56

TCP/SNA update

December 2004

In this issue

A tc PUBLICATION

TCP/SNA Update

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

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

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Bob Thomas
E-mail: info@xephon.com

Subscriptions and back-issues

A year's subscription to *TCP/SNA Update*, comprising four quarterly issues, costs $190.00 in the USA and Canada; £130.00 in the UK; £136.00 in Europe; £142.00 in Australasia and Japan; and £140.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the March 2000 issue, are available separately to subscribers for $49.50 (£33.00) each including postage.

TCP/SNA Update on-line

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

Disclaimer

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

Contributions

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

Printed in England.

Remote printing using JES/328X

Many installations using remote printers use the JES/328X print facility to satisfy their printing needs. The article is aimed at beginners who wish to implement a remote printer quickly. Implementation is discussed based on a simple example of an LPAR (LPAR A) having a channel-attached 3174 establishment controller and remote printer connected to the 3174. It is assumed that JES/328X is already installed and is available for use. VTAM, JES2, and JES/328X are the major participants in remote printing. Remote printing takes places by LU–LU communications as shown in Figure 1, and the LU details are:

- PLU (Primary LU) – JESLU
- SLU (Secondary LU) – RMT10
- TLU (Tertiary) – PRTLU.

To implement this arrangement, certain definitions are required

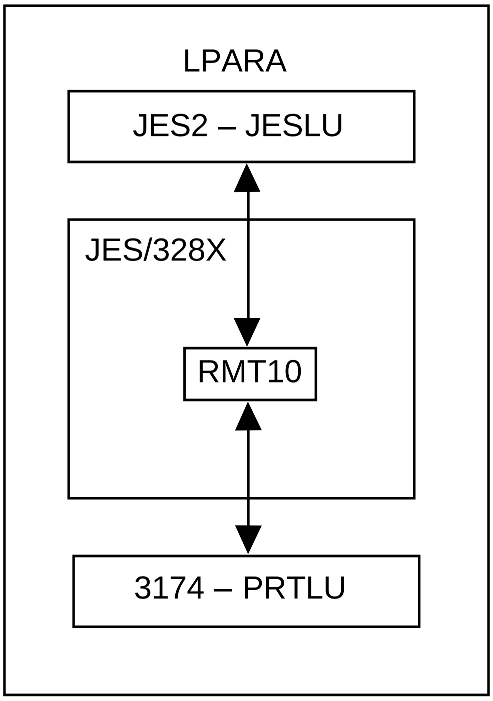


Figure 1: Remote printing

in JES2, JES/328X, and VTAM. The definitions may vary slightly from OS version to version. Here the version considered is OS/390 V2R10. The use of discretion is advised in applying the definitions given below to their installation.

JES2 PARAMETERS (JES2PARM)

The following definitions are required in the JES2 parameters for remote printing:

```
APPL(JESLU)  NODE=1

RMT1Ø    DEVTYPE=LUTYPE1,
         SETUP=PDIR

 R1Ø.PR1 WS=(W,R,Q,PMD,LIM/F,P),
           PRWIDTH=132,
           CKPTLINE=72,
           NOFCBLOD,
           CLASS=Q,
           ROUTECDE=LOCAL,
           CKPTPAGE=1
```

VTAM DEFINITIONS

The following VTAM application definitions are required for remote printing.

Primary LU (JES2 application definitions):

```
APPLJES2 VBUILD TYPE=APPL
*
JESLU APPL EAS=1Ø,
               ACBNAME=JESLU,
               AUTH=(ACQ),
               VPACING=1Ø
```

Secondary LU definitions:

```
RMT1Ø  APPL  EAS=1,AUTH=ACQ,MODETAB=MTJSX,DLOGMOD=JES4Ø96,VPACING=2
```

Tertiary LU definitions:

```
L31LØ32Ø LBUILD

   PRTLU LOCAL CUADDR=328,ISTATUS=ACTIVE,DLOGMOD=DCS4KN,TERM=3286,      X
               MODETAB=MTBAZ,FEATUR2=PRINTR
```

JES/328X PARAMETERS

The following definitions are required in the JES/328X parameters for remote printing:

```
A,SLU=RMT1Ø,PLU=JESLU,TLU=PRTLU,LUTYPE=Ø,DEVICE=FAM1
F,SLU=RMT1Ø,JES=JES2,CONSLOGP='SYS1.BAZ',CANCEL='$C  PR1'
F,SLU=RMT1Ø,BUFSIZE=2Ø48,ERRMSG=ALL,CLSDELAY=Ø,LOGMODE=DSC4KN
F,SLU=RMT1Ø,INITFORM=IGNORE,RETRYDEL=3ØØ
F,SLU=RMT1Ø,XLATE=BASE,BASE=STD
*
S,RMT1Ø
```

GETTING THE PRINTER WORKING

To get the remote printer working do the following:

1 Activate APPLJES2

2 Issue console command $S LOGON1

3 Activate APPLRMT

4 Activate Tertiary LU

5 Start the JSX started task.

After successfully issuing the above commands, datasets can be printed by using commands like PRINTDS with an appropriate output class.

Arun Kumaar R
System Software Group
Tata Consultancy Services (India)

A mainframe-centric comparison of KVMs

If you have ever connected to a mainframe from a workstation equipped with a Keyboard-Video-Mouse (KVM) switch, frustration is probably the best word to describe the experience. In my 2002 side-by-side hands-on comparison of 10 KVMs, it was clear to me that few, if any, had been tested with 3270 emulation software.

Since then, not much has changed in the KVM market. Newer models offer USB cabling instead of the separate PS/2 keyboard and mouse connections that are standard on the products reviewed here.

WHY KVM?

Whether you're building a Network Control Centre (NCC) or just running a couple of workstations as a test network of your own, having a keyboard, mouse, and monitor for each machine just isn't practical. Quite apart from the issue of desk space (footprint), there's no ergonomic solution to frequently moving between two physical keyboards and mice.

Enter the Keyboard-Video-Mouse (KVM) switch, which allows one or more keyboards, mice, and monitors to control two or more computers. However, it's often more practical to share only the keyboard and mouse, retaining a monitor per computer. As we'll see, KVMs work even better in that environment.

THE EMPHASIS IN TESTING

KVMs were not built with mainframes in mind. At least, not the way I use workstations to access mainframes. I am still heavily committed to the use of the PC's left and right Ctrl keys as *3270 Reset* and *Enter*, respectively, based on the position of those keys on every 3270 terminal that was ever made by IBM.

And that creates a problem on many KVMs, because many use one or both of the Ctrl keys as part of a hot key sequence to perform special functions, such as switching between computers. Admittedly, you generally have to hit Ctrl twice before you run into trouble, but I find myself doing that a lot.

A related problem with some KVMs affects the use of Ctrl-click and Shift-click, where *click* refers to a left-button mouse click. These are both standard Windows actions to extend a current selection to include more items in a list.

Any, even subsecond, delay in the KVM's transmitting the Ctrl or Shift to the computer often means it will arrive after the mouse click. Accepted alone, the mouse click selects the current item and deselects any previous items – which can be very inconvenient.

WHAT WAS TESTED

The focus is on the low end: two- and four-port KVMs. At first glance, it may seem senseless to test two- and four-port models from the same manufacturer, especially the ones that look almost identical. But, as we'll see, looks can be deceiving. Mainframe users may find a four-port model works quite well even when its two-port sibling is rather annoying.

When testing began in 2001, the emerging standard was PS/2 connections for both the mouse and keyboard. And that is what we tested. In 2002, USB KVMs began emerging, where the mouse and keyboard still plug into PS/2 ports on the KVM, but the connections from KVM to workstation ports are USB. After the formal tests, the Belkin was informally tested as a USB KVM.

BUYING ADVICE

Be sure to include the cables when preparing cost estimates for KVMs, especially USB KVMs. Here are a few things I learnt during testing:

- It is important to consider user ergonomics (easy reach) when placing a frequently-used KVM.
- The cables must be long enough!
- None of the USB KVMs I looked at required the more expensive USB 2.0 cables.
- All of the USB KVMs I looked at required a cable with a USB B plug on one end, rather than the more common USB A to USB A cable.
- Combined monitor/USB cables often cost little more than an equivalent monitor-only cable and are slightly cheaper than PS/2 KVM cables.

THE TESTS THEMSELVES

With the exception of the one AESP unit that requires power, all tests were performed without power, even when a power adapter was supplied with the KVM. There was never any indication that external power would have affected the test results.

One subtle difference between keystrokes:

- key-key means both keys are held down at the same time.
- key key means that each key is pressed in sequence.

The term ‘oriented’, referring to KVM cables, indicates that the round keyboard and mouse connectors are flat on one side to help line up the DIN connector with the plug. Once you get used to it, you can look at the position of the plastic key in the plug and automatically rotate the connector to the right place first time every time.

Figure 1 displays the observations and test results in table format.

KVM MANUFACTURERS

Because of the large number of products available in the marketplace, not all low-end KVMs were tested. But a fairly comprehensive list of KVM manufacturers was identified (see below). KVMs come in all sizes and shapes, and most of the companies listed specialize in certain segments of the market. KVM manufacturers we found were:

- Adder Technology – http://www.addertec.com/products
- Addlogix – http://www.addlogix.com/kvms
- AESP – http://www.signamax.com/signmax1_network/kvmswitch.htm
- APC – http://www.apcc.com/products/family/index.cfm?id=62
- Apex – now Avocent

1	Manufacturer	**AESP**	**AESP**	**Avocent**	**Belkin**
2	Brand	**CommandView**	**CommandView**	**SwitchView**	**OmniView**
3	Model number	KVM-211	098-8040	10045	F1DS102T
4	Part number (if different)	098-8025		520-195-005	
5	Similar models		098-8020 098-8060	12045	F1DS10nx $75+ (n=2,4; x=P,T,U)
6	Number of computer ports	2	4	4	2
7	Number of user ports	1	1	1	1
8	List price (US)	$84	$189	$120	$150
9	Best Web price (US)	$61	$130	$76	$98
10	Internet domain name	signamax.com	signamax.com	avocent.com	belkin.com
11	Keyboard port type	PS/2	PS/2, AT	PS/2	PS/2
12	Mouse port type	PS/2	PS/2	PS/2	PS/2
13	Computer mouse cable port type	PS/2	PS/2, serial	PS/2	PS/2, USB
14	Audio switching	None	None	None	Yes
15	Number and stated length of included cables	None	None	None	2xPS/2 ext. cables
16	Additional cables (length, cost)	6ft., C-69T-6B	6ft., C-69T-6B	6ft., $29 CPS2-6A	F1D9101-xx F1D9100-xx $24.99-$29.99 (xx = 6-10ft)
17	Maximum supported cable length	30ft. or more	100-500ft.	500 feet with LongView (LV210-AM)	500ft with extender (F1D084)
18	Cables tip the KVM back?	Yes	No	Yes	No
19	Cable combines keyboard, video and mouse into single connection to KVM switch?	No	No	No	No
20	Cable: keyboard and mouse connectors oriented?	Yes	Yes	Yes	No
21	Colour-coded keyboard/mouse ports	Purple/green	Purple/green	Purple/green	Purple/green
22	Colour-coded keyboard/mouse cables	Purple/green	Purple/green	Purple/green	Grey/black housing, purple/green plugs
23	Keyboard and mouse connect to front or back?	Back	Back	Back	Back
24	Monitor connects to front or back?	Back	Back	Back	Back
25	Computers connect to back or sides?	Back	Back	Back	Back
26	Buttons	One small Select button	Select	One, switches between computers	Two, one for each computer
27	Push button(s)	OK	OK	Must hit in centre; interprets 2 rapid pushes as 1	OK

Figure 1a: Test results for CommandView, SwitchView and OmniView (continues)

28	Status lights	PC1, PC2, Scan	Scan, Power, and Ready and Active for each computer	Active and Selected for each computer	Two, one for each computer
29	Power required?	No	Yes	No	No
30	Type of alternative power	N.a.	9v. 500ma.	6v. 700ma.	9v. 600ma.
31	Alternative power Included?	N.a.	Yes	No	No
32	Installation manual	Paper	Paper	CD-ROM	Paper (2)
33	Is the manual just for this model?	Yes	No	No	No
34	Size of KVM (Imperial)	4.6"x3.7"x1.6"	10.3"x5.9"x3.1"	8"x5"x1.8"	6.5"x6.2"x2.5"
35	(Metric)	119x95x40mm	263x150x82mm	204x125x46mm	166x155x65mm
36	Weight of KVM (Imperial)	6.5oz.	4.3lb.	15oz.	14.5oz.
37	(Metric)	180g.	1.9kg.	413g.	404g.
38	Shape of KVM	Rectangular with raised back	Rectangular with rubber feet	Rectangular	Six-sided, rounded
39	Warranty period	Lifetime	5 years	1 year	5 years
40	Command mode sequence	ScrollLock ScrollLock	Ctrl+Alt+Shift	Ctrl Ctrl	ScrollLock ScrollLock
41	Can it be changed?	No	No	Yes	No
42	Does change survive power off?	N.a.	N.a.	No	N.a.
43	Changed to what during testing?	N.a.	N.a.	Alt Alt	N.a.
44	LeftCtrl LeftCtrl	OK	OK	OK	OK
45	RightCtrl RightCtrl	OK	OK	OK	OK
46	LeftCtrl RightCtrl	OK	OK	OK	OK
47	LeftCtrl LeftCtrl Tab	OK	OK	OK	OK
48	Instant Shift-Click	OK	OK	OK	OK
49	Instant Ctrl-Click	OK	OK	OK	OK
50	Intel D850GBC BIOS boot errors on KVM without power?	No	No	No	No
51	On KVM without power, is Port 2 automatically selected when Computer #2 is powered up?	Yes	No	Yes	Yes
52	When different computer selected during startup, mouse works	Yes	Yes	Yes	Yes
53	Two computers with different screen resolutions	OK	OK	OK	OK
54	Monitor's max. screen resolution (sensed by Windows) when computer selected during entire boot/startup process	Correct	Correct	Sometimes incorrect (Port 1 XP)	Correct
55	...with different computer selected during startup	Incorrect	Incorrect	Incorrect	Incorrect
56	Change screen resolution with Control Panel Display	OK	OK	OK	OK

Figure 1a: Test results for CommandView, SwitchView, and OmniView (continued)

Figure 1a: Test results for CommandView, SwitchView, and OmniView (continued)

57	...after boot to Logon panel (Ctrl-Alt-Del) with different computer selected	OK	OK	OK	OK
58	Delay in mouse switching #1 to #2	Yes	Yes	Minimal	No
59	Delay in mouse switching #2 to #1	Yes	Yes	Minimal	No
60	A moving mouse will not switch	Yes	Yes	No	Yes
61	Intellimouse with software	Works	Jumps when switched	Works	Works
62	KVM works without monitor connected?	Yes	Yes	Yes	Yes
63	Hot key sequence to switch to Port #n	ScrollLock ScrollLock n	Ctrl+Alt+Shift n Enter	Ctrl Ctrl letter Enter	ScrollLock ScrollLock n
64	Next Computer Hot Key sequence	N.a.	N.a.	N.a.	ScrollLock ScrollLock Up
65	Previous Computer Hot Key sequence	N.a.	N.a.	N.a.	ScrollLock ScrollLock Down
66	Auto Scan capability?	Yes	Yes	Yes	Yes
67	...button(s) to push to start	Hold SELECT for 1 sec.	Hold SELECT for 1 sec.	No	No
68	...key sequence to start	ScrollLock ScrollLock S	Ctrl+Alt+Shift Zero Enter	Default: Ctrl Ctrl SG Enter	ScrollLock ScrollLock Zero
69	...key sequence to stop	Any key	Spacebar	Default: Ctrl Ctrl SH Enter	Spacebar
70	...button(s) to push to stop	SELECT	SELECT	SELECT	N.a.
71	...scanning interval	5 sec.; can be set to 5, 10, 15, 20 or 25 sec.	5 sec.; can be set to 5, 10, 25 or 40 sec.	5 sec.; can be set to 2-60 sec.	10 sec.
72	...disables keyboard?	Any key ends	Yes	No	Yes
73	...disables mouse?	No	No	No	Yes
74	...suspends so long as keyboard or mouse is active?	Yes, mouse only	No	Yes, but not by default for mouse	No
75	...does it work?	Yes	Yes	Yes	Yes
76	Controlled Scan capability?	No	No	No	No
77	...key sequence to start	N.a.	N.a.	N.a.	N.a.
78	...Next key sequence	N.a.	N.a.	N.a.	N.a.
79	...Previous key sequence	N.a.	N.a.	N.a.	N.a.
80	...key sequence to stop	N.a.	N.a.	N.a.	N.a.
81	...disables keyboard?	N.a.	N.a.	N.a.	N.a.
82	...disables mouse?	N.a.	N.a.	N.a.	N.a.
83	...does it work?	N.a.	N.a.	N.a.	N.a.
84	Caps Lock restored when switching	Yes	Yes	Yes	Yes
85	Num Lock restored when switching	Yes	Yes	Yes	Yes
86	Scroll Lock restored when switching	Yes	Yes	Yes	Yes

Documented support for:					
87	Microsoft Intellimouse	Yes	Yes	Yes	Not stated
88	Hot pluggable keyboard and mouse	Yes	Not stated	Yes	Not stated
89	Hot pluggable computers	Not stated	Not stated	Yes	Not stated
90	Maximum monitor resolution	2048x1536@75Hz	1920x1440	1600x1200@75Hz	2048x1536@85Hz
91	Video bandwidth	350MHz	250MHz	228MHz	400MHz
92	DDC2B compatible	Yes	Not stated	Yes	Not stated
93	Reset function	No	Power switch	Several, but only for mouse (commands)	No
94	Expand by connecting to another KVM	No	No	No	No

Figure 1a: Test results for CommandView, SwitchView, and OmniView (continued)

1*	**IOGEAR**	**IOGEAR**	**StarTech.com**	**StarTech.com**	**Tripp Lite**	**Tripp Lite**
2	**MiniView**	**MiniView**	**STARVIEW**	**STARVIEW**	**Tripp Lite**	**Tripp Lite**
3	GCS12	GCS14	SV211K	SV431	CS-82	CS-84
4	PPAG-4503-33A	PPAG-5701-33A			B005-002-R	B005-004-R
5	GCS12DP, GCS82A	GCS84A	SV211 SV201 SV411K SV231	SV431D SV231 SV401 SV411		B006-004-R
6	2	4	2	4	2	4
7	1	1	1	1	1	1
8	$130	$190	$90	$189	$125	$160
9	$64	$103	$51	$107	$68	$92
10	iogear.com	iogear.com	StarTech.com	StarTech.com	tripplite.com	tripplite.com
11	PS/2	PS/2	PS/2	PS/2	PS/2	PS/2
12	PS/2	PS/2	PS/2	PS/2	PS/2	PS/2
13	PS/2	PS/2	PS/2	PS/2	PS/2	PS/2
14	None	None	None	optional module	None	None
15	2x4ft.	4x4ft.	2x6ft.	None	None	None
16	6ft., $29.95 G2L1001P; 10ft., $39.95 G2L1003P	6ft., $29.95 G2L1001P; 10ft., $39.95 G2L1003P	N.a.	6–50ft. $19–$119 SVPS23N1_6 to SVPS23N1_50	6–15ft. $8.41-$17.51 P754-006 to P754-015	6–15ft. $8.41-$17.51 P754-006 to P754-015
17	Not stated	Not stated	6ft.	100ft.	Not stated	Not stated
18	No, KVM slides	No, KVM slides	Yes	No	Yes	Yes
19	No	No	Yes	No	No	No
20	Yes	Yes	No	Yes	Yes	Yes
21	Purple/green	Purple/green	Yes	Yes	Purple/green	Purple/green
22	Purple/green	Purple/green	Yes	No	Purple/green	Purple/green
23	Side	Side	Back	Back	Front	Front
24	Side	Side	Back	Back	Back	Back
25	Front and back	Front and back	Back	Back	Back	Back
26	One, switches between computers	One, switches between computers	One, switches between computers	Four, one for each computer	One, switches between computers	Four, one for each computer
27	Occasionally interprets 2 rapid pushes as 1	Must hit twice at power up	OK	OK	Rarely fails	OK
28	Two, one for each computer	Four, one for each computer	PC1 and PC2	Four (for each computer), each with a green and red bulb; Power	On Line and Selected for each computer	On Line and Selected for each computer

*Numbers refer to the features in Figure 1a.

Figure 1b: Test results for MiniView, STARVIEW and Tripp Lite (continues)

29	No	No	No	No	No	No
30	N.a.	N.a.	9v.	9v. 500ma.	N.a.	N.a.
31	N.a.	N.a.	No	Yes	N.a.	N.a.
32	Paper	Paper	Paper	Paper	Paper	Paper
33	No	No	Yes	No	Yes	Yes
34	4.2"x3.7"x1.3"	7"x3.75"x1.5"	6.2"x2.5"x0.8"	8.6"x5.3"x1.7"	5.1"x3"x1.6"	7.8"x3"x1.6"
35	107x97x35mm	178x95x37mm	20x70x155mm	220x136x43mm	129x77x40mm	199x77x40mm
36	5.2oz.	8.5oz.	8oz.	37oz.	16oz.	23.5oz.
37	146g.	240g.	228g.	1032g.	445g.	660g.
38	Rectangular with curved top	Rectangular with curved top	Rectangular	Rectangular	Rectangular	Rectangular
39	3 years	3 years	1 year	3 years	2 years	2 years
40	Varies	Ctrl+Alt+Shift	LeftCtrl LeftCtrl	LeftCtrl LeftCtrl	Varies	Alt Ctrl Shift
41	No	No	No	No	No	No
42	N.a.	N.a.	N.a.	N.a.	N.a.	N.a.
43	N.a.	N.a.	N.a.	N.a.	N.a.	N.a.
44	Sent; switches to other computer	OK	Sent, but next key ignored if typed soon	Sent, but next key ignored if typed soon	Sent; Switches to other computer	OK
45	Sent; switches to other computer	OK	Sent, but next key ignored if typed soon	Sent, but next key ignored if typed soon	Sent; Switches to other computer	OK
46	OK	OK	Sent, but next key ignored if typed soon	Sent, but next key ignored if typed soon	OK	OK
47	Sends LeftCtrls, switches to other computer and sends Tab to it	OK	Only sends LeftCtrls	Beeps and only sends LeftCtrls	Sends LeftCtrls, switches to other computer and sends Tab to it	OK
48	OK	OK	OK	OK	OK	OK
49	OK	OK	OK	OK	OK	OK
50	No	No	No	No	No	Keyboard not found
51	Yes	No	Yes	Yes	Yes	No
52	Yes	Yes	Yes	Yes	Yes	Yes
53	OK	OK	OK	OK	OK	OK
54	Correct	Correct	Correct	Correct	Correct	Correct
55	Incorrect	Incorrect	Incorrect	Correct	Incorrect	Incorrect
56	OK	OK	OK	OK	OK	OK

Figure 1b: Test results for MiniView, STARVIEW and Tripp Lite (continued)

57	OK	OK	OK	OK	OK	OK
58	No	Minimal	No	No	No	No
59	No	Minimal	No	No	No	Yes
60	No	Yes	No	No	No	Yes
61	Works	Works	Works	Works	Works	Works
62	Yes	Yes	Yes	Yes	Yes	Yes
63	N.a.	Ctrl+Alt+Shift n Enter	LeftCtrl LeftCtrl n	LeftCtrl LeftCtrl n	N.a.	Alt Ctrl Shift n Enter
64	Ctrl Ctrl	N.a.	N.a.	N.a.	Hit same Ctrl key twice	N.a.
65	N.a.	N.a.	N.a.	N.a.	N.a.	N.a.
66	Yes	Yes	Yes	Yes	Yes	Yes
67	No	No	No	Hold down buttons 3 and 4 together for 2 seconds	No	No
68	LeftShift RightShift	Ctrl+Alt+Shift Zero Enter	LeftCtrl LeftCtrl F1	LeftCtrl LeftCtrl F1	LeftShift RightShift	Alt Ctrl Shift Zero Enter
69	Spacebar	Spacebar	LeftCtrl LeftCtrl, but next keystroke is ignored	LeftCtrl LeftCtrl, but next keystroke is ignored	Spacebar	Spacebar
70	N.a.	N.a.	N.a.	N.a.	N.a.	N.a.
71	3 sec.	3 sec.	3 sec.; can be set to 3, 8, 15 or 30 sec.	3 sec.; can be set to 3, 8, 15 or 30 sec.	About 3 sec. though manual says 5 sec.	About 3 sec. though manual says 5 sec.
72	Yes	Yes	No	No	Yes	Yes
73	No	No	No	No	No	No
74	No	No	Yes	Yes	No	No
75	Yes	Yes	Yes	Yes	Yes	Yes
76	No	Yes	Yes	Yes	No	Yes
77	N.a.	Ctrl+Alt+Shift 9 Enter	LeftCtrl LeftCtrl F2	LeftCtrl LeftCtrl F2	N.a.	Alt Ctrl Shift 9 Enter
78	N.a.	RightShift	DownArrow	DownArrow	N.a.	RightShift
79	N.a.	LeftShift	UpArrow	UpArrow	N.a.	LeftShift
80	N.a.	Spacebar	Any key	Any key	N.a.	Spacebar
81	N.a.	Yes	Yes	Yes	N.a.	Yes
82	N.a.	No	No	No	N.a.	No
83	N.a.	Yes	Yes	Yes	N.a.	Yes
84	Yes	Yes	Yes	Yes	Yes	Yes

Figure 1b: Test results for MiniView, STARVIEW and Tripp Lite (continued)

85	Yes	Yes	Yes	Yes	Yes	Yes
86	Yes	Yes	Yes	Yes	Yes	Yes
Documented support for:						
87	Yes	Yes	Yes	Yes	Yes	Yes
88	Yes	Yes	Yes	Yes	No	No
89	Yes	Yes	Yes	Yes	No	No
90	1920x1440	1920x1440	1920x1440	1920x1440	1920x1440	1920x1440
91	Not stated	Not stated	Not stated	300MHz	180MHz	180MHz
92	Yes	Yes	Yes	Yes	Yes	Yes
93	No	No	None	Hold down buttons 1 and 2 together for 2 seconds	None	None
94	No	No	No	Yes	No	No

Figure 1b: Test results for MiniView, STARVIEW and Tripp Lite (continued)

- ATEN – now IOGEAR
- Avocent – http://www.avocent.com, select Products
- Belkin – http://www.belkin.com, select KVM
- Cables To Go – http://www.cablestogo.com, KVMs/Switchboxes
- Compucable – now Addlogix
- Cybex – now Avocent
- D-Link – http://www.dlink.com/products/category.asp?cid=6
- Hawking – http://www.hawkingtech.com/prodFam.php?CatId=22
- IOGEAR – http://iogear.com/main.php?loc=productcategory&category=kvm
- Lightwave – http://www.lightwavecom.com/products/rkvm/slinx_slk
- LinkSys – http://www.linksys.com/products, select KVM Switches
- Minicom – http://www.minicom.com/kvmswitches.htm
- NTI – http://www.networktechinc.com/servswt.html
- PC Concepts – http://www.pcconcepts.com/html/cab_switch.html
- Raritan – http://www.raritan.com/products/kvm_switches/prd_category.aspx
- Rose Electronics – http://www.rose.com/htm/kvmswitches.htm
- StarTech – http://www.startech.com/kvmswitches/kvmswitches.htm
- SwitchCom – http://www.switchcom.co.uk

- Tripp Lite – http://www.tripplite.com/products/cables/kvm.cfm.

CONCLUSIONS

During testing, it quickly became apparent that manufacturers had not tested their KVMs with 3270 emulators, at least not in the manner that many mainframe users use them. In fact, some product managers were quite interested in my test results, some going as far as planning changes for future models.

KVMs, at least the two- and four-port models examined here, have their limitations. If you look carefully at the test results, you'll see that only one KVM correctly handled a power up/boot/start-up of a workstation when the monitor was in use for the entire period by another workstation. All the rest returned the maximum monitor resolution and screen refresh rate handled by the KVM, rather than the maximum handled by the monitor.

This makes it easy to inadvertently set the monitor resolution above what the monitor is capable of. But there's another implication that had me scratching my head for a day or two. Like me, you have probably been unaware that increasing the monitor resolution to a point where your monitor can no longer handle the current screen refresh rate will result in Windows automatically decreasing the rate of screen refresh. But this happens only if Windows has the correct values for your monitor's maximum screen refresh rate for each monitor resolution.

I have yet to find an affordable two-port KVM with the functionality I've been looking for. I want to view both workstations simultaneously, using two monitors and one keyboard and mouse, just as a technical writer would, viewing a mainframe screen while writing documentation with work processing on another workstation.

Ergonomically, I want the selected computer on the monitor right in front of me and the ‘other’ computer on the monitor to the side. And I wanted to be able to switch with the touch of a single button. Recently, there were rumours of high-end KVMs with this capability, so maybe it will eventually filter down to the low-end.

Jon E Pearkins
Adiant (Canada)

IBM’s Communications Server for Linux Version 6.2

Communications Server for Linux, announced in May 2004 with general availability as of June 2004, is now the latest, and by far the most intriguing, member in IBM’s extremely ‘blue blooded’ Communications Server family, which has now been around for nearly 20 years – having started off as an OS/2 component. Ironically, one of the major roles envisaged for this new version is to provide, via an inspired Remote Client API capability, an attractive migration path for customers with OS/2-based SNA clients that now want to port these ‘legacy’ client applications, unchanged, to Windows or Linux.

Communications Server for Linux is currently available in two distinct versions – z/Linux and Linux for Intel. Thus the feature-packed Communications Server (CS) ‘gateway’ package, with its legendary tn2370(E) gateway, EE functionality, APPN support, and DLUR requestor, is now available on Windows (2000, 2003, and XP), AIX, z/OS (where it also includes the pivotal ACF/VTAM component), z/Linux, and Linux for Intel. Note, however, that Communications Server’s Linux support does not extend to IBM’s highly strategic (and mid-range converging) POWER-based servers, which now include the new i5s (within the iSeries), the p5s (within the pSeries), as well as the recently-announced, natively Linux, IBM OpenPower 720.

When it comes to CS on POWER, one has to remember that

Communications Server, despite its complexity, in the end is still just a Linux application. While one obviously has to port Linux to a specific platform, in this instance POWER, once Linux is available on a platform, supporting an application on top of Linux is just a case of testing and certification. And this is where with Linux, as illustrated here, we are once again running into the frustrating conundrum encountered with Java Web-to-host offerings, ie write once but needing to be diligently tested on every platform. The good news, however, is that IBM is acutely aware of this omission and intends to rectify it. The other good news is that IBM has added the z/Linux version, at no additional cost, to the WebSphere Host Integration Solution (HIS) bundle – this is, therefore, the first time a mainframe gateway has been available with HIS. Current HIS customers have the option of obtaining it, again at no charge, from IBM, though you may want to read the rest of this article before jumping at this offer.

IT IS NOT QUITE AS WONDERFUL AS YOU MIGHT EXPECT

Deciding how one can gainfully use CS/Linux, which at first blush looks highly compelling, may, however, not be as clear cut and obvious as one might initially think. For a start, the z/Linux offering might end up being considerably more germane to VSE/ESA and z/VM customers than to those running z/OS. The crux of the issue here, as this readership will immediately appreciate, has to do with cross-LPAR networking. As we know, mainframe networking is now optimized for IP-based networking; with HiperSockets, the fastest means of realizing inter-LPAR communications, being totally IP-centric. It is this bias towards IP-based communications that makes it difficult to justify where and how to use these Linux offerings.

In the end, as has always been the case when it comes to mainframe SNA, everything revolves around ACF/VTAM. CS for z/Linux, though including a plethora of APIs, among which are CPI-C, HACL for Java (*à la* Host On-Demand), LU-level session APIs, and many APPC interfaces, does not include ACF/VTAM functionality. VTAM functionality is available only on z/OS (via CS), VSE, and VM. Thus, as with other versions of CS since around 1997 including CS for z/OS, one of the key attractions of CS/Linux, whether on the mainframe or an Intel server, is going to be the powerful tn3270(E) server with its integrated support for SSL security – critical for nearly all contemporary Web-to-host solutions whether it be applet-based emulation (eg IBM's HoD), host publishing (eg IBM's HATS V5) or host integration (eg NetManage's OnWeb 7.1). And that is where the dilemma begins.

The I/O configuration of a 'tn' server is very straightforward and immutable. It 'speaks' TCP/IP on the network side (ie downstream) and has to speak SNA(/APPN) on the mainframe side. Thus if you want to use CS for z/Linux as a scalable, cost-effective 'tn' server you have to have a way to 'bridge' the SNA/ APPN (on the upstream side) with the LPAR running ACF/ VTAM (and with it, all the major mission-critical applications of the CICS, IMS, DB2 ilk). Technically this is not a problem, given that there are multiple proven means for cross-LPAR communications including real or virtual Channel-to-Channel (CTC) connections, as well as shared LANs using OSA(-Express). Note that HiperSockets, however, is not a direct option – unless, of course, you want to add the extra step (and overhead) of using EE between the LPARs. The issue thus boils down to performance and efficiency, particularly when it comes to z/OS – given that it is possible to have the 'tn' server implemented within the z/OS LAPR, which eliminates the need for a cross-LPAR 'hop'.

DEFINITE RECOMMENDATIONS FOR VSE AND VM CUSTOMERS

Since there is no native CS offering for VSE and VM (and hence no IBM-provided 'tn' server or EE), the dynamics and decisions are very different from those faced by z/OS customers. CS for z/Linux provides VSE and VM customers with all the advantages (eg scalability and resilience) of a mainframe resident 'tn' server, EE, and APPN node capability – albeit on a 'surrogate' LPAR. VM and VSE customers that need a high-capacity, high-performance tn3270(E) server (ie

250 or more concurrent sessions with sub-5-second performance) should seriously consider the z/Linux option. With this option they can gainfully consolidate multiple networking essentials, including their Web server, tn3270(E) server, host publishing/host integration server, on a single mainframe LPAR, and most likely use virtual CTC for their inter-LPAR networking.

The issues are more convoluted when it comes to z/OS. Consolidating various networking components (eg Web servers, application servers, Web-to-host schemes) hitherto running on external servers on a single mainframe LPAR is a very strategic move given that it simplifies complexity and ensures painless scalability. In this context it is also fair to say that having a ‘tn’ server on z/Linux is likely to be a better option for most medium to large z/OS shops than having it on an external server – particularly a Windows server. But then the question becomes, why implement it on a z/Linux LPAR when it could be installed, co-located with VTAM (given that CS is already installed on the z/OS LPAR in order to obtain the VTAM functionality)?

To this end it is also important to factor in that IBM does not recommend that customers who already have a z/OS tn3270(E) server think about migrating that server to z/Linux. IBM cites performance, robustness, and the exploitation of z/OS features such as Sysplex and VIPA as reasons why they recommend sticking with z/OS rather than moving to z/Linux. That in itself should give you some valuable guidance as to what you should be thinking about when weighting ‘tn’ on z/OS versus z/Linux.

BOTTOM LINE

CS for Linux is certainly a welcome addition to the strategic CS family – though the lack of support, at least at present, for Linux on POWER is frustrating. CS for Linux is a compelling migration option for customers with OS/2 SNA applications as well as those that are using Windows-based SNA Gateways (eg Microsoft's Host Integration Server). It is also an extremely attractive proposition for VSE and VM customers. z/OS customers, however, may not find it that appealing. But, again, they should evaluate it carefully to see whether there are any cost savings they might be able to realize – albeit by sacrificing some performance and efficiency.

Anura Gurugé
Strategic Consultant (USA)

@FTPPUT edit macro

Moving source back and forth between ISPF and distributed platforms can be a time-consuming activity. So I wrote a pair of edit macros to make my life a little easier. In the September issue of *TCP/SNA Update* we looked @FTPGET (see Issue 55); this issue we look at @FTPPUT.

@FTPPUT will take the current contents of an edit session and PUT it to any FTP server. Like @FTPGET, it presents a small pop-up to get the destination information and sends the data.

@FTPPUT will capture the FTP output and present it in browse, only if the FTP session has a problem. The macro is fully self-contained and needs to be placed in a PDS in the TSO SYSEXEC concatenation. It could also be placed in SYSPROC, but you will have to delete the first line so you have the required REXX comment on the first line.

FTPPUT

```
/**********************************************************************/
/*                               REXX                                 */
/**********************************************************************/
/* Purpose: Edit macro to FTP PUT the contents of the edit session    */
/*--------------------------------------------------------------------*/
```

```
/* Syntax:  @FTPPUT ftpdir ftpaddr                                   */
/*-------------------------------------------------------------------*/
/* Parms: ftpdir     - Destination directory on target host          */
/*        ftpaddr    - Target host address for receiving the file    */
/*                                                                   */
/*********************************************************************/
/*                      Change Log                                   */
/*********** @REFRESH BEGIN START    2004/03/06 13:16:32 *************/
/* Standard housekeeping activities                                  */
/*********************************************************************/
call time 'r'
parse arg parms
signal on syntax name trap
signal on failure name trap
signal on novalue name trap
probe = 'NONE'
modtrace = 'NO'
modspace = ''
call stdentry 'DIAGMSGS'
module = 'MAINLINE'
push trace() time('L') module 'From:' 0 'Parms:' parms
if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
call modtrace 'START' 0
/*********************************************************************/
/* Set local esoteric names                                          */
/*********************************************************************/
@vio   = 'VIO'
@sysda = 'SYSDA'
/*********** @REFRESH END   START    2004/03/06 13:16:32 *************/
/* Establish Edit Macro environment, Get parms and get dsn and mem   */
/*********************************************************************/
call isrwrap "MACRO (FTPDIR FTPADDR)"
call isrwrap "(DSN) = DATASET"
call isrwrap "(FILE) = MEMBER"
call isrwrap "(LRECL) = LRECL"
/*********************************************************************/
/* Find the last line and create a stem variable for each line       */
/*********************************************************************/
call isrwrap "CURSOR = .ZLAST"
call isrwrap "(LAST) = CURSOR"
do i=1 to last
    call isrwrap "(LINE) = LINE" i
    original.i = line
end
call isrwrap "UP MAX"
/*********************************************************************/
/* Put the FTPDIR and FTPADDR in the profile pool                    */
/*********************************************************************/
if ftpdir <> '' then call ispwrap "VPUT (FTPDIR) PROFILE"
if ftpaddr <> '' then call ispwrap "VPUT (FTPADDR) PROFILE"
```

```
/*********************************************************************/
/* Check to see whether the profile variables exist                  */
/*********************************************************************/
call ispwrap 8 "VGET (FTPDIR) PROFILE"
call ispwrap 8 "VGET (FTPADDR) PROFILE"
/*********************************************************************/
/* If no dsn or member exists, see if this is an HFS file            */
/*********************************************************************/
hfsname = ''
if dsn = '' & file = '' then call ispwrap 8 "VGET (HFSNAME)"
if hfsname <> '' & hfsname <> 'HFSNAME' then file = hfsname
/*********************************************************************/
/* If this is a sequential file set the filename to the dsn          */
/*********************************************************************/
if dsn <> '' & file = '' then file = dsn
if file = '' then file = 'TEMP'
/*********************************************************************/
/* Set a bogus DDNAME in case the user PF3s out                      */
/*********************************************************************/
ddname = '#BOGUS#'
/*********************************************************************/
/* Build a temporary panel                                           */
/*********************************************************************/
ftp.1  = ")ATTR                                             "
ftp.2  = " @ TYPE(DT)                                       "
ftp.3  = " # TYPE(INPUT) CAPS(OFF) COLOR(GREEN)             "
ftp.4  = " $ TYPE(INPUT) CAPS(OFF) INTENS(NON)              "
ftp.5  = ")BODY EXPAND(//) WINDOW(5Ø,7)                     "
ftp.6  = "+FTP Site can be full IP address or DNS Name      "
ftp.7  = "@Enter FTP Site ID  : #Z                          "
ftp.8  = "@Enter FTP UserID   : #Z        +                 "
ftp.9  = "@Enter FTP Password : $Z       +                  "
ftp.1Ø = "@Enter FTP Directory: #Z                          "
ftp.11 = "@Optional Suffix    : _Z   +(like .TXT - optional)"
ftp.12 = "@Trailing Blanks    : _Z   +(%YES+or%NO+)         "
ftp.13 = ")INIT                                             "
ftp.14 = " .ZVARS = '(FTPADDR FTPUSER FTPPASS FTPDIR +      "
ftp.15 = "            FTPSUFF FTPBLANK)'                    "
ftp.16 = " IF (&FTPUSER = &Z)                               "
ftp.17 = "     &FTPUSER = 'anonymous'                       "
ftp.18 = " IF (&FTPPASS = &Z)                               "
ftp.19 = "     &FTPPASS = &ZUSER                            "
ftp.2Ø = " IF (&FTPDIR = &Z)                                "
ftp.21 = "     &FTPDIR = '/'                                "
ftp.22 = " IF (&FTPBLANK = &Z)                              "
ftp.23 = "     &FTPBLANK = 'NO'                             "
ftp.24 = ")PROC                                             "
ftp.25 = " VER (&FTPADDR,NB)                                "
ftp.26 = " VER (&FTPUSER,NB)                                "
ftp.27 = " VER (&FTPPASS,NB)                                "
ftp.28 = " VER (&FTPDIR,NB)                                 "
ftp.29 = " IF  (&FTPSUFF = &Z)                               "
ftp.3Ø = "      &FTPSUFF = '$NONE$'                          "
ftp.31 = " VER (&FTPBLANK,NB,LIST,YES,NO)                    "
ftp.32 = ")END                                               "
/**********************************************************************/
/* Display the Dynamic Panel                                          */
/**********************************************************************/
call popdyn 'FTP' 4 'Enter' execname 'Parameters for' file
/**********************************************************************/
/* VPUT the FTPFILE and FTPADDR                                       */
/**********************************************************************/
call ispwrap "VPUT (FTPDIR) PROFILE"
call ispwrap "VPUT (FTPADDR) PROFILE"
/**********************************************************************/
/* Allocate a temporary dataset to hold the copy                      */
/**********************************************************************/
ddname = '@FTP'random()
dsname = userid()'.'ddname'.'strip(left(file,8))
call tsotrap "ALLOC F("ddname") DA('"dsname"') NEW REUSE UNIT(SYSDA)",
              "SPACE(1 1) CYL LRECL("lrecl") BLKSIZE(Ø) RECFM(F B)"
/**********************************************************************/
/* Copy the lines to the temporary dataset                            */
/**********************************************************************/
call tsotrap 1 "EXECIO * DISKW" ddname" (FINIS STEM ORIGINAL."
/**********************************************************************/
/* Determine if this is a Unix file or an MVS dataset                 */
/**********************************************************************/
if substr(ftpdir,1,1) = '/' then
    do
/**********************************************************************/
/* If Unix, format the path for the PUT                               */
/**********************************************************************/
     ftptype = 'FILE'
     lastchar = substr(reverse(ftpdir),1,1)
     if lastchar <> '/' & lastchar <> '\' then ftpdir = ftpdir||'/'
     if ftpsuff = '$NONE$' then
        path = ftpdir||file
     else
        path = ftpdir||file||ftpsuff
    end
else
/**********************************************************************/
/* If MVS DSN, determine whether this is a PDS member                 */
/**********************************************************************/
    do
     if pos('(',ftpdir) <> Ø then
        do
         ftptype = 'MEMBER'
         parse var ftpdir ftpdsn '(' ftpmem ')'
        end
     else
        do
         ftptype = 'DSN'
         ftpdsn = ftpdir
        end
    end
/*********************************************************************/
/* Format the FTP commands in the stem variable based on screen input*/
/*********************************************************************/
input.1 = ftpuser ftppass
if ftpblank = 'YES' then
    input.2 = "SIte TRAIL"
else
    input.2 = "SIte NOTRAIL"
select
    when ftptype = 'DSN' then
         do
          input.3 = "put" "'"dsname"' '"ftpdsn"'"
          input.4 = "quit"
          path = ftpdir
         end
    when ftptype = 'MEMBER' then
         do
          input.3 = "cd '"ftpdsn"'"
          input.4 = "put" "'"dsname"'" ftpmem
          input.5 = "quit"
          path = ftpdir
         end
    otherwise
         do
          input.3 = "put" "'"dsname"'" path
          input.4 = "quit"
         end
end
/*********************************************************************/
/* Lock the screen                                                   */
/*********************************************************************/
if ftptype = 'MEMBER' then
    call lock input.4
else
    call lock input.3
/*********************************************************************/
/* Copy the lines to the temporary INPUT dataset                     */
/*********************************************************************/
call viodd 'INPUT'
/*********************************************************************/
/* Allocate a temporary dataset to hold FTP OUTPUT                   */
/*********************************************************************/
call tsotrap "ALLOC F(OUTPUT) NEW REUSE UNIT(SYSDA) SPACE(1 1) CYL"
```

```
/**********************************************************************/
/* FTP the DSN                                                        */
/**********************************************************************/
address TSO "FTP" ftpaddr "(EXIT"
FTPRC = RC
call unlock
/**********************************************************************/
/* If there is an FTP error, browse the OUTPUT dataset                */
/**********************************************************************/
if FTPRC <> 0 then
    do
     call msg 'FTP error, RC='FTPRC 'Review the FTP output'
     call brwsdd 'OUTPUT'
     call tsotrap "FREE F(INPUT)"
     call rcexit FTPRC 'FTP from' ftpaddr 'failed'
    end
else
    do
     zedlmsg = path 'successfully loaded to' ftpaddr
     call ispwrap "SETMSG MSG(ISRZ000) COND"
    end
/**********************************************************************/
/* Clean up temporary FTP INPUT and OUTPUT files                      */
/**********************************************************************/
call tsotrap "FREE F(INPUT)"
call tsotrap "FREE F(OUTPUT)"
/**********************************************************************/
/* Shutdown                                                           */
/**********************************************************************/
shutdown: nop
/**********************************************************************/
/* Shutdown/Clean up processing                                       */
/**********************************************************************/
           if isr_verb <> 'MACRO' & ERC <> 20 then
              do
               call outtrap 'trash',0
               address TSO "FREE F("ddname") DELETE"
              end
           EXITRC = 1
/*********** @REFRESH BEGIN STOP      2002/08/03 08:42:33 *************/
/* Shutdown message and terminate                                     */
/**********************************************************************/
           call stdexit time('e')
/*********** @REFRESH END   STOP      2002/08/03 08:42:33 *************/
/*********** @REFRESH BEGIN SUBBOX    2004/03/10 01:25:03 *************/
/*                                                                    */
/* 28 Internal Subroutines provided in @FTPPUT                        */
/*                                                                    */
/* Last Subroutine REFRESH was 16 Apr 2004 19:47:20                   */
/*                                                                    */
/*                                                                   */
/* RCEXIT   - Exit on non-zero return codes                          */
/* TRAP     - Issue a common trap error message using rcexit         */
/* ERRMSG   - Build common error message with failing line number    */
/* STDENTRY - Standard Entry logic                                   */
/* STDEXIT  - Standard Exit logic                                    */
/* MSG      - Determine whether to SAY or ISPEXEC SETMSG the message */
/* DDCHECK  - Determine whether a required DD is allocated           */
/* QDSN     - Make sure there is only one set of quotes              */
/* TEMPMEM  - EXECIO a stem into a TEMP PDS                          */
/* BRWSDD   - Invoke ISPF Browse on any DD                           */
/* ISPWRAP  - Wrapper for ISPF commands                              */
/* ISRWRAP  - Wrapper for ISPF Edit commands                         */
/* TSOTRAP  - Capture the output from a TSO command in a stem        */
/* WAIT     - Wait for a specified number of seconds                 */
/* SETBORD  - Set the ISPF Pop-up active frame border colour         */
/* LOCK     - Put up a pop-up under foreground ISPF during long waits*/
/* UNLOCK   - Unlock from a pop-up under foreground ISPF             */
/* PANDSN   - Create a unique PDS(MEM) name for a dynamic panel      */
/* POPDYN   - Addpop a Dynamic Panel                                 */
/* SAYDD    - Print messages to the requested DD                     */
/* JOBINFO  - Get job-related data from control blocks               */
/* PTR      - Pointer to a storage location                          */
/* STG      - Return the data from a storage location                */
/* VIODD    - EXECIO a stem into a sequential dataset                */
/* MODTRACE - Module Trace                                           */
/*                                                                   */
/*********** @REFRESH END   SUBBOX   2004/03/10 01:25:03 *************/
/*********** @REFRESH BEGIN RCEXIT   2003/05/14 12:24:50 *************/
/* RCEXIT   - Exit on non-zero return codes                          */
/*-------------------------------------------------------------------*/
/* EXITRC   - Return code to exit with (if non zero)                 */
/* ZEDLMSG  - Message text for it for non-zero EXITRCs               */
/*********************************************************************/
rcexit: parse arg EXITRC zedlmsg
         if EXITRC <> 0 then
            do
             trace 'o'
/*********************************************************************/
/* If execution environment is ISPF then VPUT ZISPFRC                */
/*********************************************************************/
             if execenv = 'TSO' | execenv = 'ISPF' then
                do
                 if ispfenv = 'YES' then
                    do
                     zispfrc = EXITRC
/*********************************************************************/
/* Does not call ISPWRAP to avoid obscuring error message modules    */
/*********************************************************************/
                     address ISPEXEC "VPUT (ZISPFRC)"
                    end
                end
/*********************************************************************/
/* If a message is provided, wrap it in date, time, and EXITRC       */
/*********************************************************************/
             if zedlmsg <> '' then
                do
                 zedlmsg = time('L') execname zedlmsg 'RC='EXITRC
                 call msg zedlmsg
                end
/*********************************************************************/
/* Write the contents of the Parentage Stack                         */
/*********************************************************************/
             stacktitle = 'Parentage Stack Trace ('queued()' entries):'
/*********************************************************************/
/* Write to MSGDD if background and MSGDD exists                     */
/*********************************************************************/
             if tsoenv = 'BACK' then
                do
                 if subword(zedlmsg,9,1) = msgdd then
                    do
                     say zedlmsg
                     signal shutdown
                    end
                 else
                    do
                     call saydd msgdd 1 zedlmsg
                     call saydd msgdd 1 stacktitle
                    end
                end
             else
/*********************************************************************/
/* Write to the ISPF Log if foreground                               */
/*********************************************************************/
                do
                 zerrlm = zedlmsg
                 address ISPEXEC "LOG MSG(ISRZ003)"
                 zerrlm = center(' 'stacktitle' ',78,'-')
                 address ISPEXEC "LOG MSG(ISRZ003)"
                end
/*********************************************************************/
/* Unload the Parentage Stack                                        */
/*********************************************************************/
             do queued()
                pull stackinfo
                if tsoenv = 'BACK' then
                   do
                    call saydd msgdd 0 stackinfo
                   end
                else
```

```
                   do
                    zerrlm = stackinfo
                    address ISPEXEC "LOG MSG(ISRZ003)"
                   end
             end
/**********************************************************************/
/* Put a terminator in the ISPF Log for the Parentage Stack           */
/**********************************************************************/
             if tsoenv = 'FORE' then
                do
                 zerrlm = center(' 'stacktitle' ',78,'-')
                 address ISPEXEC "LOG MSG(ISRZ003)"
                end
/**********************************************************************/
/* Signal SHUTDOWN.  SHUTDOWN label MUST exist in the program         */
/**********************************************************************/
             signal shutdown
            end
         else
            return
/*********** @REFRESH END   RCEXIT   2003/05/14 12:24:50 *************/
/*********** @REFRESH BEGIN TRAP     2002/08/07 11:48:14 *************/
/* TRAP     - Issue a common trap error message using rcexit          */
/*--------------------------------------------------------------------*/
/* PARM     - N/A                                                     */
/**********************************************************************/
trap: traptype = condition('C')
       if traptype = 'SYNTAX' then
          msg = errortext(RC)
       else
          msg = condition('D')
       trapline = strip(sourceline(sigl))
       msg = traptype 'TRAP:' msg', Line:' sigl '"'trapline'"'
       call rcexit 666 msg
/*********** @REFRESH END   TRAP     2002/08/07 11:48:14 *************/
/*********** @REFRESH BEGIN ERRMSG   2002/08/10 16:53:04 *************/
/* ERRMSG   - Build common error message with failing line number     */
/*--------------------------------------------------------------------*/
/* ERRLINE  - The failing line number passed by caller from SIGL      */
/* TEXT     - Error message text passed by caller                     */
/**********************************************************************/
errmsg: nop
         parse arg errline text
         return 'Error on statement' errline',' text
/*********** @REFRESH END   ERRMSG   2002/08/10 16:53:04 *************/
/*********** @REFRESH BEGIN STDENTRY 2004/04/07 19:17:48 *************/
/* STDENTRY - Standard Entry logic                                    */
/*--------------------------------------------------------------------*/
/* MSGDD    - Optional MSGDD used only in background                  */
/**********************************************************************/
```

```
stdentry: module = 'STDENTRY'
           if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
           parse arg sparms
           push trace() time('L') module 'From:' sigl 'Parms:' sparms
           arg msgdd
           parse upper source . . execname . execdsn . . execenv .
/*********************************************************************/
/* Start-up values                                                   */
/*********************************************************************/
           EXITRC = 0
           MAXRC = 0
           ispfenv = 'NO'
           popup = 'NO'
           lockpop = 'NO'
           headoff = 'NO'
           hcreator = 'NO'
           keepstack = 'NO'
           lpar = mvsvar('SYSNAME')
           zedlmsg = 'Default shutdown message'
/*********************************************************************/
/* Determine environment                                             */
/*********************************************************************/
           if substr(execenv,1,3) <> 'TSO' & execenv <> 'ISPF' then
              tsoenv = 'NONE'
           else
              do
               tsoenv = sysvar('SYSENV')
               signal off failure
              "ISPQRY"
               ISPRC = RC
               if ISPRC = 0 then
                  do
                   ispfenv = 'YES'
/*********************************************************************/
/* Check if HEADING ISPF table exists already, if so set HEADOFF=YES */
/*********************************************************************/
                   call ispwrap "VGET (ZSCREEN)"
                   if tsoenv = 'BACK' then
                      htable = jobinfo(1)||jobinfo(2)
                   else
                      htable = userid()||zscreen
                   TBCRC = ispwrap(8 "TBCREATE" htable "KEYS(HEAD)")
                   if TBCRC = 0 then
                      do
                       headoff = 'NO'
                       hcreator = 'YES'
                      end
                   else
                      do
                       headoff = 'YES'
                      end
                  end
               signal on failure name trap
              end
/*********************************************************************/
/* MODTRACE must occur after the setting of ISPFENV                  */
/*********************************************************************/
           call modtrace 'START' sigl
/*********************************************************************/
/* Start-up message (if batch)                                       */
/*********************************************************************/
           startmsg = execname 'started' date() time() 'on' lpar
           if tsoenv = 'BACK' & sysvar('SYSNEST') = 'NO' &,
              headoff = 'NO' then
              do
               jobname = mvsvar('SYMDEF','JOBNAME')
               jobinfo = jobinfo()
               parse var jobinfo jobtype jobnum .
               say jobname center(' 'startmsg' ',61,'-') jobtype jobnum
               say
               if ISPRC = -3 then
                  do
                   call saydd msgdd 1 'ISPF ISPQRY module not found,',
                                      'ISPQRY is usually in the LINKLST'
                   call rcexit 20 'ISPF ISPQRY module is missing'
                  end
/*********************************************************************/
/* If MSGDD is provided, write the STARTMSG and SYSEXEC DSN to MSGDD */
/*********************************************************************/
               if msgdd <> '' then
                  do
                   call ddcheck msgdd
                   call saydd msgdd 1 startmsg
                   call ddcheck 'SYSEXEC'
                   call saydd msgdd 0 execname 'loaded from' sysdsname
/*********************************************************************/
/* If there are PARMS, write them to the MSGDD                       */
/*********************************************************************/
                   if parms <> '' then
                      call saydd msgdd 0 'Parms:' parms
/*********************************************************************/
/* If there is a STEPLIB, write the STEPLIB DSN MSGDD                */
/*********************************************************************/
                   if listdsi('STEPLIB' 'FILE') = 0 then
                      do
                       steplibs = dddsns('STEPLIB')
                       call saydd msgdd 0 'STEPLIB executables loaded',
                           'from' word(dddsns,1)
                       if dddsns('STEPLIB') > 1 then
                          do
```

```
                           do stl=2 to steplibs
                              call saydd msgdd Ø copies(' ',31),
                                   word(dddsns,stl)
                           end
                          end
                      end
                  end
              end
/**********************************************************************/
/* If foreground, save ZFKA and turn off the FKA display              */
/**********************************************************************/
           else
              do
               fkaset = 'OFF'
               call ispwrap "VGET (ZFKA) PROFILE"
               if zfka <> 'OFF' & tsoenv = 'FORE' then
                  do
                   fkaset = zfka
                   fkacmd = 'FKA OFF'
                   call ispwrap "CONTROL DISPLAY SAVE"
                 call ispwrap "DISPLAY PANEL(ISPBLANK) COMMAND(FKACMD)"
                   call ispwrap "CONTROL DISPLAY RESTORE"
                  end
              end
/**********************************************************************/
           pull tracelvl . module . sigl . sparms
           call modtrace 'STOP' sigl
           interpret 'trace' tracelvl
           return
/*********** @REFRESH END   STDENTRY 2ØØ4/Ø4/Ø7 19:17:48 *************/
/*********** @REFRESH BEGIN STDEXIT  2ØØ3/11/16 22:46:29 *************/
/* STDEXIT  - Standard Exit logic                                     */
/*--------------------------------------------------------------------*/
/* ENDTIME  - Elapsed time                                            */
/* Note: Caller must set KEEPSTACK if the stack is valid              */
/**********************************************************************/
stdexit: module = 'STDEXIT'
          if wordpos(module,probe) <> Ø then trace 'r'; else trace 'n'
          parse arg sparms
          push trace() time('L') module 'From:' sigl 'Parms:' sparms
          call modtrace 'START' sigl
          arg endtime
          endmsg = execname 'ended' date() time() format(endtime,,1)
/**********************************************************************/
/* if MAXRC is greater than EXITRC then set EXITRC to MAXRC           */
/**********************************************************************/
          if MAXRC > EXITRC then EXITRC = MAXRC
          endmsg = endmsg 'on' lpar 'RC='EXITRC
          if tsoenv = 'BACK' & sysvar('SYSNEST') = 'NO' &,
             headoff = 'NO' then
             do
              say
              say jobname center(' 'endmsg' ',61,'-') jobtype jobnum
/**********************************************************************/
/* Make sure this isn't a MSGDD missing error then log to MSGDD       */
/**********************************************************************/
              if msgdd <> '' & subword(zedlmsg,9,1) <> msgdd then
                 do
                  call saydd msgdd 1 execname 'ran in' endtime 'seconds'
                  call saydd msgdd 0 endmsg
                 end
             end
/**********************************************************************/
/* If foreground, reset the FKA if necessary                          */
/**********************************************************************/
          else
             do
              if fkaset <> 'OFF' then
                 do
                  fkafix = 'FKA'
                  call ispwrap "CONTROL DISPLAY SAVE"
                  call ispwrap "DISPLAY PANEL(ISPBLANK) COMMAND(FKAFIX)"
                  if fkaset = 'SHORT' then
                     call ispwrap "DISPLAY PANEL(ISPBLANK)",
                                  "COMMAND(FKAFIX)"
                  call ispwrap "CONTROL DISPLAY RESTORE"
                 end
             end
/**********************************************************************/
/* Clean up the temporary HEADING table                               */
/**********************************************************************/
          if ispfenv = 'YES' & hcreator = 'YES' then
             call ispwrap "TBEND" htable
/**********************************************************************/
/* Remove STDEXIT and MAINLINE Parentage Stack entries, if there      */
/**********************************************************************/
          call modtrace 'STOP' sigl
          if queued() > 0 then pull . . module . sigl . sparms
          if queued() > 0 then pull . . module . sigl . sparms
          if tsoenv = 'FORE' & queued() > 0 & keepstack = 'NO' then
             pull . . module . sigl . sparms
/**********************************************************************/
/* if the Parentage Stack is not empty, display its contents          */
/**********************************************************************/
          if queued() > 0 & keepstack = 'NO' then
             do
              say queued() 'Leftover Parentage Stack Entries:'
              say
              do queued()
                 pull stackundo
                 say stackundo
              end
              EXITRC = 1
             end
/*********************************************************************/
/* Exit                                                              */
/*********************************************************************/
          exit(EXITRC)
/*********** @REFRESH END   STDEXIT  2003/11/16 22:46:29 *************/
/*********** @REFRESH BEGIN MSG      2002/09/11 01:35:53 *************/
/* MSG      - Determine whether to SAY or ISPEXEC SETMSG the message */
/*-------------------------------------------------------------------*/
/* ZEDLMSG  - The long message variable                              */
/*********************************************************************/
msg: module = 'MSG'
      parse arg zedlmsg
      if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
      parse arg sparms
      push trace() time('L') module 'From:' sigl 'Parms:' sparms
      call modtrace 'START' sigl
/*********************************************************************/
/* If this is background or OMVS use SAY                             */
/*********************************************************************/
      if tsoenv = 'BACK' | execenv = 'OMVS' then
         say zedlmsg
      else
/*********************************************************************/
/* If this is foreground and ISPF is available, use SETMSG           */
/*********************************************************************/
         do
          if ispfenv = 'YES' then
/*********************************************************************/
/* Does not call ISPWRAP to avoid obscuring error message modules    */
/*********************************************************************/
             address ISPEXEC "SETMSG MSG(ISRZ000)"
          else
             say zedlmsg
         end
      pull tracelvl . module . sigl . sparms
      call modtrace 'STOP' sigl
      interpret 'trace' tracelvl
      return
/*********** @REFRESH END   MSG      2002/09/11 01:35:53 *************/
/*********** @REFRESH BEGIN DDCHECK  2002/09/11 01:08:30 *************/
/* DDCHECK  - Determine whether a required DD is allocated           */
/*-------------------------------------------------------------------*/
/* DD       - DDNAME to confirm                                      */
/*********************************************************************/
ddcheck: module = 'DDCHECK'
          if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
          parse arg sparms
          push trace() time('L') module 'From:' sigl 'Parms:' sparms
          call modtrace 'START' sigl
          arg dd
          dderrmsg = 'OK'
          LRC = listdsi(dd "FILE")
/**********************************************************************/
/* Allow sysreason=3 to verify SYSOUT DD statements                   */
/**********************************************************************/
          if LRC <> 0 & strip(sysreason,'L',0) <> 3 then
             do
              dderrmsg = errmsg(sigl 'Required DD' dd 'is missing')
              call rcexit LRC dderrmsg sysmsglvl2
             end
          pull tracelvl . module . sigl . sparms
          call modtrace 'STOP' sigl
          interpret 'trace' tracelvl
          return
/*********** @REFRESH END   DDCHECK  2002/09/11 01:08:30 **************/
/*********** @REFRESH BEGIN QDSN     2002/09/11 01:15:23 **************/
/* QDSN     - Make sure there are only one set of quotes              */
/*--------------------------------------------------------------------*/
/* QDSN     - The DSN                                                 */
/**********************************************************************/
qdsn: module = 'QDSN'
       if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
       parse arg sparms
       push trace() time('L') module 'From:' sigl 'Parms:' sparms
       call modtrace 'START' sigl
       parse arg qdsn
       qdsn = "'"strip(qdsn,"B","'")"'"
       pull tracelvl . module . sigl . sparms
       call modtrace 'STOP' sigl
       interpret 'trace' tracelvl
       return qdsn
/*********** @REFRESH END   QDSN     2002/09/11 01:15:23 **************/
/*********** @REFRESH BEGIN TEMPMEM  2004/03/06 13:13:28 **************/
/* TEMPMEM  - EXECIO a stem into a TEMP PDS                           */
/*--------------------------------------------------------------------*/
/* TEMPMEM  - The member to create                                    */
/**********************************************************************/
tempmem: module = 'TEMPMEM'
          if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
          parse arg sparms
          push trace() time('L') module 'From:' sigl 'Parms:' sparms
          call modtrace 'START' sigl
          arg tempmem
/**********************************************************************/
/* Create a unique DD name                                            */
/**********************************************************************/
```

```
          if length(tempmem) < 7 then
             tempdd = '$'tempmem'$'
          else
             tempdd = '$'substr(tempmem,2,6)'$'
/********************************************************************/
/* If TEMPDD exists, the FREE it                                    */
/********************************************************************/
          if listdsi(tempdd 'FILE') = 0 then "FREE F("tempdd")"
/********************************************************************/
/* Generate the TEMPDSN and carve out the DSN                       */
/********************************************************************/
          tempdsn = pandsn(tempmem)
          parse var tempdsn temppds '(' .
/********************************************************************/
/* ALLOCATE the TEMP DD and member                                  */
/********************************************************************/
          call tsotrap "ALLOC F("tempdd") DA("qdsn(tempdsn)") NEW",
                       "LRECL(80) BLKS(0) DIR(1) SPACE(1) CATALOG",
                      "UNIT("@sysda") RECFM(F B)"
/********************************************************************/
/* Write the STEM to the TEMP DD                                    */
/********************************************************************/
          call tsotrap 'EXECIO * DISKW' tempdd '(STEM' tempmem'. FINIS'
/********************************************************************/
/* DROP the stem variable                                           */
/********************************************************************/
          interpret 'drop' tempmem'.'
          pull tracelvl . module . sigl . sparms
          call modtrace 'STOP' sigl
          interpret 'trace' tracelvl
          return tempdd
/*********** @REFRESH END   TEMPMEM  2004/03/06 13:13:28 *************/
/*********** @REFRESH BEGIN BRWSDD   2002/09/11 01:05:08 *************/
/* BRWSDD   - Invoke ISPF Browse on any DD                          */
/*------------------------------------------------------------------*/
/* BRWSDD   - Any DD to browse                                      */
/********************************************************************/
brwsdd: module = 'BRWSDD'
         if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
         parse arg sparms
         push trace() time('L') module 'From:' sigl 'Parms:' sparms
         call modtrace 'START' sigl
         arg brwsdd
         if brwsdd = '' then call rcexit 90 'Browse DD missing'
         call ispwrap "LMINIT DATAID(DATAID) DDNAME("brwsdd")"
/********************************************************************/
/* Browse the VIO dataset                                           */
/********************************************************************/
         call ispwrap "BROWSE DATAID("dataid")"
/********************************************************************/
```

```
/* FREE and DELETE the VIO dataset                                   */
/*********************************************************************/
         call ispwrap "LMFREE DATAID("dataid")"
         call tsotrap "FREE F("brwsdd")"
         pull tracelvl . module . sigl . sparms
         call modtrace 'STOP' sigl
         interpret 'trace' tracelvl
         return
/*********** @REFRESH END   BRWSDD   2002/09/11 01:05:08 *************/
/*********** @REFRESH BEGIN ISPWRAP  2002/09/11 01:11:43 *************/
/* ISPWRAP  - Wrapper for ISPF commands                              */
/*-------------------------------------------------------------------*/
/* VALIDRC  - Optional valid RC from the ISPF command, defaults to 0 */
/* ISPPARM  - Valid ISPF command                                     */
/*********************************************************************/
ispwrap: module = 'ISPWRAP'
          if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
          parse arg sparms
          push trace() time('L') module 'From:' sigl 'Parms:' sparms
          call modtrace 'START' sigl
          parse arg ispparm
          zerrlm = 'NO ZERRLM'
/*********************************************************************/
/* If the optional valid_rc parm is present use it, if not assume 0  */
/*********************************************************************/
          parse var ispparm valid_rc isp_cmd
          if datatype(valid_rc,'W') = 0 then
             do
              valid_rc = 0
              isp_cmd = ispparm
             end
          address ISPEXEC isp_cmd
          IRC = RC
/*********************************************************************/
/* If RC = 0 then return                                             */
/*********************************************************************/
          if IRC <= valid_rc then
             do
              pull tracelvl . module . sigl . sparms
              call modtrace 'STOP' sigl
              interpret 'trace' tracelvl
              return IRC
             end
          else
             do
              perrmsg = errmsg(sigl 'ISPF Command:')
              call rcexit IRC perrmsg isp_cmd strip(zerrlm)
             end
/*********** @REFRESH END   ISPWRAP  2002/09/11 01:11:43 *************/
/*********** @REFRESH BEGIN ISRWRAP  2002/09/11 01:12:34 *************/
```

```
/* ISRWRAP  - Wrapper for ISPF Edit commands                         */
/*-------------------------------------------------------------------*/
/* VALIDRC  - Optional valid RC from the ISPF command, defaults to 0 */
/* ISRPARM  - Valid ISPF Edit command                                */
/*********************************************************************/
isrwrap: module = 'ISRWRAP'
          if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
          parse arg sparms
          push trace() time('L') module 'From:' sigl 'Parms:' sparms
          call modtrace 'START' sigl
          parse arg isrparm
/*********************************************************************/
/* If the optional valid_rc parm is present use it, if not assume 0  */
/*********************************************************************/
          parse var isrparm valid_rc isr_cmd
          if datatype(valid_rc,'W') = 0 then
             do
              valid_rc = 0
              isr_cmd = isrparm
             end
          parse var isr_cmd isr_verb .
          address ISREDIT isr_cmd
          ERC = RC
/*********************************************************************/
/* If RC = 0 then return                                             */
/*********************************************************************/
          if ERC <= valid_rc then
             do
              pull tracelvl . module . sigl . sparms
              call modtrace 'STOP' sigl
              interpret 'trace' tracelvl
              return ERC
             end
          else
             do
              if isr_verb = 'MACRO' & ERC = 20 then
                 do
                  call rcexit ERC 'is an Edit Macro and not valid to',
                             'run outside of ISPF Edit'
                 end
              else
                 do
                  rerrmsg = errmsg(sigl 'ISPF Edit Command:')
                  call rcexit ERC rerrmsg isr_cmd
                 end
             end
/*********** @REFRESH END   ISRWRAP  2002/09/11 01:12:34 *************/
/*********** @REFRESH BEGIN TSOTRAP  2002/12/15 05:18:45 *************/
/* TSOTRAP  - Capture the output from a TSO command in a stem        */
/*-------------------------------------------------------------------*/
```

```
/* VALIDRC  - Optional valid RC, defaults to zero                   */
/* TSOPARM  - Valid TSO command                                     */
/********************************************************************/
tsotrap: module = 'TSOTRAP'
          if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
          parse arg sparms
          push trace() time('L') module 'From:' sigl 'Parms:' sparms
          call modtrace 'START' sigl
          parse arg tsoparm
/********************************************************************/
/* If the optional valid_rc parm is present use it, if not assume 0  */
/********************************************************************/
          parse var tsoparm valid_rc tso_cmd
          if datatype(valid_rc,'W') = 0 then
             do
              valid_rc = 0
              tso_cmd = tsoparm
             end
          call outtrap 'tsoout.'
          tsoline = sigl
          address TSO tso_cmd
          CRC = RC
          call outtrap 'off'
/********************************************************************/
/* If RC = 0 then return                                            */
/********************************************************************/
          if CRC <= valid_rc then
             do
              pull tracelvl . module . sigl . sparms
              call modtrace 'STOP' sigl
              interpret 'trace' tracelvl
              return CRC
             end
          else
             do
              trapmsg = center(' TSO Command Error Trap ',78,'-')
              terrmsg = errmsg(sigl 'TSO Command:')
/********************************************************************/
/* If RC <> 0 then format output depending on environment          */
/********************************************************************/
              if tsoenv = 'BACK' | execenv = 'OMVS' then
                 do
                  say trapmsg
                  do c=1 to tsoout.0
                     say tsoout.c
                  end
                  say trapmsg
                  call rcexit CRC terrmsg tso_cmd
                 end
              else
```

```
/*********************************************************************/
/* If this is foreground and ISPF is available, use the ISPF LOG     */
/*********************************************************************/
                 do
                  if ispfenv = 'YES' then
                     do
                      zedlmsg = trapmsg
/*********************************************************************/
/* Does not call ISPWRAP to avoid obscuring error message modules  */
/*********************************************************************/
                      address ISPEXEC "LOG MSG(ISRZ000)"
                      do c=1 to tsoout.0
                         zedlmsg = tsoout.c
                         address ISPEXEC "LOG MSG(ISRZ000)"
                      end
                      zedlmsg = trapmsg
                      address ISPEXEC "LOG MSG(ISRZ000)"
                      call rcexit CRC terrmsg tso_cmd,
                          ' see the ISPF Log (Option 7.5) for details'
                     end
                  else
                     do
                      say trapmsg
                      do c=1 to tsoout.0
                         say tsoout.c
                      end
                      say trapmsg
                      call rcexit CRC terrmsg tso_cmd
                     end
                 end
             end
/*********** @REFRESH END   TSOTRAP  2002/12/15 05:18:45 *************/
/*********** @REFRESH BEGIN WAIT     2003/05/18 04:03:43 *************/
/* WAIT     - Wait for a specified number of seconds                 */
/*-------------------------------------------------------------------*/
/* SECONDS  - Number of seconds to wait                              */
/* WMODE    - Use any value to stop printing batch wait messages     */
/*********************************************************************/
wait: module = 'WAIT'
       if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
       parse arg sparms
       push trace() time('L') module 'From:' sigl 'Parms:' sparms
       call modtrace 'START' sigl
/*********************************************************************/
       arg seconds wmode
       if datatype(seconds,'W') = 0 then seconds = 10
       RC = syscalls('ON')
/*********************************************************************/
/* If foreground ISPF lock the screen                                */
/*********************************************************************/
```

```
       if tsoenv = 'FORE' & ispfenv = 'YES' then
          call lock seconds 'second wait was requested'
/**********************************************************************/
/* If background, report the wait time                                */
/**********************************************************************/
       if tsoenv = 'BACK' & wmode = '' then
          call saydd msgdd 0 seconds 'second wait was requested'
/**********************************************************************/
/* Call USS SLEEP                                                     */
/**********************************************************************/
       address SYSCALL "SLEEP" seconds
/**********************************************************************/
/* If foreground ISPF lock the screen                                 */
/**********************************************************************/
       if tsoenv = 'FORE' & ispfenv = 'YES' then
          call unlock
       RC = syscalls('OFF')
/**********************************************************************/
       pull tracelvl . module . sigl . sparms
       call modtrace 'STOP' sigl
       interpret 'trace' tracelvl
       return
/*********** @REFRESH END   WAIT     2003/05/18 04:03:43 *************/
/*********** @REFRESH BEGIN SETBORD  2002/09/11 01:16:41 *************/
/* SETBORD  - Set the ISPF Pop-up active frame border colour          */
/*--------------------------------------------------------------------*/
/* COLOR    - Colour for the Active Frame Border                      */
/**********************************************************************/
setbord: module = 'SETBORD'
          if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
          parse arg sparms
          push trace() time('L') module 'From:' sigl 'Parms:' sparms
          call modtrace 'START' sigl
          arg color
/**********************************************************************/
/* Parse and validate colour                                          */
/**********************************************************************/
          if color = '' then color = 'YELLOW'
/**********************************************************************/
/* Build a temporary panel                                            */
/**********************************************************************/
          ispopt11.1=")BODY                        "
          ispopt11.2="%Command ===>_ZCMD        + "
          ispopt11.3=")INIT                        "
          ispopt11.4="&ZCMD = ' '                  "
          ispopt11.5="VGET (COLOR) SHARED          "
          ispopt11.6="&ZCOLOR = &COLOR             "
          ispopt11.7=".RESP = END                  "
          ispopt11.8=")END                         "
/**********************************************************************/
```

```
/* Allocate and load the Dynamic Panel                                */
/**********************************************************************/
          setdd = tempmem('ISPOPT11')
/**********************************************************************/
/* LIBDEF the DSN, VPUT, @TFCOLOR, and run CUAATTR                    */
/**********************************************************************/
          call ispwrap "LIBDEF ISPPLIB LIBRARY ID("setdd") STACK"
          call ispwrap "VPUT (COLOR) SHARED"
          call ispwrap "SELECT PGM(ISPOPT) PARM(ISPOPT11)"
          call ispwrap "LIBDEF ISPPLIB"
          call tsotrap "FREE F("setdd") DELETE"
          pull tracelvl . module . sigl . sparms
          call modtrace 'STOP' sigl
          interpret 'trace' tracelvl
          return
/*********** @REFRESH END   SETBORD  2002/09/11 01:16:41 *************/
/*********** @REFRESH BEGIN LOCK     2003/10/18 09:33:25 *************/
/* LOCK     - Put up a pop-up under foreground ISPF during long waits*/
/*--------------------------------------------------------------------*/
/* LOCKMSG  - Message for the pop-up screen                           */
/**********************************************************************/
lock: module = 'LOCK'
       if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
       parse arg sparms
       push trace() time('L') module 'From:' sigl 'Parms:' sparms
       call modtrace 'START' sigl
       parse arg lockmsg
       if lockmsg = '' then lockmsg = 'Please be patient'
       if tsoenv = 'FORE' then
          do
/**********************************************************************/
/* Use the length of the lockmsg to determine the pop-up size         */
/**********************************************************************/
           if length(lockmsg) < 76 then
              locklen = length(lockmsg) + 2
           else
              locklen = 77
           if locklen <= 10 then locklen = 10
/**********************************************************************/
/* Build a temporary panel                                            */
/**********************************************************************/
           lock.1 = ")BODY EXPAND(//) WINDOW("locklen",1)"
           lock.2 = "%&LOCKMSG                    "
           lock.3 = ")END                         "
/**********************************************************************/
/* Lock the screen and put up a pop-up                                */
/**********************************************************************/
           call ispwrap "CONTROL DISPLAY LOCK"
           call popdyn 'LOCK' 8 execname 'Please be patient'
           lockpop = 'YES'
          end
       pull tracelvl . module . sigl . sparms
       call modtrace 'STOP' sigl
       interpret 'trace' tracelvl
       return
/*********** @REFRESH END   LOCK     2003/10/18 09:33:25 *************/
/*********** @REFRESH BEGIN UNLOCK   2003/10/18 09:33:19 *************/
/* UNLOCK   - Unlock from a pop-up under foreground ISPF              */
/*--------------------------------------------------------------------*/
/* PARM     - N/A                                                     */
/**********************************************************************/
unlock: module = 'UNLOCK'
         if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
         parse arg sparms
         push trace() time('L') module 'From:' sigl 'Parms:' sparms
         call modtrace 'START' sigl
         if tsoenv = 'FORE' then
             do
              if lockpop = 'YES' then
                 do
                  call ispwrap "REMPOP"
                  lockpop = 'NO'
                 end
              if popup = 'YES' then
                 do
                  call setbord 'BLUE'
                  call ispwrap "REMPOP"
                  popup = 'NO'
                 end
             end
         pull tracelvl . module . sigl . sparms
         call modtrace 'STOP' sigl
         interpret 'trace' tracelvl
         return
/*********** @REFRESH END   UNLOCK   2003/10/18 09:33:19 *************/
/*********** @REFRESH BEGIN PANDSN   2002/09/11 01:14:24 *************/
/* PANDSN   - Create a unique PDS(MEM) name for a dynamic panel       */
/*--------------------------------------------------------------------*/
/* PANEL    - Dynamic panel name                                      */
/**********************************************************************/
pandsn: module = 'PANDSN'
         if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
         parse arg sparms
         push trace() time('L') module 'From:' sigl 'Parms:' sparms
         call modtrace 'START' sigl
         arg panel
         unique = right(time('s'),7,0)||'('||panel||')'
         pandsn = userid()||'.'||execname||'.'||panel||'.T'||unique
         pull tracelvl . module . sigl . sparms
         call modtrace 'STOP' sigl
         interpret 'trace' tracelvl
         return pandsn
/*********** @REFRESH END   PANDSN   2002/09/11 01:14:24 *************/
/*********** @REFRESH BEGIN POPDYN   2002/09/11 01:15:11 *************/
/* POPDYN   - Addpop a Dynamic Panel                                  */
/*--------------------------------------------------------------------*/
/* DYN      - Dynamic panel name                                      */
/* DYNROW   - Default row for ADDPOP, defaults to 1                   */
/* DYNMSG   - ADDPOP Window title                                     */
/**********************************************************************/
popdyn: module = 'POPDYN'
         if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
         parse arg sparms
         push trace() time('L') module 'From:' sigl 'Parms:' sparms
         call modtrace 'START' sigl
         parse arg dyn dynrow dynmsg
/**********************************************************************/
/* Set the default ADDPOP row location                                */
/**********************************************************************/
         if dynrow = '' then dynrow = 1
/**********************************************************************/
/* Set the default ADDPOP window title to the current exec name      */
/**********************************************************************/
         if dynmsg = '' then dynmsg = execname
/**********************************************************************/
/* Check if the RETURN option is specified in the DYNMSG             */
/**********************************************************************/
         dynreturn = 'NO'
         if word(dynmsg,1) = 'RETURN' then
            parse var dynmsg dynreturn dynmsg
/**********************************************************************/
/* Allocate and load the Dynamic Panel                                */
/**********************************************************************/
         dyndd = tempmem(dyn)
/**********************************************************************/
/* LIBDEF the POPDYN panel                                            */
/**********************************************************************/
         call ispwrap "LIBDEF ISPPLIB LIBRARY ID("dyndd") STACK"
/**********************************************************************/
/* Change the Active Frame Colour                                     */
/**********************************************************************/
         call setbord 'YELLOW'
/**********************************************************************/
/* set the POPUP variable if this is not a LOCK request               */
/**********************************************************************/
         if dyn = 'LOCK' & popup = 'NO' then
            popup = 'NO'
         else
            popup = 'YES'
/**********************************************************************/
```

```
/* Put up the pop-up                                                 */
/*********************************************************************/
         zwinttl = dynmsg
         call ispwrap "ADDPOP ROW("dynrow")"
         DRC = ispwrap(8 "DISPLAY PANEL("dyn")")
         call ispwrap "LIBDEF ISPPLIB"
         call tsotrap "FREE F("dyndd") DELETE"
/*********************************************************************/
/* Change the Active Frame Colour                                    */
/*********************************************************************/
         call setbord 'BLUE'
/*********************************************************************/
/* Determine how to return                                           */
/*********************************************************************/
         if dynreturn = 'NO' then
            call rcexit DRC 'terminated by user'
         if dynreturn = 'RETURN' & DRC = 8 then
            do
             call ispwrap "REMPOP"
             popup = 'NO'
            end
         pull tracelvl . module . sigl . sparms
         call modtrace 'STOP' sigl
         interpret 'trace' tracelvl
         return DRC
/*********** @REFRESH END   POPDYN   2002/09/11 01:15:11 *************/
/*********** @REFRESH BEGIN SAYDD    2004/03/29 23:48:37 *************/
/* SAYDD    - Print messages to the requested DD                     */
/*-------------------------------------------------------------------*/
/* MSGDD    - DDNAME to write messages to                            */
/* MSGLINES - number of blank lines to put before and after          */
/* MESSAGE  - Text to write to the MSGDD                             */
/*********************************************************************/
saydd: module = 'SAYDD'
        if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
        parse arg sparms
        push trace() time('L') module 'From:' sigl 'Parms:' sparms
        call modtrace 'START' sigl
        parse arg msgdd msglines message
        if words(msgdd msglines message) < 3 then
           call rcexit 33 'Missing MSGDD or MSGLINES'
        if datatype(msglines) <> 'NUM' then
           call rcexit 34 'MSGLINES must be numeric'
/*********************************************************************/
/* If this is not background then bypass                             */
/*********************************************************************/
        if tsoenv <> 'BACK' then
           do
            pull tracelvl . module . sigl . sparms
            call modtrace 'STOP' sigl
            interpret 'trace' tracelvl
            return
           end
/*********************************************************************/
/* Confirm the MSGDD exists                                          */
/*********************************************************************/
        call ddcheck msgdd
/*********************************************************************/
/* If a number is provided, add that number of blank lines before    */
/* the message                                                       */
/*********************************************************************/
        msgb = 1
        if msglines > 0 then
           do msgb=1 to msglines
              msgline.msgb = ' '
           end
/*********************************************************************/
/* If the linesize is too long, break it into multiple lines and     */
/* create continuation records                                       */
/*********************************************************************/
        msgm = msgb
        if length(message) > 60 & substr(message,1,2) <> '@@' then
           do
            messst = lastpos(' ',message,60)
            messseg = substr(message,1,messst)
            msgline.msgm = date() time() strip(messseg)
            message = strip(delstr(message,1,messst))
            do while length(message) > 0
               msgm = msgm + 1
               if length(message) > 55 then
                  messst = lastpos(' ',message,55)
               if messst > 0 then
                  messseg = substr(message,1,messst)
               else
                  messseg = substr(message,1,length(message))
               msgline.msgm = date() time() 'CONT:' strip(messseg)
               message = strip(delstr(message,1,length(messseg)))
            end
           end
        else
/*********************************************************************/
/* Build print lines. Default strips and prefixes date and timestamp */
/* @BLANK - Blank line, no date and timestamp                        */
/* @      - No stripping, retains leading blanks                     */
/* @@     - No stripping, No date and timestamp                      */
/*********************************************************************/
           do
            select
               when message = '@BLANK@' then msgline.msgm = ' '
               when word(message,1) = '@' then
```

```
                    do
                     message = substr(message,2,length(message)-1)
                     msgline.msgm = date() time() message
                    end
               when substr(message,1,2) = '@@' then
                    do
                     message = substr(message,3,length(message)-2)
                     msgline.msgm = message
                    end
               otherwise msgline.msgm = date() time() strip(message)
            end
           end
/**********************************************************************/
/* If a number is provided, add that number of blank lines after     */
/* the message                                                        */
/**********************************************************************/
        if msglines > 0 then
           do msgt=1 to msglines
              msge = msgt + msgm
              msgline.msge = ' '
           end
/**********************************************************************/
/* Write the contents of the MSGLINE stem to the MSGDD               */
/**********************************************************************/
        call tsotrap "EXECIO * DISKW" msgdd "(STEM MSGLINE. FINIS"
        drop msgline. msgb msgt msge
        pull tracelvl . module . sigl . sparms
        call modtrace 'STOP' sigl
        interpret 'trace' tracelvl
        return
/*********** @REFRESH END   SAYDD    2004/03/29 23:48:37 *************/
/*********** @REFRESH BEGIN JOBINFO  2002/09/11 01:12:59 *************/
/* JOBINFO  - Get job related data from control blocks               */
/*--------------------------------------------------------------------*/
/* ITEM     - Optional item number desired, default is all           */
/**********************************************************************/
jobinfo: module = 'JOBINFO'
          if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
          parse arg sparms
          push trace() time('L') module 'From:' sigl 'Parms:' sparms
          call modtrace 'START' sigl
          arg item
/**********************************************************************/
/* Chase control blocks                                               */
/**********************************************************************/
          tcb      = ptr(540)
          ascb     = ptr(548)
          tiot     = ptr(tcb+12)
          jscb     = ptr(tcb+180)
          ssib     = ptr(jscb+316)
          asid     = c2d(stg(ascb+36,2))
          jobtype  = stg(ssib+12,3)
          jobnum   = strip(stg(ssib+15,5),'L',0)
          stepname = stg(tiot+8,8)
          procstep = stg(tiot+16,8)
          program  = stg(jscb+360,8)
          jobdata  = jobtype jobnum stepname procstep program asid
/**********************************************************************/
/* Return job data                                                    */
/**********************************************************************/
          if item <> '' & (datatype(item,'W') = 1) then
             do
              pull tracelvl . module . sigl . sparms
              call modtrace 'STOP' sigl
              interpret 'trace' tracelvl
              return word(jobdata,item)
             end
          else
             do
              pull tracelvl . module . sigl . sparms
              call modtrace 'STOP' sigl
              interpret 'trace' tracelvl
              return jobdata
             end
/*********** @REFRESH END   JOBINFO   2002/09/11 01:12:59 *************/
/*********** @REFRESH BEGIN PTR       2002/07/13 15:45:36 *************/
/* PTR      - Pointer to a storage location                           */
/*--------------------------------------------------------------------*/
/* ARG(1)   - Storage Address                                         */
/**********************************************************************/
ptr: return c2d(storage(d2x(arg(1)),4))
/*********** @REFRESH END   PTR       2002/07/13 15:45:36 *************/
/*********** @REFRESH BEGIN STG       2002/07/13 15:49:12 *************/
/* STG      - Return the data from a storage location                 */
/*--------------------------------------------------------------------*/
/* ARG(1)   - Location                                                */
/* ARG(2)   - Length                                                  */
/**********************************************************************/
stg: return storage(d2x(arg(1)),arg(2))
/*********** @REFRESH END   STG       2002/07/13 15:49:12 *************/
/*********** @REFRESH BEGIN VIODD     2004/04/09 01:16:19 *************/
/* VIODD    - EXECIO a stem into a sequential dataset                 */
/*--------------------------------------------------------------------*/
/* VIODD    - The member to create                                    */
/* VIOLRECL - The LRECL for the VIODD (defautls to 80)                */
/**********************************************************************/
viodd: module = 'VIODD'
        if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
        parse arg sparms
        push trace() time('L') module 'From:' sigl 'Parms:' sparms
        call modtrace 'START' sigl
        arg viodd violrecl viorecfm
        if viodd = '' then call rcexit 88 'VIODD missing'
        if violrecl = '' then violrecl = 8Ø
        if viorecfm = '' then viorecfm = 'F B'
/*********************************************************************/
/* If DD exists, FREE it                                             */
/*********************************************************************/
        if listdsi(viodd 'FILE') = Ø then
           call tsotrap "FREE F("viodd")"
/*********************************************************************/
/* ALLOCATE a VIO DSN                                                */
/*********************************************************************/
        call tsotrap "ALLOC F("viodd") UNIT("@vio") SPACE(1 5)",
                     "LRECL("violrecl") BLKSIZE(Ø) REUSE",
                     "RECFM("viorecfm") CYLINDERS"
/*********************************************************************/
/* Write the stem variables into the VIO DSN                         */
/*********************************************************************/
        call tsotrap "EXECIO * DISKW" viodd "(STEM" viodd". FINIS"
/*********************************************************************/
/* DROP the stem variable                                            */
/*********************************************************************/
        interpret 'drop' viodd'.'
        pull tracelvl . module . sigl . sparms
        call modtrace 'STOP' sigl
        interpret 'trace' tracelvl
        return
/*********** @REFRESH END   VIODD    2ØØ4/Ø4/Ø9 Ø1:16:19 *************/
/*********** @REFRESH BEGIN MODTRACE 2ØØ3/12/31 21:56:54 *************/
/* MODTRACE - Module Trace                                           */
/*-------------------------------------------------------------------*/
/* TRACETYP - Type of trace entry                                    */
/* SIGLINE  - The line number called from                            */
/*********************************************************************/
modtrace: if modtrace = 'NO' then return
           arg tracetyp sigline
           tracetyp = left(tracetyp,5)
           sigline = left(sigline,5)
/*********************************************************************/
/* Adjust MODSPACE for START                                         */
/*********************************************************************/
           if tracetyp = 'START' then
              modspace = substr(modspace,1,length(modspace)+1)
/*********************************************************************/
/* Set the trace entry                                               */
/*********************************************************************/
           traceline = modspace time('L') tracetyp module sigline sparms
/*********************************************************************/
/* Adjust MODSPACE for STOP                                          */
```

```
/**********************************************************************/
           if tracetyp = 'STOP' then
              modspace = substr(modspace,1,length(modspace)-1)
/**********************************************************************/
/* Determine where to write the traceline                             */
/**********************************************************************/
           if ispfenv = 'YES' & tsoenv = 'FORE' then
/**********************************************************************/
/* Write to the ISPF Log, do not use ISPWRAP here                     */
/**********************************************************************/
              do
               zedlmsg = traceline
               address ISPEXEC "LOG MSG(ISRZ000)"
              end
           else
              say traceline
/**********************************************************************/
/* SAY to SYSTSPRT                                                    */
/**********************************************************************/
           return
/*********** @REFRESH END   MODTRACE 2003/12/31 21:56:54 *************/
```

Robert Zenuk
Systems Programmer (USA)

Enhancing mainframe-to-mainframe Enterprise Extender (EE)

Enterprise Extender (EE), an integral component within all of IBM's Communication Server 'gateway' and PComm emulation offerings, has, since 1996, always been (without exception) IBM's recommendation for routing SNA/APPN traffic across IP networks – whether they be intranets, VPNs, extranets, or even the Internet. Rather than in anyway being a partisan marketing-oriented posture, this unwavering endorsement of EE by IBM is based on multiple, extremely germane, SNA-centric, technical factors.

For a start, EE is the only SNA-over-IP transport scheme that supports *bona fide* SNA LU name-based, end-to-end routing (ie the so-called 'SNA routing' performed by SNA subarea nodes, in particular ACF/NCP Type 4 nodes). Hence IBM's partiality for EE over DLSw and AnyNet, neither of which supports SNA routing. Furthermore, DLSw, the router-oriented mechanism for transporting SNA/APPN across IP backbones, is not available on mainframes. Thus, DLSw-based SNA-over-IP transport can, at best, only start at a channel-attached router. This precludes QDIO mode operation with Gigabit Ethernet OSA-Express ports, the fastest and most efficient of modern mainframe I/O mechanisms, given that QDIO supports only IP-based traffic. To get around this limitation, DLSw fans, ironically, recommend using EE between the channel-attached router and the mainframe – given that Cisco routers also support EE, which is now an Internet standard in the form of RFC 2353, though they call it SNASw. EE, in contrast to DLSw and AnyNet, also supports SNA CoS traffic prioritization.

When combined with the Extended Border Node functionality (another standard feature of Communications Server for z/OS), EE is also the only IBM-mandated solution, at present, for replacing some 3745/3746-centric SNI configurations with IP-based networking across OSA-Express adapters. This is clearly reiterated in IBM's June 2004 *Statement of Direction Update on SNA Support*. The bottom line here is that given all these EE-only capabilities when it comes to SNA-over-IP, it is not surprising that IBM consistently refers to EE as offering the best of both worlds when it comes to SNA/APPN and IP.

SO SHOULDN'T EE BE MORE POPULAR?

Despite its many incontrovertible strengths, and IBM's relentless backing of it, EE is not nearly as prevalent in the TCP/SNA world as one would expect. Given that TCP/SNA professionals are invariably pragmatic and pedantic (to a fault) when it comes to networking technology, there have to be some very valid reasons why they continue to drag their heels when it comes to embracing EE – regardless of IBM's efforts to encourage more widespread usage. In the case of enterprises migrating to SNA-over-IP, there are four key reasons why TCP/SNA professionals have shied away from EE to date. These are:

1 Since EE is HPR-based, using EE to route SNA-over-IP requires the definition and activation of appropriate APPN nodes as well as DLUS/DLUR functionality.

2 DLSw, despite its limitations *vis-à-vis* SNA routing, SNA CoS, and QDIO, is a well understood, highly proven, and relatively easily implemented mechanism that does not require any changes to the existing SNA-related definitions – and certainly does not require the implementation of APPN nodes.

3 EE does not use TCP – thus making firewall traversal, network management, and SSL-based security both bifurcated and difficult.

4 EE does not support multiple IP stacks within the same LPAR, thus foiling large customers who now prefer to segment their IP-centric network into multiple operational sub-nets (eg each serving a particular corporate division or extranet population) using separate IP stacks for each sub-net.

While the first two objections above will continue to hold sway when it comes to 'branch office' networks, EE, despite its APPN requirements, is still the only real option for optimal mainframe-to-mainframe networking – whether it be cross-domain or SNI. The primary reason for this is the lack of DLSw functionality on mainframes. The second reason is that throughput is invariably a requirement when it comes to inter-mainframe networking and QDIO mode is the best way to realize such performance. So one way or another, with 37*xx*s now officially discontinued and 37*xx* spares getting scarce, TCP/SNA professionals looking at mainframe-to-mainframe connections are being forced to consider EE – particularly as an SNI replacement scheme.

It is at this juncture that the last two EE-related objections raise their head. Fortunately there is, at last, an attractive solution for both of these problems when it comes to mainframe-to-mainframe EE: the recently-announced (August 2004) Apias 3.1 offering from William Data Systems (www.willdata.com). Many of you are no doubt already familiar with William Data Systems thanks to the popularity of their mainframe-based EXIGENCE and IMPLEX network management offerings and the innovative RouteView tool for graphically managing ACF/VTAM and ACF/NCP Path Tables.

APIAS 3.1 ADDS MUCH-NEEDED FEATURES

Apias 3.1, which works alongside EE (as peer EE node), in essence imbues IBM's standard Communications Server for z/OS-based EE with three additional features:

1 Transportation of EE traffic, mainframe-to-mainframe, using TCP/IP – rather than the User Datagram Protocol (UDP) normally used by EE.

2 End-to-end authentication and data encryption using SSL with digital certificates.

3 Ability to use multiple IP stacks with EE, thus enabling customers to securely partition their IP networks into sub-nets.

Of these features, the pivotal one is that of TCP/IP enabling EE. There are four key reasons why it is often best to use EE with TCP/IP, instead of EE's native UDP. These are:

1 Controlled firewall traversal through already open and closely monitored TCP ports.

2 Ensuring that EE connections and traffic can be continually monitored and managed alongside other TCP/IP traffic, in particular tn3270(E), HTTP, and FTP, rather than as a separate EE-specific protocol.

3 The ability to use SSL's end-to-end security.

4 Reliance on a connection-oriented, guaranteed delivery mechanism in line with the traditional data link schemes hitherto used for SNA connections.

USING TCP RATHER THAN UDP IS PIVOTAL

EE opting to use UDP rather than TCP can be attributed to a fit of 'irrational exuberance' on the part of some well-intentioned technocrats in what were the early and heady days of SNA/IP integration – when all today's headaches relating to security and management were still way out on the horizon. The preference for UDP stems from the fact that HPR, in marked contrast to SNA and APPN, has a powerful, connectionless, Layer 2 routing mechanism known as Automatic Network Routing (ANR). The architects of EE wanted to highlight and preserve this connectionless routing theme of HPR – given that EE is meant to be HPR-over-IP.

UDP is IP's connectionless datagram mechanism. TCP, on the other hand is IP's connection-oriented (ie session-based), guaranteed-delivery transport mechanism. UDP works and works well. But it is not TCP/IP and that is the rub.

Many corporations, quite rightly, are hesitant to open the UDP-related ports (ie ports 12000 to 12004) on their firewalls. UDP, as a connectionless protocol, is an ideal means for delivering malicious payloads and mounting denial-of-service attacks. EE's reliance on UDP was thus a major impediment when it came to controlled firewall traversal. This is now no longer an issue with Apias 3.1.

TCP/IP is also, indubitably, the most widely used of the Internet protocols given that it is the basis for HTTP(S), FTP, SMTP/POP (ie e-mail), telnet, and tn3270(E)/tn5250. TCP/IP is the protocol that network professionals know best. It is the protocol that they continually monitor and manage using the IP monitors. It is the TCP/IP activity window that invariably has pride of place within most IP monitoring stations.

While high-end IP monitors do indeed provide unstinted support for UDP, it is still the case that UDP is a marginal minority protocol compared with TCP. With Apias 3.1, EE no longer has to be dependent on UDP. Instead EE can join the mainstream by using TCP (and SSL) like all the other major corporate applications.

BOTTOM LINE

While DLSw is likely to prevail when it comes to branch networks, EE indubitably is the optimum solution for mainframe-to-mainframe SNA-over-IP connections, whether it be SNI or just straightforward cross-domain. EE's use of UDP was a problem particularly when it came to security and network management. Apias 3.1 is a way to easily overcome these hitherto frustrating limitations of EE. Apias 3.1 is an ACF authorized z/OS task that runs in its own address space. All of its networking functions are realized using a standard IBM IP stack. Apias 3.1 is a well-behaved 'sockets' application that, moreover, relies exclusively on documented and open IBM APIs. Thus, Apias 3.1 in no way compromises the integrity, reliability, or stability of EE, IBM's Communications Server, or IBM IP stacks. Therefore, if you are planning to use EE for mainframe-to-mainframe communications, you should definitely evaluate Apias 3.1 as a must-have adjunct to EE.

Anura Gurugé
Strategic Consultant (USA)

FTP tool for mass data transfer

Data transfer from mainframe to PC or any server location is a time-consuming process. If the volume of data is huge, it may take as much as several hours to transfer. In this article I have outlined the difficulties associated with the conventional data transfer facility and an approach through which one can cut the data transfer time substantially.

In on-line mode, file transfer from a mainframe is accomplished using the IND$FILE transfer facility provided by the ISPF command shell. Typically, all mainframe emulators provide an interface to transfer files between the host (mainframe) and PC. This facility is good when the amount of data to transfer is small. With large amounts of data to transfer it poses two major issues:

1 The time taken to transfer the data is enormously large because the rate of data transfer through IND$FILE is very low.

2 The TSO terminal/ emulator remains locked as long as the transfer is in progress.

Both the above issues could be resolved using customized FTP jobs in online or batch mode. The customized FTP job does not require the use of a mainframe emulator. Consequently, the issue of a locked emulator session never happens. It connects to the mainframe by establishing a normal FTP session from a DOS command prompt. One can write a customized FTP tool in any text-processing editor available on one's PC. It could then be executed on any other PC provided the PC is connected to the LAN. The whole idea is that the mainframe connection must be available on the LAN for FTP to work properly.

Writing customized FTP jobs is a simple task. The first step is to write an actual FTP download or upload job. The steps below describe how to write the FTP download/upload job:

1 Open a fresh file in Notepad or any other text-processing editor.

2 Put the following lines into it:

```
OPEN <IP address of the mainframe OR its alias>
<USERID>
<Password>
```

Please note that the correct contact address for Xephon Inc is PO Box 550547, Dallas, TX 75355, USA. The phone number is (214) 340 5690, the fax number is (214) 341 7081, and the e-mail address to use is info@xephon.com.

```
GET <Mainframe file name to be transferred to PC> C:\FILE1.TXT
PUT <Mainframe file name to be received from PC> C:\FILE2.TXT
BYE
```

Here, replace ***<**IP address of the mainframe OR its alias>* with the actual IP address of the mainframe. Similarly, replace *<USERID>* and *<Password>* with the USERID and password that you use to log-on to a mainframe session.

GET is an FTP command to transfer the data from the host to a PC. Replace *<Mainframe file name to be transferred to PC>* with an actual file name, within quotes. PUT is just the opposite of GET. You can use multiple GET and PUT statements in the same FTP job to transfer multiple files.

The BYE command terminates the session at the end of data transfer. If you do not want to terminate the session automatically, do not write the BYE command.

The job will download the files to the C: drive of your PC. If required, change the destination appropriately.

3 Save the file with a .TXT extension. You can give this file any name you wish. Let's assume it is called loginftp.txt.

The second step is to write the driver batch program that would trigger the actual FTP job. The steps below describe how to write to write a driver program:

1 Open a fresh file in Notepad or any other text-processing editor.

2 Put the following line into it:

```
ftp -s:loginftp.txt
```

Note that loginftp.txt is the same file that we just created.

3 Save the file with a .BAT extension. You can give this file any name you wish. Let's assume it is called DataTransfer.BAT.

Now the customized FTP job is ready. Simply double-click the DataTransfer.BAT file. It will establish the FTP session with the mainframe and start uploading or downloading the data until the data transfer completes.

In an experiment that involved the transfer of files from a mainframe to a PC using the conventional method and the FTP job, we found that the latter to be 800 times faster than the former.

Yash Pal Samnani
Program Analyst
Infosys Technologies Limited (USA)

Using FTP trace for diagnostics

The FTP trace can let you, at a diagnostic location at the central site, view FTP commands entered from a remote location. This can be quite useful when helping users in diagnosing problems. They may have received error codes or entered commands improperly. You just have to tell them to re-create the problem scenario, and this time you can watch what they are doing as they do it.

The way you do this is by using the FTP trace facility. The first thing that has to be done is to set up SYSLOGD.

USING SYSLOGD

The syslog daemon (SYSLOGD) can capture informational or debugging messages about the operation of system functions or socket applications. You can run the syslog daemon as a started task, bring it up under Unix System Services or even as a job. You must configure SYSLOGD using a configuration file such as the one shown below:

```
(C) COPYRIGHT International Business Machines Corp. 1995,1999
# All Rights Reserved
# Licensed Materials - Property of IBM
```

```
#
# /etc/syslog.conf - control output of syslogd
#
# A # sign denotes a comment.
# Each line of this file must contain at least two parts:
# A message source list and a destination
#
(Several comment lines have been deleted here)

# The following has been uncommented and can be used in an
# actual installation.  The target output file MUST EXIST before
# the syslog deamon is started.
#
# Log all entries to /tmp/syslog.log
   *.*          /tmp/syslog.log
   *.info       /dev/console
#
```

Notice the statements:

```
*.*          /tmp/syslog.log
*.info       /dev/console
```

The statement '*.* /tmp/syslog.log' will route all messages to the file /tmp/syslog.log. This file must exist before starting SYSLOGD. The statement '*.info   /dev/console' will route all messages with a priority of information or above. Syslog messages can vary from a priority of critical to debug. These priorities are defined in RFC3164.

CAUTION!

Make sure that you do not have debug messages going to the console from syslog! The routing should be priority information or higher. If you turn on FTP tracing and you are routing all messages to the console, you will have a very high risk of flooding the console and causing serious system problems.

The configuration parameter shown below in the syslog configuration file controls the types of message that will go to the console:

```
*.info       /dev/console
```

OPERATIONAL CONSIDERATIONS

Starting an FTP trace varies from z/OS to OS/390. For z/OS, to start tracing the FTP task on the mainframe, enter from the system console:

```
MODIFY ftptaskname,DEBUG=(CMD,USERID(user id*))
```

for the FTP server and user id you wish to trace. For example:

```
MODIFY FTPD1,DEBUG=(CMD,USERID(WLJ*))
```

For OS/390, use the command:

```
MODIFY ftptaskname,UTRACE=user id
```

After the trace has been successfully started, you may ask the user to use FTP so you can watch it for any problems.

Finally, stop the FTP trace. For z/OS, enter from the system console:

```
MODIFY ftptaskname,NOTRACE
```

For OS/390, the command is:

```
MODIFY ftptaskname,NOUTRACE
```

FTP LOGGING WITH Z/OS 1.4

With z/OS 1.4, the FTPLOGGING statement in FTP.DATA for the FTP server can be used to indicate whether the FTP server should log FTP server activity. The following types of activity are logged:

- Connectivity
- Authentication
- Access
- Allocation.

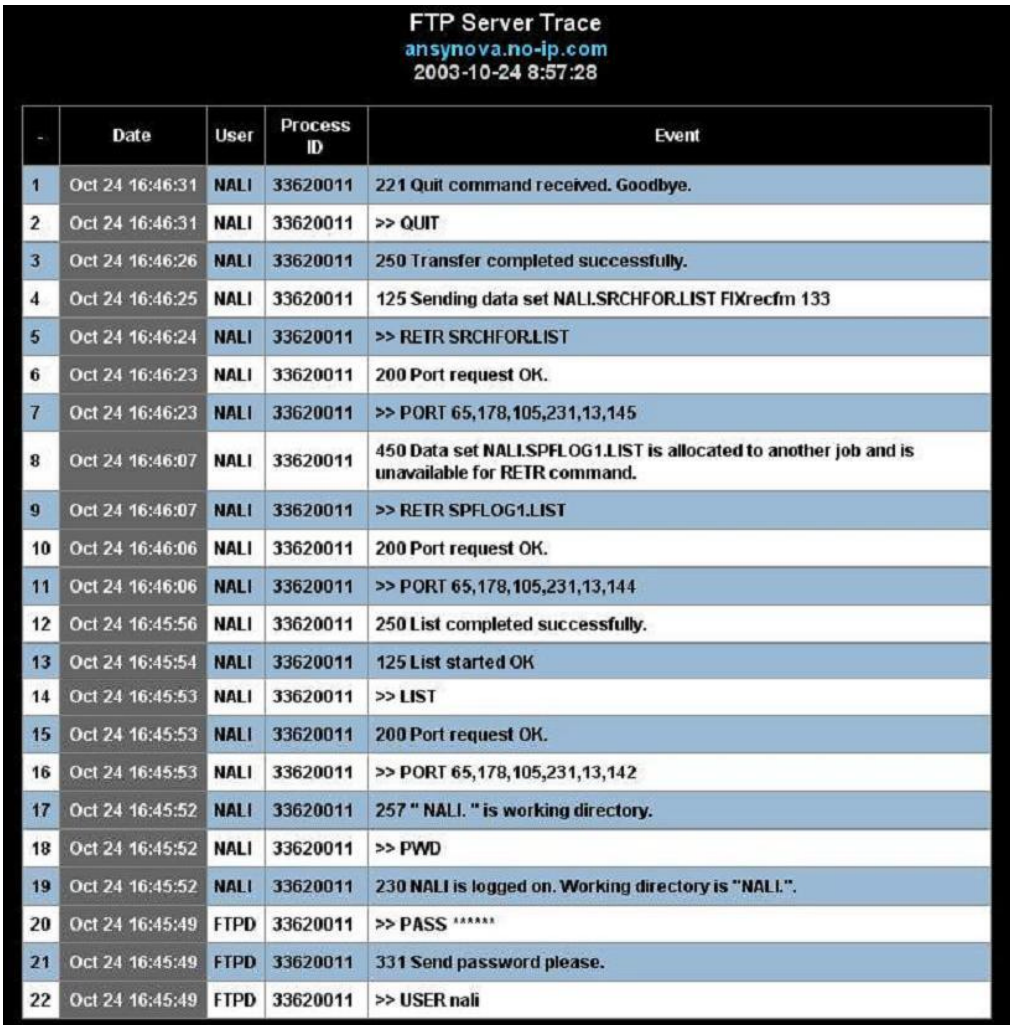
These messages are similar to the FTP trace statements.

The drawback to this method is that you must be on at least z/OS 1.4. Many installations are not yet at this release level.

You must also create logs for all users – this could create a great deal of data. Lastly, this feature cannot be turned on and off dynamically. One could modify the FTP.DATA and then restart the FTP server to achieve dynamic logging – but this seems a little brutal.

FTP SERVER TRACE

Now, how can you actually use the FTP trace? Figure 1 shows a sample FTP server trace with two attempts at file



FTP Server Trace
ansynova.no-ip.com
2003-10-24 8:57:28

-	Date	User	Process ID	Event
1	Oct 24 16:46:31	NALI	33620011	221 Quit command received. Goodbye.
2	Oct 24 16:46:31	NALI	33620011	>> QUIT
3	Oct 24 16:46:26	NALI	33620011	250 Transfer completed successfully.
4	Oct 24 16:46:25	NALI	33620011	125 Sending data set NALI.SRCHFOR.LIST FIXrecfm 133
5	Oct 24 16:46:24	NALI	33620011	>> RETR SRCHFOR.LIST
6	Oct 24 16:46:23	NALI	33620011	200 Port request OK.
7	Oct 24 16:46:23	NALI	33620011	>> PORT 65,178,105,231,13,145
8	Oct 24 16:46:07	NALI	33620011	450 Data set NALI.SPFLOG1.LIST is allocated to another job and is unavailable for RETR command.
9	Oct 24 16:46:07	NALI	33620011	>> RETR SPFLOG1.LIST
10	Oct 24 16:46:06	NALI	33620011	200 Port request OK.
11	Oct 24 16:46:06	NALI	33620011	>> PORT 65,178,105,231,13,144
12	Oct 24 16:45:56	NALI	33620011	250 List completed successfully.
13	Oct 24 16:45:54	NALI	33620011	125 List started OK
14	Oct 24 16:45:53	NALI	33620011	>> LIST
15	Oct 24 16:45:53	NALI	33620011	200 Port request OK.
16	Oct 24 16:45:53	NALI	33620011	>> PORT 65,178,105,231,13,142
17	Oct 24 16:45:52	NALI	33620011	257 " NALI. " is working directory.
18	Oct 24 16:45:52	NALI	33620011	>> PWD
19	Oct 24 16:45:52	NALI	33620011	230 NALI is logged on. Working directory is "NALI.".
20	Oct 24 16:45:49	FTPD	33620011	>> PASS ******
21	Oct 24 16:45:49	FTPD	33620011	331 Send password please.
22	Oct 24 16:45:49	FTPD	33620011	>> USER nali

Figure 1: Sample FTP server trace

transmission. An FTP server trace allows you to view any commands the user enters at their remote PC as they enter them. This will allow you to diagnose the problems the user is experiencing when submitting FTPs. You will be able to see exactly what files they are trying to send or receive. You can also view any error messages they receive.

Then we created the output shown in Figure 1 with some technology that we use. You can also browse the syslog output for these messages.

The log is presented in reverse chronological order. Some interesting events in the log are as follows:

- Entry # 22 – user NALI starts a new FTP session.
- Entry # 20 – the password is successfully entered.
- Entries # 19–10 – the user gets to the desired file to transmit.
- Entry #9 – the user asks for file SPFLOG1.LIST.
- Entry #8 – the file is being used by another program. The return code 450 is returned.
- Entry #5 – the user asks for file SRCHFOR.LIST.
- Entry #3– the file SPFLOG1.LIST is transmitted successfully.
- Entry #1 – the FTP session is terminated.

SUMMARY

The FTP trace can be quite useful in assisting remote users from the central site. Simply remember a few commands and then browse the syslog output!

Nalini Elkins
Inside Products (USA)

March 2001–December 2004 index

Items below are references to articles that have appeared in *TCP/SNA Update* since ssue 41, March 2001. References show the issue number followed by the page number(s). Subscribers can download copies of all issues in Acrobat PDF format from Xephon’s Web site.

Computer Associates has announced the availability of new Measured Workload Pricing for the company's mainframe management software products. This combination of usage-based pricing and best practices management capabilities enables CA customers to maximize the performance of their data centre investments.

Measured Workload Pricing bases licensing costs on measured system usage (rather than total potential hardware capacity) enabling customers to align IT spending with changing business requirements. Customers can establish licence costs based on measured levels of usage, while reserving the ability to increase usage as needed to support growing demand.
Measured Workload Pricing is based on quarterly reports generated by IBM's native Sub-Capacity Reporting Tool (SCRT).

The solution sets include network management, which is said to maximize operational performance through the integrated management of SNA and TCP/IP networks, overcoming fragmented management processes that diminish effectiveness and drive up the cost of network ownership.

For further information contact:
Computer Associates One Computer Associates Plaza, Islandia, NY 11749, USA.
Tel: (631) 342 6000.
URL: ca.com/mainframe/mwp.

* * *

Aton International has updated its TN3270 terminal emulation package for Pocket PCs running Windows Mobile and Pocket PC Phone Edition. The product uses the TN3270 terminal protocol to connect to CICS, VM/CMS, or MVS. The new version adds full-screen landscape mode, soft keyboard, and session switching, allowing access to mainframe and Web services data.

For further information contact:
Aton International, 731 Morgan Place, Los Altos, CA 94024-4901, USA.
Tel: (650) 938 9328.
URL: www.aton.com/Products/products.htm.

* * *

Attachmate has announced 'next generation' emulation with EXTRA! Mainframe Server Edition Version 8.0 for Windows XP.

The product offers a security framework to safeguard mainframes, centralized management from anywhere on the network (including the mainframe), portal integration of host access clients, and integration with Microsoft Office applications to boost user productivity.

Because mainframes participate in services-oriented initiatives that expose them to new threats, the product adds enterprise security.

For further information contact:
Attachmate, 3617 131st Avenue SE, Bellevue, WA 98006, USA.
Tel: (425) 644 4010.
URL: www.attachmate.com/extraeval.

* * *