSortC='BytesIn') &SSC='5'
IF (&SSortC='BytesOut') &SSC='6'
IF (&PSortO='A') &PSO='1'
IF (&PSortO='D') &PSO='2'
IF (&SSortO='A') &PSO='1'
IF (&SSortO='D') &PSO='2'
)PROC
 &ZPFKEY = .PFKEY
VER(&PSC RANGE,1,6)
VER(&PSO RANGE,1,2)
VER(&SSC RANGE,1,6)
VER(&SSO RANGE,1,2)
IF (&PSC='1') &PSortC='IP'
IF (&PSC='2') &PSortC='Port'
IF (&PSC='3') &PSortC='Appl'
IF (&PSC='4') &PSortC='ID'
IF (&PSC='5') &PSortC='BytesIn'
IF (&PSC='6') &PSortC='BytesOut'
IF (&SSC='1') &SSortC='IP'
IF (&SSC='2') &SSortC='Port'
IF (&SSC='3') &SSortC='Appl'
IF (&SSC='4') &SSortC='ID'
IF (&SSC='5') &SSortC='BytesIn'
IF (&SSC='6') &SSortC='BytesOut'
IF (&PSO='1') &PSortO='A'
IF (&PSO='2') &PSortO='D'
IF (&SSO='1') &SSortO='A'
IF (&SSO='2') &SSortO='D'
)END
```

ISPF MESSAGES

MNS00

```
MNSØØØ .TYPE=WARNING
'Option not valid'
MNSØØ1 .TYPE=WARNING
'Supply IP addres'
MNSØØ2 .TYPE=WARNING
'Supply CICS name'
MNSØØ3 .TYPE=WARNING
'Supply terminal ID'
MNSØØ4 .TYPE=WARNING
'Enter option'
```

Marcin Grabinski
SPIN, Poland (www.spinet.com.pl)

FTPSMFEX and FTPOSTPR exits

This article shows a possible use for each of two ftp exits – FTPSMFEX and FTPOSTPR (available only from OS/390 2.10 onwards).

The first can be used to view the SMF record for ftp (118) before it's written, enabling you to modify it or take other actions.

The second is called at the end of the ftp process and enables you to take action on the return code base.

We decided to code these exits to alert our operators to the fact that an incoming ftp had failed. If the failure is due to cancelling the ftp, the server, or the client, no interception will take place; however, all other failures (an x37 error condition, for example) are signalled.

The FTPSMFEX exit provides useful information such as userid, client ip address, ftp return code, and dataset name transferred.

The FTPOSTPR exit, whose sample is in C language, is useful for coding post-processing actions after file transfer completion, as a job is submitted. However, we can't use it this way, as we organized the ftp post-processing on the transferred-datasetname base, and this exit has no information about the dataset name. We decided to implement it all the same, because it provides a complete error message when ftp fails, and this kind of information is not available with the FTPSMFEX exit.

The scheme is as follows: when an ftp fails, FTPSMFEX submits two jobs via intrdr. The first contains the information needed to identify the error, and the second is a CA-7 demand of a job that ends with the error (so that our operators see the anomaly).

The FTPOSTPR exit then submits a third job via intrdr, with the complete ftp error message.

The source of the exits is given below.

FTPSMFEX

```
//COØ76ASM JOB (1ØØØ,AØØØ),CLASS=J,MSGCLASS=9
//*----------------------------------------------------*
//*                                                    *
//*   ASSEMBLY AND CATALOGING OF MPF EXITS             *
//*                                                    *
//*----------------------------------------------------*
//STEPØØ2Ø EXEC PGM=ASMA9Ø,PARM=('DECK,NOOBJECT,LIST')
//SYSLIB    DD   DSN=SYS1.MACLIB,DISP=SHR
//          DD   DSN=SYS1.MODGEN,DISP=SHR
//          DD   DSN=TCPIP.SEZACMAC,DISP=SHR
//SYSUT1    DD   DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(17ØØ,(9ØØ,1ØØ)),
//              SEP=(SYSLIB)
//SYSUT2    DD   DSN=&&SYSUT2,UNIT=SYSDA,SPACE=(17ØØ,(6ØØ,5Ø)),
//              SEP=(SYSLIB,SYSUT1)
//SYSUT3    DD   DSN=&&SYSUT3,UNIT=SYSDA,SPACE=(17ØØ,(6ØØ,5Ø))
//SYSPRINT DD   SYSOUT=*,DCB=BLKSIZE=1Ø89
//SYSPUNCH  DD  DSN=&&OBJSET,UNIT=SYSDA,SPACE=(8Ø,(2ØØ,5Ø)),
//              DISP=(MOD,PASS)
//SYSIN     DD   *
FTPSMFEX CSECT
FTPSMFEX AMODE 31
FTPSMFEX RMODE ANY
         SAVE  (14,12)
         BALR  12,Ø
         USING *,12
         B     BEGIN
         DC    C'FTPSMFEX '
         DC    C' &SYSDATE '
         DC    C' &SYSTIME '
         DS    ØF
*
BEGIN    LR    R2,R1                   *PARM POINTER
         USING PARMS,R2
         L     R9,PTRSMFR              *-> SMF RECORD
         USING SMFFTP76,R9
         CLC   SMFFTSLR,=C'25Ø'        *FTP ENDED NORMALLY ?
         BE    WRITESMF                *YES QUIT
FTPKO    EQU   *
         GETMAIN R,LV=WORKLEN          *GET AREA FOR DYNALLOC
         LTR   R15,R15
         BZ    ALLIR
         WTO   'FTPSMFEX: ERROR DURING GETMAIN'
         B     WRITESMF
* PREPARES THE ALLOCATION OF INTERNAL READER VIA DYNALLOC
ALLIR    EQU    *
         ST    R13,4(,R1)               ADDRESS OF PREVIOUS SA
```

```
         ST    R1,8(,R13)                ADDRESS OF NEXT SA
         LR    R13,R1                    OUR SAVEAREA ADDRESS
         USING WORKAREA,R13              ADDRESSABILITY
**********************************************************************
* SUBMIT VIA INTERNAL READER
**********************************************************************
SUBJOB   EQU   *                      BUILD REQUEST BLOCK
         LA    R6,SVC99STR
         USING S99RBP,R6              R6 -> START WORKAREA (S99RBPTR)
         LA    R3,S99RBPTR+4
         USING S99RB,R3               R3 -> RB (S99RB)
         ST    R3,S99RBPTR            PUT IN WORKAREA RB ADDRESS
         OI    S99RBPTR,S99RBPND      FIRST BIT ON (END POINTER)
         XC    S99RB(RBLEN),S99RB     ZEROES  RB
         MVI   S99RBLN,RBLEN          LOAD LENGTH IN RB
         MVI   S99VERB,S99VRBAL       SET VERB CODE TO ALLOCATE
         LA    R4,S99RB+RBLEN
         USING S99TUPL,R4             R4 -> END RB START POINTER LIST TU
         ST    R4,S99TXTPP            LOAD IN RB ADDRESS POINTER LIST TU
         LA    R1,SYSOUTAL            ALLOCATE SYSOUT
         ST    R1,S99TUPTR            AND STORE IN TUP
         LA    R4,S99TUPL+L'S99TUPTR NEXT TUP ENTRY
         LA    R1,SYSOUTIR            ALLOCATE INTERNAL READER
         ST    R1,S99TUPTR            AND STORE IN TUP
         LA    R4,S99TUPL+L'S99TUPTR NEXT TUP ENTRY
         LA    R1,SYSOUTBK            ALLOCATE BLKSIZE
         ST    R1,S99TUPTR            AND STORE IN TUP
         LA    R4,S99TUPL+L'S99TUPTR NEXT TUP ENTRY
         LA    R1,SYSOUTRL            ALLOCATE LRECL
         ST    R1,S99TUPTR            AND STORE IN TUP
         LA    R4,S99TUPL+L'S99TUPTR NEXT TUP ENTRY
         LA    R1,SYSOUTRF            ALLOCATE RECFM
         ST    R1,S99TUPTR            AND STORE IN TUP
         LA    R4,S99TUPL+L'S99TUPTR NEXT TUP ENTRY
         LA    R1,DEALLOC             DEALLOCATE AT CLOSE
         ST    R1,S99TUPTR            AND STORE IN TUP
         LA    R5,S99TUPL+L'S99TUPTR*2 POINT TO TEXT UNITS
         USING S99TUNIT,R5         R5 REG. BASE DSECT TEXT UNIT
         LA    R4,S99TUPL+L'S99TUPTR NEXT TUP ENTRY
         ST    R5,S99TUPTR            AND STORE IN TUP
         OI    S99TUPTR,S99TUPLN      INDICATE END OF TUP LIST
         MVC   S99TUNIT(RDDNLEN),RETDDN MOVE RETURN DDNAME TEXT
         DROP  R3,R4                  INFORM THE ASSEMBLER
         LR    R1,R6                  RB INTO R1
         DYNALLOC                     SVC 99 REQUEST
         LTR   R15,R15                ALLOCATE OKAY
         BZ    OPEN                   YES- DO OPEN
ERRDYNAM EQU   *
         WTO   'FTPSMFEX: ERROR DURING INTRDR DYNAMIC ALLOCATION'
```

```
         B     WRITESMF
OPEN     EQU   *
         MVC   DCB,INTRDR               COPY DCB TO GETMAINED AREA
         LA    R3,DCB                   ADDRESS OF DCB
         USING IHADCB,R3                INFORM THE ASSEMBLER
         MVC   DCBDDNAM,S99TUPAR        GENERATED DDNAME
         DROP  R5                       INFORM THE ASSEMBLER
         MVC   OPENL,OPENLIST           COPY OPEN LIST
         LA    R1,OPENL                 REMOTE PARAMETER LIST
         OPEN  ((R3),OUTPUT),MF=(E,(R1)) OPEN INTERNAL READER
         TM    DCBOFLGS,X'1Ø'            OPEN OKAY?
         BNZ   CJOBCRD
ERROPEN  EQU   *
         WTO   'FTPSMFEX: ERROR DURING INTRDR OPEN'
         B     WRITESMF                  YES
CJOBCRD  EQU   *                       *FIRST JOB
         MVC   JCLSUB,JOBCARD          *jobcard
         PUT   (R3),JCLSUB
         MVC   JCLSUB,PGM               *exec pgm=...
         PUT   (R3),JCLSUB
         MVC   JCLSUB,STARS             * * * * * * * *
         PUT   (R3),JCLSUB
         MVC   JCLSUB,COMM1            * ALERT     .....
         PUT   (R3),JCLSUB
         MVC   JCLSUB,BLNK             *
         PUT   (R3),JCLSUB
         MVC   JCLSUB,COMM2            * user:------ dataset:-----
         MVC   JCLSUB+1Ø(8),SMFFTPSU
         MVC   JCLSUB+27(44),SMFFTDSN
         PUT   (R3),JCLSUB
         CLI   SMFFTMEM,C' '
         BE    NOPDS
         MVC   JCLSUB,COMM22           *            member:-----
         MVC   JCLSUB+27(8),SMFFTMEM
         PUT   (R3),JCLSUB
NOPDS    EQU   *
         MVC   JCLSUB,BLNK             *
         PUT   (R3),JCLSUB
         MVC   JCLSUB,COMM3            * RETURN CODE      :
         MVC   JCLSUB+19(3),SMFFTSLR
         PUT   (R3),JCLSUB
BUILDBC  EQU   *
         XR    RØ,RØ
         L     RØ,SMFFTTBC
         CVD   RØ,DOPPIA
         UNPK  BYTECNT,DOPPIA
         OI    BYTECNT+14,X'FØ'
         MVC   JCLSUB,COMM5            *byte count-----
         MVC   JCLSUB+19(15),BYTECNT
```



```
         PUT   (R3),JCLSUB
BUILDIP  EQU   *                       BUILD OF CLIENT IP ADDRESS
         XR    RØ,RØ
         IC    RØ,SMFFTPSL
         CVD   RØ,DOPPIA
         UNPK  IPNUM1,DOPPIA+6(2)      FIRST IP NUMBER
         OI    IPNUM1+2,X'FØ'
         IC    RØ,SMFFTPSL+1
         CVD   RØ,DOPPIA
         UNPK  IPNUM2,DOPPIA+6(2)      SECOND IP NUMBER
         OI    IPNUM2+2,X'FØ'
         IC    RØ,SMFFTPSL+2
         CVD   RØ,DOPPIA
         UNPK  IPNUM3,DOPPIA+6(2)      THIRD IP NUMBER
         OI    IPNUM3+2,X'FØ'
         IC    RØ,SMFFTPSL+3
         CVD   RØ,DOPPIA
         UNPK  IPNUM4,DOPPIA+6(2)      FORTH IP NUMBER
         OI    IPNUM4+2,X'FØ'
         MVC   JCLSUB,COMM6            *client ip-----
         MVI   IPNUM1+3,C'.'
         MVI   IPNUM2+3,C'.'
         MVI   IPNUM3+3,C'.'
         MVC   JCLSUB+19(15),IPNUM
         PUT   (R3),JCLSUB
*        MVC   JCLSUB,COMM7            *client local-----
*        MVC   JCLSUB+18(44),SMFFTDS2
*        PUT   (R3),JCLSUB
         MVC   JCLSUB,STARS            * * * * * * * * * *
         PUT   (R3),JCLSUB
         MVC   JCLSUB,COMM8            *SEE  JOB.....
         PUT   (R3),JCLSUB
         MVC   JCLSUB,STARS            * * * * * * * * * *
         PUT   (R3),JCLSUB
CJOBCR2  EQU   *                       *SECOND JOB
         MVC   JCLSUB,JOBCAR2          *jobcard
         PUT   (R3),JCLSUB
         MVC   JCLSUB,PGM2             *EXEC PGM
         PUT   (R3),JCLSUB
         MVC   JCLSUB,DD2              *DD
         PUT   (R3),JCLSUB
         MVC   JCLSUB,SYSIN2           *SYSIN
         PUT   (R3),JCLSUB
         PUT   (R3),EOF                   /*EOF INFORM JES2
         MVC   CLOSEL,CLOSELST            MOVE CLOSE LIST
         LA    R1,CLOSEL                  REMOTE PARAMETER LIST ADDRESS
         CLOSE ((R3)),MF=(E,(R1))         CLOSE AND DEALLOCATE IR
         LR    R1,R13                    SAVE NEW SAVEAREA ADDRESS
         L     R13,PREVSA                SAVE NEW SAVEAREA ADDRESS
```

```
         FREEMAIN R,LV=WORKLEN,A=(R1)
         LTR   R15,R15
         BZ    WRITESMF
WRITESMF DS    ØH
         SR    R15,R15                 *NOT ONE OF OURS - WRITE SMFREC
DONE     RETURN (14,12),RC=(15)    RETURN TO CALLER
*
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
*-------------------------------------------------------------------
*    TEXT UNIT USED
*-------------------------------------------------------------------
         DS    ØF
RBLEN    EQU   (S99RBEND-S99RB)
SYSOUTAL DS     ØX
         DC     AL2(DALSYSOU)            ALLOCATE SYSOUT
         DC     X'ØØØ1'
         DC     X'ØØØ1'
         DC     C'A'
SYSOUTIR DS     ØX
         DC     AL2(DALSPGNM)            PROGRAM NAME INTRDR
         DC     X'ØØØ1'
         DC     X'ØØØ6'
         DC     C'INTRDR'
SYSOUTBK DS     ØX
         DC     AL2(DALBLKSZ)            BLKSIZE
         DC     X'ØØØ1'
         DC     X'ØØØ2'
         DC     X'177Ø'                  6ØØØ
SYSOUTRL DS     ØX
         DC     AL2(DALLRECL)            LRECL
         DC     X'ØØØ1'
         DC     X'ØØØ2'
         DC     X'ØØ5Ø'                  8Ø
SYSOUTRF DS     ØX
```



```
         DC     AL2(DALRECFM)            RECFM
         DC     X'ØØØ1'
         DC     X'ØØØ1'
         DC     X'9Ø'                    FB
DEALLOC  DS     ØX
         DC     AL2(DALCLOSE)            DEALLOCATE AT CLOSE
         DC     X'ØØØØ'
RETDDN   DS     ØX
         DC     AL2(DALRTDDN)            RETURN DDNAME
         DC     X'ØØØ1'
         DC     X'ØØØ8'
         DS     CL8                      DDNAME
RDDNLEN  EQU    *-RETDDN                 TEXT LENGTH
INTRDR   DCB   DDNAME=XXXXXX,DSORG=PS,MACRF=(PM)
DCBLEN   EQU   *-INTRDR
OPENLIST OPEN  (,),MF=L
OPENLEN  EQU   *-OPENLIST
CLOSELST CLOSE (,),MF=L
CLOSELEN EQU   *-CLOSELST
JOBCAR2  DC    CL8Ø'//XETCPFTP JOB ($AØØØØPØØØØØØ),CLASS=A,MSGCLASS=O'
PGM2     DC    CL8Ø'//STEPØØ1Ø EXEC U7SVC,PARM=''/LOGON ACABUS'''
DD2      DC    CL8Ø'//UCC7DATA DD *'
SYSIN2   DC    CL8Ø' DEMAND,JOB=XETCPFTP'
JOBCARD  DC    CL8Ø'//XETCPFT1 JOB ($AØØØØPØØØØØØ),CLASS=A,MSGCLASS=O'
PGM      DC    CL8Ø'//ALERT  EXEC PGM=IEFBR14'
STARS    DC    CL8Ø'//* * * * * * * * * * * * * * * * * * * * * * * *'
COMM1    DC    CL8Ø'//* ALERT: failed transmission        '
BLNK     DC    CL8Ø'//*                         '
COMM2    DC    CL8Ø'//* User:          Dataset: '
COMM22   DC    CL8Ø'//*                member : '
COMM3    DC    CL8Ø'//* return code  : '
COMM5    DC    CL8Ø'//* byte tras.   : '
COMM6    DC    CL8Ø'//* client ip    : xxx.xxx.xxx.xxx'
COMM7    DC    CL8Ø'//* client local : '
COMM8    DC    CL8Ø'//* see     job XETCPFT2    '
EOF      DC    CL8Ø'/*EOF'
PARMS    DSECT
PTRRC    DC    F'Ø'
PTRSMFR  DC    F'Ø'
*
FTPREC   DSECT                         *FTP SERVER SMF RECORD
FTPCMD   DC    CL4' '                  *FTP SUBCOMMAND
FTPREMIP DC    AL4(Ø)                  *FOREIGN HOST IP ADDRESS
FTPLOCIP DC    AL4(Ø)                  *LOCAL IP ADDRESS
         DS    ØF                      *REMAINDER OF RECORD NOT USED
WORKAREA DSECT
SAVEAREA DS    CL72                    SAVEAREA
PREVSA   EQU   SAVEAREA+4,4            ADDRESS OF PREVIOUS SAVEAREA
         DS    ØF
```

```
DOPPIA    DS     D
BYTECNT   DS     CL15
IPNUM     DS     ØCL15
IPNUM1    DS     CL3
          DS     CL1
IPNUM2    DS     CL3
          DS     CL1
IPNUM3    DS     CL3
          DS     CL1
IPNUM4    DS     CL3
          DS     ØF
DCB       DS     XL(DCBLEN)                DCB
OPENL     DS     XL(OPENLEN)               OPEN  LIST
CLOSEL    DS     XL(CLOSELEN)              CLOSE LIST
JCLSUB    DS     XL8Ø                      JCL AREA
          DS     ØF                        ALIGNMENT
SVC99STR  DS     XL66                      SVC99 STORAGE
WORKLEN   EQU    *-WORKAREA
*
          IEFZB476                * FOR DYNALLOC ERROR MESSAGES
          IEFZB4DØ                * DYNALLOC MACRO
          IEFZB4D2                * EQU FOR DYNALLOC KEYS
          DCBD DSORG=QS,DEVD=DA
          EZASMF76 FTP=YES        * SMF 118 FOR FTP
          END
//*
//STEPØØ3Ø EXEC PGM=IEWL,PARM=('AC=1,AMODE=31,RMODE=24,XREF,LET,LIST'),
//          COND=(8,LT,STEPØØ2Ø)
//SYSLIN    DD   DSN=&&OBJSET,DISP=(OLD,DELETE)
//          DD   DDNAME=SYSIN
//SYSLMOD   DD   DSN=TCPIP.USER.SEZALINK(FTPSMFEX),DISP=SHR
//SYSLIB    DD   DSN=TCPIP.USER.SEZALINK,DISP=SHR
//SYSUT1    DD   DSN=&&SYSUT1,UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD)),
//              SPACE=(1Ø24,(5Ø,2Ø))
//SYSPRINT DD   SYSOUT=*
```

FTPOSTPR

```
/*****************************************************************/
/*

  Function: Sample FTP User exit that allows for post-FTP
            processing
  Parameters being passed in from the FTP server via the
  parameter list:

   +Ø --   Pointer to the word with the user exit return code
   +4 --   Pointer to the number of parameters passed in
```

```
   +8 --   Pointer to the 8-byte buffer containing the USERID
   +12--   Pointer to the 4-byte client IP address
   +16--   Pointer to the 2-byte client port number
   +2Ø--   Pointer to the 4-byte character string with current
             directory type:
               MVS or HFS (left justified)
   +24--   Pointer to a buffer that contains the current directory
             value, the first two bytes hold the length of the
             remaining buffer.
   +28--   Pointer to the 4-character byte field that contains the
             current filetype (SEQ, JES, SQL), left justified
   +32--   Pointer to the 3-character byte field that contains the
             current FTP reply code
   +36--   Pointer to buffer that contains FTP reply string; first
             two bytes contain the length of the remaining buffer
   +4Ø--   Pointer to the 4-byte field that contains the current
             FTP command code
   +44--   Pointer to the 1-char byte field that contains the current
             CONDDISP setting-
               C for catalog, D for delete
   +48--   Pointer to the 4-byte binary field that contains the close
             reason code:
               Ø -- transfer completed normally
               4 -- transfer aborted before data connection was
                      established
               8 -- transfer aborted with socket communication errors
              12 -- transfer aborted after data connection was
                      established
              16 -- transfer aborted with SLQ file errors after data
                      connection was established
*/
/*****************************************************************/
/*                                                               */
/*       FTPOSTPR USER EXIT                                      */
/*                                                               */
/*****************************************************************/

œpragma linkage(FTPOSTPR, fetchable)


œdefine _XOPEN_SOURCE_EXTENDED 1

œinclude <stdio.h>
œinclude <stdlib.h>
œinclude <syslog.h>
œinclude <dynit.h>
```

```
/* set up structure needed for current directory value         */
typedef struct ...
                 short dirlen;
                 char  dirnameí11ØØù;
               _currdir;

/* set up structure needed for reply string value              */
typedef struct ...
                 short replylen;
                 char  replyí12ØØù;
               _replystr;



/* beginning of FTPOSTPR function                              */
int FTPOSTPR(int *exitrc, int *numparms, char exitusridí8ù,
             unsigned long *clientIP, unsigned int *clientport,
             char dirtypeí4ù, currdir *cwd, char filetypeí4ù,
             char replycodeí3ù, replystr *rs, char cmdcodeí4ù,
             char *conddispvalue, int *closerc)
...

   char useridí9ù;
   char tempí6Øù;
   char temprepcí4ù;
   char bufferí8Øù;
   int intrc = Ø;
   replystr temprc;
   unsigned long cip = *clientIP;
   FILE *fpr;
   char *jc = "//XETCPFT2 JOB ($AØØØØPØØØØØØ),CLASS=A,MSGCLASS=O";
   __dyn_t ip;

   memset(temprepc,'‡Ø',4);
   memcpy(temprepc,replycode,3);
   if (strcmp(temprepc,"25Ø") == Ø) ...
     *exitrc  = intrc ;
     return;
     _
   memset(userid,'‡Ø',9);
   memcpy(userid,exitusrid,8);

   temprc = *rs;
   memset(temp,'‡Ø',6Ø);
   memcpy(temp,temprc.reply,59);

   intrc = dyninit(&ip);     /* initialization of dynalloc structure */
   if (intrc) ...
     syslog(LOG_ERR,"dyninit");
```



```
    *exitrc  = intrc ;
    return;
    _
  ip.__sysout='A';
  ip.__sysoutname="INTRDR";
  ip.__ddname="INTRDRDD";

  intrc = dynalloc(&ip) ;        /* allocation of internal reader */
  if (intrc ,= Ø)  ...
    syslog(LOG_ERR,"dynalloc");
    *exitrc  = (1ØØØØ + ip.__infocode) ;
    return;
    _

  fpr = fopen("DD:INTRDRDD","w,lrecl=8Ø,recfm=fb,blksize=312Ø");
  if(fpr == NULL) ...
    syslog(LOG_ERR,"fopen");
    dynfree(&ip);                     /* free internal reader  */
    *exitrc  = 2Ø999 ;
    return;
    _

  memset(buffer,' ',79);              /* initialize record to write */
  bufferí8Øù='';
  strncpy(buffer,jc,49);              /* write jobcard */
  intrc = fputs(buffer,fpr);
  if (intrc == EOF)  ...
    syslog(LOG_ERR,"fputs");
    *exitrc  = (6ØØØ1) ;
    fclose(fpr);                      /* close internal reader */
    dynfree(&ip);                     /* free internal reader  */
    return;
    _
  strcpy(buffer,"‡n//ALERT EXEC PGM=IEFBR14");
  fputs(buffer,fpr);
  strcpy(buffer,"‡n//* ");
  fputs(buffer,fpr);
  strcpy(buffer,"‡n//* ftp failed: ");
  strcat(buffer,userid);
  fputs(buffer,fpr);
  strcpy(buffer,"‡n//* return code: ");
  strcat(buffer,temprepc);
  fputs(buffer,fpr);
  strcpy(buffer,"‡n//* ");
  strcat(buffer,temp);
  fputs(buffer,fpr);

  intrc = fclose(fpr);                /* close internal reader */
  if (intrc) ...                        /* and submit            */
    syslog(LOG_ERR,"fclose");
```

```
     dynfree(&ip);                    /* free internal reader  */
     *exitrc  = (30000 + intrc) ;
     return;
     _
   intrc = dynfree(&ip);              /* free internal reader  */
   if (intrc) ...
     syslog(LOG_ERR,"dynfree");
     *exitrc  = (40000 + intrc) ;
     return;
     _
_
```

COMPILE AND LINK JOB

I've included the compile and link job below, as we encountered many problems caused by coding the wrong compiler options.

```
//CO076LXC JOB ($A0000P000000),CLASS=A.MSGCLASS=9
//*******************************************************************
//*                                                                *
//*  COMPILE, PRE-LINK (REQUIRED FOR REENTRANCY) LINK EDIT AND     *
//*  EXECUTE A C PROGRAM                                           *
//*                                                                *
//*  C/C++ FOR MVS/ESA                                             *
//*                                                                *
//*  RELEASE LEVEL: 03.02.00  (VERSION.RELEASE.MODIFICATION LEVEL) *
//*                                                                *
//*******************************************************************
//*
//EDCCPLG  PROC  INFILE='SYS5.LIB.JCL(FTPOSTPR)',
//  CREGSIZ='4M',                      < COMPILER REGION SIZE
//  CRUN=,                             < COMPILER RUNTIME OPTIONS
//  CPARM='DEFINE(MVS),SOURCE,LIST,OFFSET,LONGNAME,NOOPTIMIZE',
//  CPARM2='RENT',
//  CPARM3=,                           < COMPILER OPTIONS
//  SYSLBLK='3200',                    < BLOCKSIZE FOR &&LOADSET
//  LIBPRFX='CEE',                     < PREFIX FOR LIBRARY DSN
//  LNGPRFX='CBC',                     < PREFIX FOR LANGUAGE DSN
//* CLANG='EDCMSGE', < NOT USED IN THIS RELEASE. KEPT FOR COMPATIBILITY
//  PLANG='EDCPMSGE',                  < PRE-LINKER MESSAGE NAME
//  PREGSIZ='2048K',                   < PRE-LINKER REGION SIZE
//  PPARM='MAP,NCAL,NOER',        < PRE-LINKER OPTIONS
//  LREGSIZ='1024K',                   < LINK EDIT  REGION SIZE
//  LPARM='XREF,LET,LIST,MAP,RENT,AC=1',      < LINK EDIT  OPTIONS
//  DCB80='(RECFM=FB,LRECL=80,BLKSIZE=3200)',      <DCB FOR LRECL 80
//  DCB3200='(RECFM=FB,LRECL=3200,BLKSIZE=12800)', <DCB FOR LRECL 3200
//  OUTFILE='TCPIP.USER.SEZALINK'
//*
```

```
//*----------------------------------------------------------------------
//*  COMPILE STEP:
//*----------------------------------------------------------------------
//**MPILE EXEC PGM=CBC32ØPP,REGION=&CREGSIZ,
//COMPILE EXEC PGM=CBCDRVR,REGION=&CREGSIZ,
//    PARM=('&CRUN/&CPARM &CPARM2 &CPARM3')
//STEPLIB  DD  DSNAME=&LIBPRFX..SCEERUN,DISP=SHR
//         DD  DSNAME=&LNGPRFX..SCBCCMP,DISP=SHR
//SYSMSGS  DD  DUMMY,DSN=CEE.SCEEMSGP(&PLANG),DISP=SHR
//SYSIN    DD  DSNAME=&INFILE,DISP=SHR
//USERLIB   DD DSN=TCPIP.SEZACMAC,DISP=SHR
//SYSLIB   DD  DSNAME=&LIBPRFX..SCEEH.H,DISP=SHR
//         DD  DSNAME=&LIBPRFX..SCEEH.SYS.H,DISP=SHR
//SYSLIN   DD  DSNAME=&&LOADSET,UNIT=VIO,
//             DISP=(MOD,PASS),SPACE=(TRK,(3,3)),
//             DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=&SYSLBLK)
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSCPRT  DD  SYSOUT=*
//SYSUT1   DD  UNIT=VIO,SPACE=(32ØØØ,(3Ø,3Ø)),DCB=&DCB8Ø
//SYSUT4   DD  UNIT=VIO,SPACE=(32ØØØ,(3Ø,3Ø)),DCB=&DCB8Ø
//SYSUT5   DD  UNIT=VIO,SPACE=(32ØØØ,(3Ø,3Ø)),DCB=&DCB32ØØ
//SYSUT6   DD  UNIT=VIO,SPACE=(32ØØØ,(3Ø,3Ø)),DCB=&DCB32ØØ
//SYSUT7   DD  UNIT=VIO,SPACE=(32ØØØ,(3Ø,3Ø)),DCB=&DCB32ØØ
//SYSUT8   DD  UNIT=VIO,SPACE=(32ØØØ,(3Ø,3Ø)),DCB=&DCB32ØØ
//SYSUT9   DD  UNIT=VIO,SPACE=(32ØØØ,(3Ø,3Ø)),
//             DCB=(RECFM=VB,LRECL=137,BLKSIZE=882)
//SYSUT1Ø  DD  SYSOUT=*
//SYSUT14  DD  UNIT=VIO,SPACE=(32ØØØ,(3Ø,3Ø)),
//             DCB=(RECFM=FB,LRECL=32ØØ,BLKSIZE=128ØØ)
//*
//*-------------------------------------------------------------
//* PRE-LINKEDIT STEP:
//*-------------------------------------------------------------
//PLKED   EXEC PGM=EDCPRLK,PARM='&PPARM',COND=(4,LT,COMPILE),
//    REGION=&PREGSIZ
//STEPLIB  DD  DSNAME=&LIBPRFX..SCEERUN,DISP=SHR
//SYSMSGS  DD  DSNAME=CEE.SCEEMSGP(&PLANG),DISP=SHR
//SYSLIB   DD  DUMMY
//SYSIN    DD  DSN=*.COMPILE.SYSLIN,DISP=(MOD,PASS)
//         DD  DDNAME=SYSIN2
//SYSMOD   DD  DSNAME=&&PLKSET,UNIT=VIO,DISP=(NEW,PASS),
//             SPACE=(32ØØØ,(3Ø,3Ø)),
//             DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=&SYSLBLK)
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSIN2   DD  DUMMY
//*
//*----------------------------------------------------------------------
//* LINKEDIT STEP:
//*----------------------------------------------------------------------
//LKED   EXEC PGM=IEWL,COND=((4,LT,COMPILE),(4,LT,PLKED)),
//    REGION=&LREGSIZ,PARM='&LPARM'
//SYSLIB   DD  DSN=&LIBPRFX..SCEESPC,DISP=SHR
//         DD  DSNAME=&LIBPRFX..SCEELKED,DISP=SHR
//         DD  DISP=SHR,DSN=SYS1.CSSLIB
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DSNAME=*.PLKED.SYSMOD,DISP=(OLD,DELETE)
//         DD  DDNAME=SYSIN
```

IBM's latest foray into Web-to-host

On 27 May 2003, IBM announced Version 3.1 of its Host Integration Solution bundle, Version 3 having been unveiled in September 2002. Version 3.1 adds two new and very interesting offerings to this already value-packed bundle:

- *WebSphere Application Server Network Deployment (WAS ND) V5* (where the V5 denotes that it is a valid, 'in sync' companion product to WebSphere Application Server V5). WAS ND is a major revitalization of what used to be known as IBM's Edge Server. This, as its name suggested, was meant to be deployed at the edge of the network, or at the very least in front of the data centres. At a minimum, it would provide load-balancing across servers, dynamic content caching, and centralized security. Performing these functions as close to the end users as possible minimizes the traffic that has to cross a corporate backbone. Edge server functionality therefore improves response times, reduces network congestion, and enhances server efficiency. And centralized security ensures that users can be validated before they get even close to an IT server.

  As well as all of the edge server functionality, WAS ND V5 also includes a private UDDI registry for advertising in-house Web services, as well as a Web services gateway that allows users outside a corporate firewall to securely invoke Web services located behind the firewall.

- *WebSphere Studio Developer (WSSD) V5.0*. WSDD is a template and wizard-empowered rapid application development environment for building Java and Web applications as well as Web services. It also allows you to easily augment Java and Web applications with HTML, JSP, and JavaScript embellishments. With the inclusion of WSDD V5.0, IBM appears to have removed the Screen Customizer 2.0 'GUI builder' from the HIS bundle.

All in all, with the addition of WAS ND and WSDD, IBM has made the already attractive value proposition of the Host Integration Solution (HIS) even more compelling.

HIS provides an enormous amount of premium IBM software at fairly aggressive pricing, and this has always been the rationale for considering IBM's HIS approach. It provides you with everything you need – and more – to implement a comprehensive, multi-faceted Web-to-host solution that even extends to host integration: for example, the ability to capture and reuse the business logic in existing mission-critical applications in the form of either JavaBeans or XML Web services. IBM's Host Publisher, which is included within HIS, is one of the most powerful solutions on the market for realizing host integration.

The bottom line here is that, with HSS V3.1, IBM has yet again upped the ante with regard to Web-to-host. The big question is, however, whether this has all simply come too late. The problem is that Web-to-host in the TCP/SNA world (where this was an umbrella term to cover all approaches to accessing mainframe or AS/400 applications using Web browser-centric solutions) never actually lived up to its expectations.

A QUICK RECAP OF WHAT HAPPENED TO WEB-TO-HOST

Web-to-host, which was born around 1996, was never mere hype. Not only did it work, but it worked very well and, what's more, could genuinely slash host access costs. An added bonus was that most Web-to-host solutions also included powerful but relatively easy-to-use facilities for rejuvenating harsh, textual 3270/5250 'green screens'. Web-to-host solutions were either browser-invoked 'thin-clients' (ie Java- or ActiveX-based 3270/5250 emulators) or zero footprint, server-based 3270/5250-to-HTML converters. So, although Web-to-host pricing was typically one third to one fifth that of previous 'fat client' terminal emulators, the major cost savings with Web-to-host came from not having to install and maintain host emulation software on individual desktops.

Repeated studies have shown that eliminating this need to install and maintain terminal emulation software on individual desktops would save enterprises, on average, $150 per desktop per year. For enterprises with 500 or more host access 'seats' this should have been a 'no brainer'. Yet, despite these hard-to-ignore economics, and the fact that in 2000 at least 65 vendors were actively promoting various flavours of this technology, less than 40% of the overall mainframe and AS/400 installed base pursued Web-to-host. Amongst the many possible reasons for this were Y2K and 9/11, coupled with the fact that there was no true and vociferous market leader. Although IBM, with Host On-Demand and later with Host Publisher V2.2 (July 2000), had leading-technology offerings, it wasn't until 2002 that it finally asserted its clout in this marketplace. By then, however, Web-to-host was in the doldrums and many of the other vendors had fallen by the wayside.

IBM TIPS ITS HAT

In September 2002, IBM introduced the WebSphere Host Access Transformation Server (HATS) V4 (the V4 referring to its allegiance to and dependence on WebSphere Application Server (WAS) V4.0). Others, including ResQNet, iE, Novell, and Zephyr, had introduced similar products much earlier, but they lacked IBM's credibility, reach, or influence, and HATS was long overdue. In effect, HATS validated technology that had been introduced by other vendors two to three years before IBM. In fact, IBM, as only IBM can, had started to market a product similar to HATS from the French Crys@lid Server company just a week before it unveiled HATS – but that's IBM for you.

The HATS server is a J2EE-compliant, 'on-the-fly', 3270/5250-to-HTML converter. It's now supported on OS/390, OS/400, Windows NT/2000, AIX, and Sun Solaris 7.0(+) platforms. The big thing with HATS is that it's a zero-footprint (ie no software other than a Web browser is required at the client end), host-to-HTML conversion that is rules-based, and is a 'near load-and-go' (ie 'plug-and-play') solution.

As with other comparable offerings, HATS consists of two components: the Java-based runtime server component and a wizard-empowered studio component that runs on a Windows XP system. A neat twist, however, is that the HATS studio is fully integrated with the 'open-source' Eclipse imitative-based WebSphere Studio. With HIS V3.1 this studio is further augmented with WSSD V5.0.

HATS performs rules-based conversion. A set of default rules is provided by IBM, and these rules can then be modified using the studio. HATS supports macros to facilitate programmed traversal across multiple host screens. In a neat touch, it can also reuse previously defined Host On-Demand and Host Publisher macros. The HATS transformation rules support a now familiar array of GUI elements such as drop-down lists, hot links, tables, buttons, and tabbed folders, as well as enabling the inclusion of HTML 'objects' such as backgrounds, logos, pictures, and Web links.

A novel feature (which will be loved by some, hated by others, and will totally baffle those in the middle) is the local print capability realized using Adobe Acrobat Portable Document Format (PDF) files. This is clever, but there must be a lot of 'green screen' users who have never yet seen a PDF. It will take some deft persuasion to get these users comfortable with the notion of local print happening in the form of a file that has then to be printed, as a second step after being opened using Acrobat.

COMPOSITION OF HIS V3.1

V3.1 is the fourth major release of the HIS concept, which was first available in mid-1998 with just four products in the bundle. The V3.1 bundle, with eight products, is four times as big. The eight products that make up the HIS V3.1 bundle are:

1 *Host On-Demand V7.0*, best-of-breed, Java applet-based 'thin client' 3270, 5250, and VT emulator with integrated digital certificate-based security and integration with IBM's WebSphere Portal family via a 'portlet'.

2 *Personal Communications (PComm) V5.6 for Windows*, IBM's feature-rich 'fat client' terminal emulator that has been around for so long that one can safely say that it is indeed the grandaddy of all 3270/5250 emulators.

3 *Host Publisher V4.0*, which is a powerful 3270/5250-to-HTML/XML converter that can also be used for host integration.

4 HATS V4.0.

5 *WebSphere Application Server*, IBM's market-leading J2EE-compliant software platform for deploying and executing Java applications, Java components, and Web services.

6 WAS ND V5.0.

7 WSSD V5.0.

8 *Communications Server (CS) for AIX and Windows NT/2000*, IBM's long-standing but technology-leading 'host gateway' with powerful tn3270(E) and tn5250 server capabilities.

BOTTOM LINE

HIS has always been a cost-effective and convenient way to obtain all of the software you need to implement a comprehensive Web-to-host solution set that covers all bases, whether it be 'tn' communications using a 'fat client' (ie Pcomm), 'thin-client' access using Host On-Demand, zero footprint HTML-based access using Host Publisher or HATS, or even Host Integration. The inclusion of WAS and CS server components provided the icing on the cake. By including WAS ND, IBM has made the deal even sweeter.

With HIS V3 and V3.1, there are two very distinct pricing structures. You can either buy licences per seat (ie each user who needs host access) or buy a concurrent user licence (which is really the only option if you plan to offer Web-based access using Host Publisher or HATS). The US pricing for V3.0 is $303 per seat or $446 per concurrent user. The concurrent user approach will typically prove better value, given that not all users require host access at the same time. Given the amount of software included, the concurrent user pricing is not as bad as it might look at first sight. The bottom line, however, is that if you are only now looking at Web-to-host, the HIS V3.1 is a good place to start your evaluations.

Anura Gurugé
Strategic Consultant (USA)

TCP/SNA news

SDS has announced its VIP software, offering browser-based real-time performance monitoring, management, and troubleshooting for z/OS (or OS/390) TCP/IP environments. Features include state-of-the-art trace, ping, & NetView interface.

URL: http://www.sdsusa.com/vip

* * *

Computer Associates has announced NetMaster Network Management for TCP/IP 7.0, for mainframe-connected TCP/IP and SNA networks. It offers Web browser access to real-time management and historical reporting, and also has Web interfaces for problem diagnosis and performance management.

URL: http://www3.ca.com/Files/ProductAnnouncements/uni_netmaster_network_management_for_tcpip_7.pdf

* * *

IBM has announced Version 3.1 of its WebSphere Host Integration software, with improved graphical editing environment, better stylesheet editor, and better integration between WebSphere Studio and WebSphere Application Server.

IBM has also announced Version 3.1 of its WebSphere Host Integration Solution for iSeries.

URL: http://www.ibm.com/software/webservers

* * *

IBM has announced z/VM V4.4, which includes self-protection enhancements to TCP/IP for z/VM, providing better protection and management of the communication stack, and cascaded FICON director support to help detect and report miscabling actions.

URL: http://ibm.com/eserver/zseries/zvm/

* * *

HostBridge Technology is making available, at no cost, a collection of sample programs that it has developed to make it easy for a CICS program to send an outbound TCP/IP or http request.

These programs could serve as sample code to those interested in writing their own CICS socket I/O programs or adding socket support within their own programs. The programs do not require HostBridge, but they were originally written for one of its customers.

URL: http://www.hostbridge.com/downloads

* * *

Spinnaker Networks has announced SPEC SFS benchmark results, claiming its SpinServer 4100 to be the best performing NAS solution on the market.

URL: http://www.spinnakernet.com

* * *